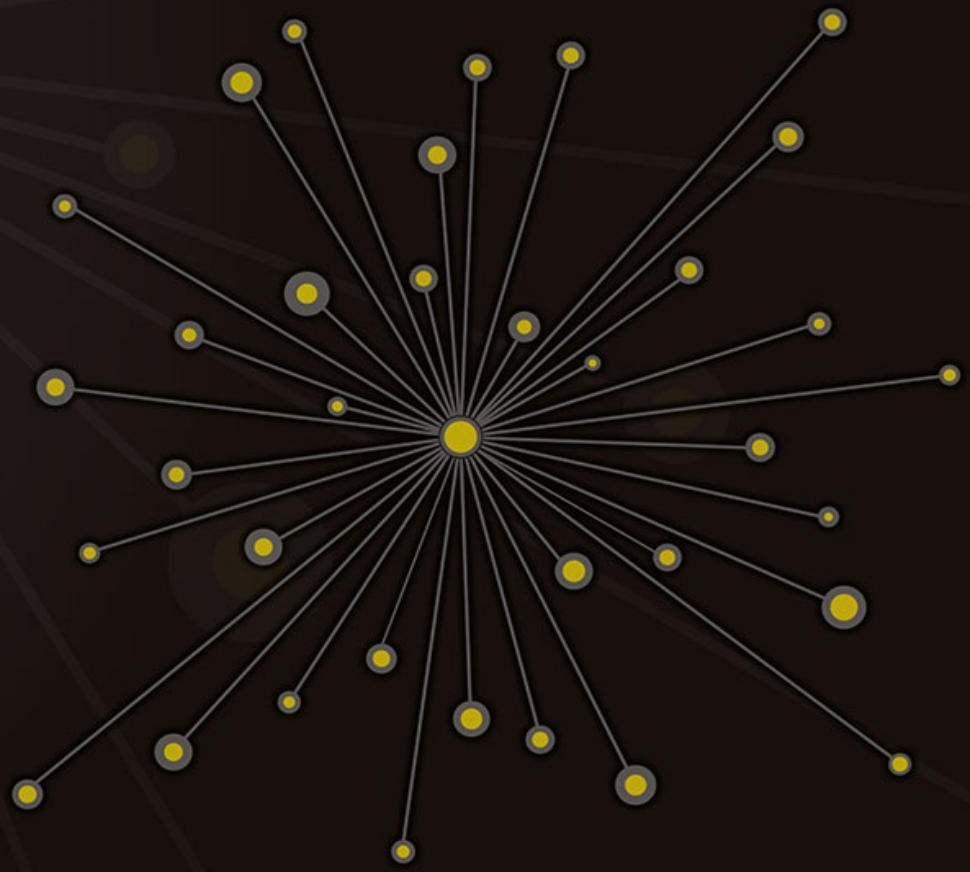
ELSEVIER

DOUGLAS E. COMER

Interligação de Redes com TCP/IP

Princípios, protocolos e arquitetura

Tradução da 6ª Edição



CAMPUS

Volume 1

INTERLIGAÇÃO DE REDES COM TCP/IP

PRINCÍPIOS, PROTOCOLOS E ARQUITETURA

DOUGLAS E. COMER

Volume 1

6ª edição

Tradução de Tássia Fernanda Alvarenga

Revisão Técnica de Sérgio Guedes

Do original: *Internetworking with TCP/IP: Vol I: Principles, Protocols, and Architecture. 6th edition.* Tradução autorizada do idioma inglês da edição publicada por Pearson Education, Inc. Copyright © 2014, 2006, 2000 Pearson Education, Inc.

Copidesque: Isis Batista Pinto
Revisão: Casa Editorial BBM
Editoração Eletrônica: Estúdio Castellani
Freitas Bastos: Freitas Bastos

Elsevier Editora Ltda.
Conhecimento sem Fronteiras
Rua Sete de Setembro, 111 – 16º andar
20050-006 – Centro – Rio de Janeiro – RJ – Brasil

Rua Quintana, 753 – 8º andar
04569-011 – Brooklin – São Paulo – SP – Brasil

Serviço de Atendimento ao Cliente
0800-0265340
atendimento1@elsevier.com

ISBN: 978-85-352-7863-7
ISBN (versão digital): 978-85-352-7864-4
ISBN (edição original): 978-0-13-608530-0

CIP-Brasil. Catalogação na Publicação

Sindicato Nacional dos Editores de Livros, RJ

C725i Comer, Douglas E.
Interligação de redes com TCP/IP / Douglas E. Comer; tradução Tássia Fernanda Alvarenga. – 6. ed. – Rio de Janeiro: Elsevier, 2015.
il.; 28 cm.

Tradução de: Internetworking with TCP/IP, v. 1, 6th edition
ISBN 978-85-352-7863-7

1. TCP/IP (Protocolo de rede de computador). I. Título.

14-17850

CDD: 004.6
CDU: 004.7

Para Chris

Sobre o autor

Dr. Douglas Comer, ilustre professor de Ciência da Computação na Universidade de Purdue e ex-vice-presidente de Pesquisa da Cisco, é um especialista reconhecido internacionalmente em interligação de redes de computadores, em protocolos TCP/IP e em Internet. Autor de inúmeros artigos de referência e livros técnicos, é um pioneiro no desenvolvimento de currículos de laboratórios para pesquisa e educação.

Autor prolífico, os livros populares de Comer foram traduzidos para mais de quinze línguas e são usados na indústria, assim como na ciência da computação, na engenharia e em departamentos comerciais ao redor do mundo. Sua série de três volumes *Interligação de redes com TCP/IP* é um marco e revolucionou as redes e seu ensino. Seus livros didáticos e manuais de laboratório inovadores moldaram e continuam a moldar currículos de graduação e pós-graduação.

A precisão e o discernimento dos livros do Dr. Comer refletem sua extensa base de conhecimento de sistemas computacionais. Sua pesquisa abrange tanto hardware como software. Ele criou um sistema operacional completo, escreveu drivers de dispositivo e implementou software de protocolo de rede para computadores convencionais e para processadores de rede. O software resultante tem sido usado pela indústria em diversos produtos.

Comer criou cursos em que hoje leciona sobre protocolos de rede e tecnologias de computador para diversos públicos, incluindo engenheiros e acadêmicos. Seus laboratórios educacionais inovadores permitem a ele e a seus alunos criarem e implementarem protótipos funcionais de grandes e complexos sistemas, medindo o desempenho dos protótipos resultantes. Ele continua a ensinar em indústrias, universidades e participa de conferências no mundo todo. Além disso, Comer presta consultoria em projeto de redes e sistemas de computador a indústrias.

Por mais de dezoito anos, o professor Comer trabalhou como editor chefe do periódico de pesquisa *Software – Practice and Experience*. É Fellow da ACM, Fellow da Purdue Teaching Academy e recebedor de diversos prêmios, incluindo um prêmio Usenix Lifetime Achievement.

Outras informações poderão ser encontradas em:

www.cs.purdue.edu/people/comer

E informações sobre os livros podem ser encontradas em:

www.comerbooks.com

Prólogo

É um raro prazer ser convidado a escrever um prefácio para a sexta edição do livro já clássico de Doug Comer sobre TCP/IP e Internet. Em 2012, havia quase 3 bilhões de pessoas *on-line*. Algo como 6,5 bilhões de telefones móveis estavam em uso, e a maior parte deles era formada por "smartphones" com acesso à Internet via rede wireless e Wi-Fi. Na verdade, os sistemas sem fio desviam o tráfego para o Wi-Fi quando isso é possível, a fim de verter carga. Os dados mais recentes da TeleGeography são de que 77 Tbps (terabits por segundo) fluem através da Internet. Um componente substancial do tráfego é o vídeo, mas, cada vez mais, são transferidos grandes arquivos de dados que contêm informação de sequência genética, dados de telescópios, sistemas de sensores, do Grande Colisor de Hádrons e de outros instrumentos científicos.

Aprendemos muito sobre o TCP/IP em muitos contextos, e este texto reúne muito desse conhecimento. Aprendemos que a memória buffer pode não ser nossa amiga se grandes quantidades dela estiverem alocadas em um dispositivo de rede, em que há uma grande queda na capacidade. Este é o chamado problema "buffer bloat", descrito no Capítulo 11. Quando uma ligação de alta velocidade encontra um link de baixa velocidade, grandes buffers levam um longo tempo para esvaziar indo no sentido de baixa velocidade, o que tem o efeito de aumentar a demora e afeta o controle de fluxo TCP, gerando congestionamento grave, com todo os seus efeitos colaterais negativos. Aprendemos também que existem condições sob as quais o TCP/IP trabalha de forma ineficiente..Estou pensando aqui em ambientes altamente instáveis e com atrasos variáveis. Exemplos incluem a comunicação interplanetária e comunicações táticas (incluindo móvel, bem como militar). Para essas condições, os novos tipos de protocolos, chamados de "atraso e interrupção de rede tolerante" (DTN), são necessários para complementar coisas como TCP. Na verdade, o DTN pode

ser executado por meio de TCP ou UDP ou virtualmente por qualquer outro subsistema de transmissão. Os protocolos que implementam esse tipo de rede já estão em uso na Estação Espacial Internacional e nos robôs, agora, em Marte!

Novas ideias, tais como Software Defined Networking e o protocolo OpenFlow da Universidade de Stanford, descritos no Capítulo 28, também estão colorindo o futuro da Internet. Embora esses sistemas possam operar para suportar arquiteturas de Internet convencionais, eles também são capazes de ir além das noções convencionais de endereçamento para suportar roteamento baseado em conteúdo, entre outras coisas. Gestão de fluxos fim a fim funciona bem com esses sistemas. Além disso, parece oportuno revisitar a comunicação sem fio e perguntar como os modos de transmissão poderiam influenciar maior evolução da Internet. Imagine satélites "chovendo" IP ou baixando UDP packets em centenas de milhões de receptores. Em contextos terrestres, a capacidade de irradiar 360 graus permite que vários receptores recebam uma transmissão. Avanços no compartilhamento do espectro e no uso de antenas beamforming fazem desta uma área ainda mais rica e interessante para explorar.

A Internet continua a se expandir e a mudar de forma inesperada. Além de dispositivos que os humanos usam, uma nova onda de sensores, câmeras e atuadores está sendo conectada, o que nos dará acesso a controle remoto de tudo, de dados científicos a luzes em um prédio e a processos de fabricação. Referimo-nos aos novos dispositivos como uma Internet das coisas; eles estão descritos no Capítulo 30.

Como acho que este livro demonstra amplamente, a Internet ainda é emocionante. É preciso que muita pesquisa seja realizada em apoio a aplicações novas e desafiadoras, oportunidades de crescimento colaborativo no dia a dia.

Bem-vindo à Internet do século 21, época em que a inovação ainda é a ordem do dia. Este livro fornece a base de que você precisa para entender e participar.

Vint Cerf,
Internet Evangelist, Google
Presidente, ACM. Março, 2013.

Prefácio

Interligação de redes e TCP/IP agora dominam toda a rede – até mesmo as empresas de telefonia que antes eram bastiões exclusivas de redes com circuito comutado adotaram a tecnologia IP. Mais duas mudanças revolucionárias estão ocorrendo que contam com internetworking: o paradigma de computação em nuvem e a Internet das coisas. No modelo de nuvem, computação e armazenamento são realizados em centros de dados em nuvem. Os usuários contam com a Internet para fazer upload, download, acessar suas informações e compartilhar dados com outras pessoas. A expressão *Internet das coisas* é utilizada para caracterizar uma Internet de dispositivos com inteligência incorporada que atuam de forma autônoma, em vez de dispositivos, como smartphones e laptops, que são operados por um ser humano. Usar a tecnologia da Internet permite aos dispositivos embarcados a comunicação com os servidores remotos, bem como entre si; a infraestrutura cibernética resultante já inclui dispositivos em residências, escritórios e lojas, bem como sensores que medem as condições ambientais e estruturas civis, como pontes e barragens.

Muitos leitores pediram que o texto fosse atualizado para refletir as mudanças recentes, sugerindo temas e ênfases específicos. Vinte anos após sua invenção, o IPv6 finalmente está ganhando aceitação. Voz e vídeo substituíram transferência de arquivos como os principais usos da Internet. A sexta edição atende às sugestões dos leitores por meio da reorganização e atualização de capítulos existentes e da introdução de material novo. Em particular, capítulos sobre as primeiras aplicações de Telnet e FTP foram eliminados para abrir espaço para material mais recente. Um novo capítulo sobre a Internet das coisas considera o uso do TCP/IP em uma rede de sensores sem fio. Um novo capítulo sobre redes definidas por software examina o uso de OpenFlow que, embora não seja um padrão IETF, se tornou uma parte importante na gestão de rede e de Internet.

Para atender a um pedido muitas vezes repetido, o capítulo sobre camadas de protocolos foi movido mais para o início do texto. Os instrutores são avisados, no entanto, de que a estratificação em camadas não é uma arquitetura rígida que explica todos os protocolos. Os alunos devem vê-la como uma diretriz básica, mas um pouco simplista, que nos ajuda a entender os protocolos. No Capítulo 30, por exemplo, aprendemos que os protocolos para uma route-over mesh graduam os limites entre camadas pela adição de shims e misturam encaminhamento de IP com alcançabilidade da Camada 2. Cada capítulo foi atualizado para focar as ideias e tecnologias que estão sendo utilizadas atualmente na Internet. A alteração mais significativa consiste na integração da discussão do IPv-6 com o IPv-4. Nos capítulos, descreve-se um princípio, explica-se o projeto geral, e então expõe-se como o princípio é aplicado aos IPv4 e IPv-6. Os leitores verão que essas duas versões de IP estão intimamente inter-relacionadas e que é impossível entender as mudanças introduzidas pelo IPv-6 sem entender os conceitos e princípios do IPv-4.

Como nas edições anteriores, que foram muito populares, todo o texto enfoca conceitos e princípios. Os capítulos iniciais descrevem a motivação para a interligação em redes e fornecem os fundamentos da tecnologia TCP/IP da internet. Veremos que interligação por rede é uma abstração poderosa que nos permite lidar com a complexidade de múltiplas tecnologias de comunicação subjacentes escondendo os detalhes de hardware de rede. Vamos entender os níveis de serviço de rede que a internet proporciona e ver como os aplicativos usam esse serviço. Capítulos posteriores trarão mais detalhes. No texto são comentados tanto a arquitetura das interconexões da rede como os princípios dos protocolos básicos que tornam tais redes interligadas função de um único sistema de comunicações integradas.

Após a leitura do livro, você irá entender como é possível interligar múltiplas redes físicas em um sistema coordenado, como os protocolos de internet funcionam nesse ambiente, e como programas de aplicação usam os sistema resultantes. Como exemplo específico, aprenderá os detalhes do TCP-IP global da Internet, inclusive a arquitetura de seu sistema de roteamento e o protocolo de aplicação que ele suporta. Entenderá também algumas das limitações de abordagem da internet e dos protocolos TCP/IP.

Elaborado tanto como texto escolar como de referência profissional, o livro está escrito em um nível para graduandos avançados ou graduados. Para profissionais, ele proporciona uma introdução abrangente tanto para a tecnologia TCP/IP como para a arquitetura da Internet. Embora não tenha o propósito de substituir documentos-padrão para protocolos, é um excelente

ponto de partida para o aprendizado sobre interligação por rede, pois fornece uma visão geral que enfatiza princípios. Além disso, dá a perspectiva ao leitor a qual pode ser extremamente difícil de obter a partir de documentos unicamente sobre protocolo.

Quando usado em sala de aula, o texto fornece material mais que suficiente para um semestre de curso tanto em nível de graduação como para graduados. Em um curso de graduação, encorajo os professores a incluir concepção e implementação de projetos significativos, bem como leituras de literatura que forneçam base para um maior aprofundamento. Muitos dos exercícios sugerem tais sutilezas; resolvê-los, muitas vezes, exige que os alunos leiam padrões de protocolo e apliquem energia criativa para compreender as consequências. Para os cursos de graduação, muitos dos detalhes são desnecessários. Deve-se esperar dos alunos a compreensão dos conceitos básicos descritos no texto, e eles devem ser capazes de descrever e de usar os protocolos fundamentais.

Em todos os níveis, a experiência prática aguça os conceitos e ajuda os alunos a ganharem intuição. Assim, encorajo os instrutores a propor projetos que deem aos alunos oportunidades para usar os serviços e protocolos da Internet. Em um curso para graduandos, a maioria dos projetos consistirá em escrever aplicativos que usam a rede. No meu curso de graduação, eu tenho alunos escrevendo um analisador de rede simplificada (ou seja, dado um pacote no sistema binário, imprimir o valor de cada campo). O projeto do semestre no meu curso de pós-graduação sobre interligação de rede na Universidade de Purdue exige que os alunos construam um software significativo para protocolo IP; o projeto tradicional envolve a implementação de um roteador IP. Nós fornecemos o hardware e o código fonte para um sistema operacional, incluindo o driver do dispositivo para as interfaces de rede; os alunos constroem um roteador de trabalho que interliga três redes com MTU diferentes. O curso é extremamente rigoroso, os estudantes trabalham em equipes e os resultados têm sido relevantes (muitas empresas recrutam formandos do curso). Embora tal experiência seja mais segura quando a rede de laboratórios de instrução está isolada das instalações de computação da produção, descobrimos que os alunos apresentam maior entusiasmo e obtêm mais benefícios quando eles têm acesso à Internet global e podem testar se seus protocolos interagem com versões comerciais.

O livro está organizado em cinco partes principais. Os Capítulos 1 e 2 formam uma introdução que fornece uma visão geral do assunto por meio da discussão das tecnologias existentes de rede. Em particular, o Capítulo 2 faz uma revisão sobre hardware de redes físicas. O objetivo é fornecer a

intuição sobre a funcionalidade que o hardware fornece e o que é possível fazer para não gastar muito tempo em detalhes de hardware. Os Capítulos de 3 a 11 descrevem o TCP/IP da Internet a partir do ponto de vista de um único host, mostrando os protocolos que um host contém e como eles operam. Cobrem as abstrações da internet, a noção de camada de protocolo, a base de endereçamento e encaminhamento da Internet e os protocolos de transporte. Os Capítulos de 12 a 14 consideram a arquitetura de uma internet quando vista de forma global. Exploram a arquitetura de roteamento e o uso de roteadores de protocolos para troca de informação de roteamento. Os Capítulos de 15 a 19 abordam variações e extensões de tecnologia básica, incluindo transmissão múltipla, classificação de pacotes, visualização de rede e mobilidade. Em particular, o capítulo sobre a mobilidade explica por que ela é difícil em uma rede IP. Finalmente, os Capítulos de 20 a 30 discutem serviços de aplicação disponíveis na Internet (incluindo gestão de rede), segurança de rede e a Internet das Coisas. Os Capítulos apresentam o modelo cliente-servidor de interação, dando vários exemplos de aplicações que usam tal modelo, e mostram como a interação cliente-servidor se aplica ao bootstrap dos computadores e à gestão da rede. O Capítulo 28 explica uma nova abordagem ao gerenciamento de rede conhecida como Rede Definida por Software (Software Defined Networking – SDN) e o protocolo principal, OpenFlow. Apesar de não ser parte oficial dos padrões TCP/IP, a tecnologia SDN foi incluída por ter gerado considerável excitação.

Os capítulos foram organizados de baixo para cima. Em vez de começar vendo a Internet como uma caixa-preta e aprender como usá-la, o texto começa com uma visão geral de hardware e continua adicionando os conceitos de protocolos necessários para criar a Internet.

Essa visão de baixo para cima vai atrair qualquer um que esteja interessado em engenharia, pois segue o padrão usado quando se constrói um sistema. Em algumas aulas, os professores preferem iniciar pelos Capítulos 20 e 21 de programação cliente-servidor, que permitem a seus alunos iniciarem mais cedo a escrita dos aplicativos de rede. Embora escrever aplicativos que *usem* a Internet seja importante, exorto os professores a incluírem também atividades que ajudem aos estudantes entenderem as tecnologias básicas (ou seja, protocolos e pacotes). Em uma atividade de laboratório, por exemplo, tive alunos que elaboraram um protocolo muito básico para contatar outro terminal e transferir dois pacotes: um contendo um nome de arquivo e o outro contendo dados do arquivo. Entre o emissor e o receptor, um aplicativo randomicamente para, duplica-se, atrasa e muda o conteúdo dos pacotes. O experimento é feito

com UDP, tornando trivial sua implementação. No entanto, os alunos tornam-se muito conscientes de como é difícil projetar protocolos.

É necessário um mínimo de conhecimento para entender o material. Os leitores não precisam de matemática avançada, nem saber teoria da informação ou teoremas de comunicação de dados; o livro descreve a rede física como uma caixapreta ao redor da qual uma interligação de rede pode ser construída. Espera-se que os leitores tenham um entendimento básico de sistemas de computadores e estejam familiarizados com estrutura de dados tais como pilhas, esperas e árvores. Ele deve ter uma noção básica sobre os serviços que um sistema operacional executa e de que processos podem ser executados de forma concorrente. Não se pressupõe conhecimento prévio da tecnologia da Internet: o texto não só relata de forma clara todos os princípios de projeto, mas também discute motivações e consequências.

Muita gente merece crédito por contribuir com sugestões e ideias para várias edições do texto ao longo dos anos. Para esta edição, um conjunto de revisores fez comentários sobre a organização e sobre itens que necessitavam de atualização, além de terem ajudado a verificar detalhes técnicos. Agradeço a Anthony Barnard, Tom Calabrese, Ralph Droms, Tom Edmunds, Raymond Kelso, Lee Kirk, John Lin, Dave Roberts, Gustavo Rodriguez-Rivera e Bhaskar Sharma, que revisaram, todos, o rascunho do manuscrito. John e Ralph foram especialmente úteis. Barry Shein contribuiu com o exemplo de código cliente-servidor no Capítulo 21.

Como de costume, minha esposa Christine prestou a maior ajuda. Ela gastou horas com o manuscrito, identificando ambiguidades, encontrando inconsistências e suavizando o texto.

Douglas E. Comer
Março, 2013.

O QUE OUTROS DISSERAM A RESPEITO DA SEXTA EDIÇÃO DE INTERLIGAÇÃO DE REDES COM TCP/IP

"Este é o livro a que recorro para obter explicações claras dos princípios básicos e dos desenvolvimentos mais recentes em tecnologias TCP/IP. É uma referência 'essencial' para os profissionais de rede."

Dr. Ralph Droms,

Cisco Systems.

Coordenador do grupo de trabalho DHCP.

"Excelente livro! Obrigado!"

Henrik Sundin, *NTI Gymnasiet*,

Estocolmo, Suécia.

"A 6ª edição do clássico *Interligação de redes*, de Comer, documenta a evolução contínua e acelerada da Internet, enquanto prediz o futuro com clareza e compreensão incomparáveis."

Dr. Paul V. Mockapetris,

Inventor do Domain Name System.

"… uma verdadeira obra-prima."

Mr. Javier Sandino,

engenheiro de sistemas.

"O livro sobre TCP/IP mais bem escrito que eu já li. Dr. Comer expõe ideias complexas de forma clara, com excelentes gráficos e explicações. Com esta edição, o Dr. Comer torna contemporâneo este livro clássico."

Dr. John Lin,

Bell Laboratories.

"Esta atualização para a referência definitiva sobre as principais tecnologias da Internet confirma a reputação de Doug Comer pela apresentação clara e precisa de informações essenciais; deve ser a pedra angular da biblioteca de qualquer profissional de Internet."

Dr. Lyman Chapin,

Interisle Consulting Group,

Ex-presidente IAB.

"Um dos melhores livros que já li. O verdadeiro gênio não é apenas fluente em seu campo, como também sabe expressar suas ideias de forma simples. Obrigado, Dr. Comer, por escrever este excelente livro!"

Marvin E. Miller,

CIO, The ACS Corporation.

"Em um mundo de complexidade, a capacidade de transmitir conhecimento em oposição à informação a partir de um site de busca é bastante difícil. Poucos fazem parecer tão fácil como Doug Comer, cujo livro *Interligação de Redes*, volume 1, continua a desempenhar uma função fundamental em nos ensinar sobre o papel dos protocolos numa Internet continuamente em mudança."

Dr. Balachander Krishnamurthy,

AT&T Labs.

"Está acontecendo uma rápida evolução da Internet, já que o mundo inteiro a usa a cada dia, a cada hora, ou até mesmo de forma contínua (como é o caso dos meus netos). Comer monitora com precisão os fundamentos tecnológicos relevantes para a construção da Internet atualmente."

Dan Lynch,

Fundador, INTEROP.

Sumário

CAPÍTULO 1

Introdução e visão geral

CONTEÚDOS DO CAPÍTULO

1.1 A motivação para interligação de redes

A comunicação pela Internet tornou-se parte fundamental da vida. Redes sociais, tais como Facebook, promovem conexões entre um grupo de amigos e permite que eles compartilhem interesses. A World Wide Web contém informações sobre temas diversos como política, condições atmosféricas, preços das ações, colheitas e tarifas aéreas. Familiares e amigos usam a Internet para compartilhar fotos e manter contato com chamadas telefônicas VoIP e chats de vídeo ao vivo. Os consumidores usam a Internet para adquirir bens e serviços e para acesso a bancos. Empresas fazem encomendas e pagamentos eletronicamente. A mudança para a computação em nuvem vai colocar mais informações e serviços *on-line*.

Embora pareça operar como uma rede unificada, a Internet não é composta de uma única tecnologia de rede, pois nenhuma tecnologia isolada é suficiente para todos os usos. Em vez disso, o hardware de rede é projetado para situações e orçamentos específicos. Alguns grupos precisam de redes de alta velocidade para conectar computadores em um único prédio. Outros precisam de redes sem fio de baixo custo. Como o hardware de baixo custo que funciona bem dentro de um prédio não pode se espalhar a grandes distâncias geográficas, é preciso usar uma alternativa para conectar máquinas milhares de quilômetros distantes entre si.

Na década de 1970, foi criada uma tecnologia que tornou possível conectar várias redes individuais dispersas e operá-las como uma unidade

coordenada. Conhecida como *interligação de redes*, a tecnologia estabeleceu as bases para a Internet, acomodando várias tecnologias de hardware básicas, oferecendo um modo de interconectar as redes e definindo um conjunto de convenções de comunicação que as redes utilizam para interoperar. A tecnologia de interconexão de redes oculta os detalhes do hardware de rede e permite que os computadores se comuniquem independentemente de suas conexões de rede física.

A tecnologia da Internet é um exemplo de *interconexão de sistema aberto*. Ela é chamada *aberta* porque, diferentemente dos sistemas disponíveis de um fornecedor em particular, as especificações estão disponíveis publicamente. Assim, qualquer companhia individual pode montar o software necessário para se comunicar por uma interconexão de redes. Mais importante, a tecnologia inteira foi projetada para promover a comunicação entre máquinas com arquiteturas de hardware diversificadas, usar quase todo tipo de hardware de rede comutada por pacotes e acomodar não só uma grande variedade de aplicações, mas também diversos sistemas operacionais de computador.

1.2 A Internet TCP/IP

Nas décadas de 1970 e 1980, agências do governo dos Estados Unidos observaram a importância e o potencial da tecnologia da Internet e financiaram pesquisas que possibilitaram uma Internet global.* Este livro discute princípios e ideias que resultaram de pesquisa financiada pela *Defense Advanced Research Projects Agency* (DARPA).** A tecnologia DARPA inclui um conjunto de padrões de rede que especifica os detalhes de como os computadores se comunicam, além de um conjunto de convenções para interconexão de redes e encaminhamento de tráfego. Oficialmente chamada *TCP/IP Internet Protocol Suite* e normalmente referenciada como *TCP/IP* (os nomes de seus dois padrões principais), ela pode ser usada para comunicação entre qualquer conjunto de redes interconectadas. Por exemplo, TCP/IP pode ser usado para interconectar um conjunto de redes dentro de um único prédio, dentro de um *campus* físico ou entre um conjunto de *campi*.

Embora a tecnologia TCP/IP seja isoladamente digna de nota, ela é especialmente interessante porque sua viabilidade foi demonstrada em grande escala. Ela forma a tecnologia básica da Internet global, que conecta

aproximadamente 2 bilhões de indivíduos em lares, escolas, corporações e laboratórios do governo em praticamente todas as áreas povoadas do planeta. Um sucesso fantástico, a Internet demonstra a viabilidade da tecnologia TCP/IP e mostra como ela pode acomodar uma grande variedade de tecnologias básicas de hardware.

1.3 Serviços de Internet

Não se pode apreciar os detalhes técnicos por trás do TCP/IP sem entender os serviços que ele oferece. Esta seção analisa rapidamente os serviços de internet, destacando aqueles que a maioria dos usuários acessa, e deixa para os capítulos posteriores a discussão de como os computadores se conectam a uma rede TCP/IP e como a funcionalidade é implementada.

Grande parte de nossa discussão sobre os serviços terá como foco os padrões chamados *protocolos*. Protocolos como o TCP e o IP fornecem as regras sintáticas e semânticas para comunicação. Eles fornecem os detalhes dos formatos de mensagem, descrevem como um computador responde quando chega uma mensagem e especificam como um computador trata de erros ou outras condições anormais. Mais importante, os protocolos nos permitem discutir comunicação de computador independente do hardware de rede de qualquer fornecedor em particular. De certa forma, os protocolos são para a comunicação o que os algoritmos representam para a computação. Um algoritmo permite que alguém especifique ou entenda uma computação sem conhecer os detalhes de uma linguagem de programação particular ou do conjunto de instruções de uma CPU. De modo semelhante, um protocolo de comunicação permite que alguém especifique ou entenda a comunicação de dados sem depender do conhecimento detalhado do hardware de rede de determinado fornecedor.

Ocultar os detalhes de baixo nível da comunicação ajuda a melhorar a produtividade de várias maneiras. Primeiro, como os programadores tratam de abstrações de protocolo de alto nível, eles não precisam aprender ou lembrar-se de tantos detalhes sobre determinada configuração de rede. Assim, eles podem criar novos programas rapidamente. Segundo, como os programas confeccionados utilizando abstrações de alto nível não são restritos a determinada arquitetura de computador ou determinado hardware de rede, não precisam ser modificados quando as redes ou os computadores são substituídos ou reconfigurados. Terceiro, como os programas de aplicativos confeccionados utilizando protocolos de alto nível são independentes do hardware básico, eles podem oferecer comunicação direta entre um par de computadores qualquer. Os

programadores não precisam criar uma versão especial do aplicativo para cada tipo de computador ou cada tipo de rede. Em vez disso, o software criado para usar protocolos é de uso geral; o mesmo código pode ser compilado e executado em qualquer computador.

Veremos que os detalhes de cada serviço disponível na Internet são dados por um protocolo separado. As próximas seções referem-se a protocolos que especificam alguns dos serviços em nível de aplicação, além daqueles usados para definir serviços em nível de rede. Capítulos posteriores explicam cada um desses protocolos em detalhes.

1.3.1 Serviços de Internet em nível de aplicação

Do ponto de vista de um usuário, a Internet parece consistir em um conjunto de programas que utiliza a rede para realizar tarefas úteis de comunicação. Usamos o termo *interoperabilidade* para nos referir à capacidade de os sistemas de computação diversificados cooperarem na solução de problemas computacionais. Devido à Internet ter sido projetada para se ajustar a redes e computadores heterogêneos, a interoperabilidade foi um requisito básico. Em consequência, programas aplicativos de redes normalmente exibem em alto grau tal característica. Na verdade, a maioria dos usuários acessa os programas aplicativos sem entender os tipos de computadores e redes que estão sendo acessados, os protocolos de comunicação, nem o caminho em que os dados trafegam até seu destino. Assim, um usuário pode acessar uma página na web a partir de um sistema desktop ligado por cabo a um modem ou de um iPad conectado a uma rede sem fio 4G.

Os serviços da Internet mais populares e difundidos são descritos a seguir.

- *World Wide Web.* A Web tornou-se a maior fonte de tráfego na Internet global entre 1994 e 1995, e permanece assim. Muitos serviços populares, incluindo busca na Internet (por exemplo, Google) e redes sociais (por exemplo, Facebook), usam a tecnologia web. Estimativas atribuem cerca de um quarto de toda a Internet ao tráfego no Facebook. Embora os usuários façam distinção entre os vários serviços baseados na web, veremos que todos eles usam o mesmo protocolo em nível de aplicação.
- *Acesso em nuvem (cloud access) e computador remoto (remote desktop).* Trata-se de locais de computação em nuvem, instalações de

armazenagem em *centros de dados em nuvem* e arranjos para que os usuários acessem os serviços por meio da Internet. Uma tecnologia de acesso, conhecida como *serviço de desktop remoto*, permite que um usuário acesse um computador em um centro de dados remoto como se o computador fosse local. O usuário só precisa de um dispositivo de interface com a tela, teclado, mouse ou touchpad, e uma conexão de rede. Quando o computador do centro de dados atualiza a exibição do vídeo, o serviço de desktop remoto captura as informações, envia através da Internet e exibe na tela do usuário. Quando o usuário move o mouse ou pressiona uma tecla, o serviço de desktop remoto envia as informações para o centro de dados. Assim, o usuário tem acesso completo a um PC poderoso, mas só precisa levar um dispositivo básico de interface, como um tablet.

- *Transferência de arquivos.* O protocolo de transferência de arquivos permite aos usuários enviar e receber uma cópia de um arquivo de dados. Muitos downloads de arquivos, incluindo downloads de filmes, requerem um mecanismo de transferência de arquivos. Devido ao fato de sempre solicitarem a transferência de arquivo por uma página da web, os usuários podem não estar cientes de que um pedido de transferência de arquivos foi executado.
- *Correio eletrônico (e-mail).* O correio eletrônico, que já foi responsável por grandes quantidades de tráfego na Internet, foi substituído, em grande parte, por aplicações da web. Muitos usuários agora acessam e-mail através de uma aplicação web que lhes permite ler as mensagens em sua caixa postal, selecionar uma mensagem para processamento e encaminhar a mensagem ou enviar uma resposta. Uma vez que um usuário especifica um envio de mensagem, o sistema subjacente usa um protocolo de transferência de e-mail para enviar a mensagem para a caixa de correio do destinatário.
- *Serviços de voz e vídeo.* Tanto o streaming de vídeo como o de áudio já respondem pelo transporte de uma fração significativa de bits através da Internet em nível mundial, e essa tendência vai continuar. Mais importante, uma mudança significativa está ocorrendo; o upload de vídeo está aumentando, especialmente porque os usuários estão usando dispositivos móveis para enviar vídeo de eventos ao vivo.

Retornaremos a essas e outras aplicações em capítulos posteriores, quando as examinaremos com mais detalhes. Veremos exatamente como as

aplicações utilizam os protocolos TCP/IP básicos e por que ter padrões para protocolos de aplicação tem ajudado a garantir que eles sejam difundidos.

1.3.2 Serviços de Internet em nível de rede

Um programador que cria programas aplicativos para rede possui uma visão da Internet totalmente diferente daquela que possui um usuário que simplesmente executa aplicações, como um site de busca na web. Em nível de rede, a Internet oferece dois grandes tipos de serviço que todos os programas aplicativos utilizam. Embora não seja importante no momento entender os detalhes desses serviços, eles são fundamentais para uma visão geral do TCP/IP.

- *Connectionless Packet Delivery Service*. O envio por pacotes, explicado com detalhes ao longo do texto, forma a base para todos os serviços da internet. A entrega sem conexão é uma abstração do serviço que a maioria das redes de comutação de pacotes oferece. Significa simplesmente que uma rede TCP/IP encaminha pequenas mensagens de um computador para outro com base na informação de endereço transportada na mensagem. Como o serviço sem conexão encaminha cada pacote separadamente, uma internet não garante, de forma confiável, a entrega em ordem. Entretanto, por mapear diretamente na maioria das tecnologias de hardware, o serviço sem conexão é extremamente eficiente. Mais importante, devido ao projeto promover a entrega de pacote sem conexão como base para todos os serviços de internet, os protocolos TCP/IP podem se adaptar a uma grande gama de hardwares de rede.
- *Serviço de transporte com fluxo confiável (reliable stream transport service)*. A maioria das aplicações precisa que o software de comunicação recupere automaticamente erros de transmissão, pacotes perdidos ou falhas dos comutadores (switches) intermediários ao longo do caminho entre emissor e receptor. Consequentemente, elas precisam de um serviço de transporte confiável para resolver esses problemas. O serviço de transporte confiável de fluxo permite que uma aplicação em um computador estabeleça uma "conexão" com uma aplicação em outro computador e permite que a aplicação transfira arbitrariamente grandes volumes de dados pela conexão, como se fosse uma conexão direta de hardware permanente. Por trás disso, os protocolos de comunicação dividem o fluxo de dados em pequenos

pacotes e os enviam, um de cada vez, esperando que o receptor confirme a recepção.

Muitas redes fornecem serviços básicos semelhantes àqueles esboçados acima, de modo que alguém poderia questionar o que distingue os serviços TCP/IP dos outros. As principais características que os distinguem são apresentadas a seguir:

- *Independência da tecnologia de rede.* Embora o TCP/IP seja baseado na tecnologia de comutação de pacotes, ele é independente de qualquer marca ou tipo de hardware em particular; a Internet global inclui uma série de tecnologias de rede. Os protocolos TCP/IP definem a unidade de transmissão de dados, denominada *datagrama*, e especificam como transmiti-los em uma rede em particular, mas nada em um datagrama está vinculado a um hardware específico.
- *Interconexão universal.* A Internet permite a comunicação de qualquer par de computadores. Cada computador recebe um *endereço* que é reconhecido universalmente pela Internet. Cada datagrama transporta os endereços de sua origem e destino. Os dispositivos de comutação intermediários utilizam o endereço de destino para tomar decisões de encaminhamento; um emissor só precisa saber o endereço do receptor e a Internet se encarrega de encaminhar os datagramas.
- *Confirmações fim a fim.* Os protocolos TCP/IP da Internet fornecem confirmações entre a origem e o destino final, e não entre máquinas sucessivas ao longo do caminho, mesmo que a origem e o destino não se conectem a uma rede física comum.
- *Padrões de protocolo de aplicação.* Além dos serviços básicos em nível de transporte (como as conexões de fluxo confiável), os protocolos TCP/IP incluem padrões para muitas aplicações comuns, incluindo protocolos que especificam como acessar uma página na web, transferir arquivos, e enviar e-mail. Assim, ao projetar programas aplicativos que utilizam o TCP/IP, os programadores normalmente procuram os softwares existentes que tenham os protocolos de aplicação que permitam os serviços de comunicação de que precisam.

Capítulos posteriores discutirão os detalhes dos serviços fornecidos ao programador, assim como exemplos de padrões de aplicação de protocolo.

1.4 História e escopo da Internet

Parte do que torna a tecnologia TCP/IP tão interessante é sua adoção universal e o tamanho e a taxa de crescimento da Internet global. A DARPA começou a trabalhar em direção a uma tecnologia de interconexão de redes em meados da década de 1970, com a arquitetura e os protocolos tomando sua forma atual por volta de I977-1979. Nessa época, a DARPA era conhecida como a principal agência financiadora de pesquisa de redes de comutação de pacotes, pioneira de muitas ideias na comutação de pacotes, com sua conhecida *ARPANET*. A ARPANET usava interconexões de linha alugada ponto a ponto convencional, mas a DARPA também financiou a exploração da comutação de pacotes por redes de rádio e canais de comunicação por satélite. Na realidade, a diversidade crescente das tecnologias de hardware de rede ajudou a forçar a DARPA a estudar a interconexão de rede e levou adiante a interligação de redes.

A disponibilidade do financiamento de pesquisa da DARPA chamou a atenção e a imaginação de vários grupos de pesquisa, especialmente aqueles pesquisadores que já haviam adquirido experiência anterior no uso da comutação de pacotes na ARPANET. A DARPA marcou reuniões informais de pesquisadores para compartilhar ideias e discutir resultados de experiências. Informalmente, o grupo era conhecido como *Internet Research Group*. Em 1979, tantos pesquisadores estavam envolvidos no esforço do TCP/IP que a DARPA criou um comitê informal para coordenar e guiar o projeto dos protocolos e a arquitetura da Internet emergente. Denominado *Internet Control and Configuration Board* (ICCB), o grupo se reunia regularmente até 1983, quando foi reorganizado.

A Internet global começou por volta de 1980, quando a DARPA começou a converter máquinas conectadas às suas redes de pesquisa aos novos protocolos TCP/IP. Já em funcionamento, a ARPANET rapidamente se tornou o backbone da nova Internet e foi usada para muitas das primeiras experiências com TCP/IP. A transição para a tecnologia de Internet foi completada em janeiro de 1983, quando a Secretaria de Defesa determinou que todos os computadores conectados a redes de longa distância usassem TCP/IP. Ao mesmo tempo, a Defense Communication Agency (DCA) dividiu a ARPANET em duas redes separadas, uma para continuar a pesquisa e uma para comunicação militar. A parte de pesquisa reteve o nome ARPANET; a parte militar, que era um pouco maior, tornou-se conhecida como a *rede militar* (MILNET).

Para encorajar pesquisadores universitários a adotarem e usarem os

novos protocolos, a DARPA tornou a implementação disponível a baixo custo. Nesse tempo, a maior parte dos departamentos de ciência de computação estavam executando uma versão do sistema operacional UNIX disponível no *BerkeleySoftware Distribution* da Universidade da Califórnia, normalmente chamado BerkeleyUNIX ou *BSD UNIX*. Ao financiar a Bolt Beranek e Newman, Incorporated (*BBN*) para implementar seus protocolos TCP/IP para uso com UNIX e ao financiar a Berkeley para integrar os protocolos com sua distribuição de software, a DARPA foi capaz de alcançar mais de 90% dos departamentos de ciência de computação das universidades. O novo protocolo chegou em uma época particularmente significativa, pois muitos departamentos estavam acabando de adquirir segundo ou terceiro computadores e conectando-os a redes locais. Os departamentos precisavam de protocolos de comunicação que fornecessem serviços como transferência de arquivos.

Além de um conjunto de programas utilitários, o Berkeley UNIX criou uma nova abstração no sistema operacional conhecida como *socket*, para permitir que programas aplicativos acessassem protocolos de Internet. Uma generalização do mecanismo UNIX para I/O, a interface socket tem opções para vários tipos de protocolos de rede, além do TCP/IP. A introdução da abstração de socket foi importante porque permitiu que os programadores usassem protocolos TCP/IP com pouco esforço. A interface socket tornou-se o padrão de fato, sendo usada na maioria dos sistemas operacionais.

Percebendo que a comunicação de redes logo seria uma parte crucial da pesquisa científica, a National Science Foundation (NSF) assumiu um papel ativo na expansão da Internet TCP/IP no sentido de alcançar o máximo possível de cientistas. No final da década de I970, a NSF patrocinou um projeto conhecido como *Computer Science NETwork* (*CSNET*), cujo objetivo foi conectar todos os cientistas de computação. A partir de I985, a NSF iniciou um programa para estabelecer redes de acesso em torno de seus seis centros de supercomputação, e em 1986 expandiu os esforços de rede financiando uma nova rede de backbone, conhecida como *backbone NSFNET*. A NSF também forneceu muito dinheiro para redes regionais, cada uma conectando as principais instituições de pesquisa científica em determinada área.

Por volta de 1984, a Internet chegou a mais de 1.000 computadores. Em 1987, o tamanho cresceu para mais de 10.000. Em 1990, o tamanho superou 100.000, e, em 1993, ultrapassou 1 milhão. Em 1997, mais de 10 milhões de

computadores foram permanentemente ligados à Internet e, em 2001, o tamanho excedeu 100 milhões. Em 2011, a Internet atingiu mais de 800 milhões de computadores permanentemente conectados.

O crescimento inicial da Internet não ocorreu simplesmente porque as universidades e grupos financiados pelos governos adotaram os protocolos. Grandes corporações de computadores se conectaram à Internet, assim como muitas outras grandes empresas, incluindo companhias de petróleo, indústria automobilística, empresas de eletrônicos, empresas farmacêuticas e operadoras de telecomunicações. Pequenas e médias empresas começaram a se conectar na década de 1990. Além disso, muitas empresas fizeram experiências usando protocolos TCP/IP em suas intranets corporativas internas antes de escolherem para fazer parte da Internet global.

1.5 O Internet Architecture Board (IAB)

Como o conjunto de protocolos TCP/IP não surgiu de um fornecedor específico ou de uma sociedade profissional reconhecida, é natural perguntar: "Quem definiu a direção técnica e decidiu quando os protocolos se tornariam padronizados?". A resposta é um grupo conhecido como *Internet Architecture Board* (*IAB*),* formado em 1983, quando a DARPA reorganizou o Internet Control and Configuration Board. O IAB estabeleceu o foco e a coordenação para grande parte da pesquisa e desenvolvimento por trás dos protocolos TCP/IP, e conduziu a evolução da Internet. Ele decidiu quais protocolos seriam parte obrigatória do conjunto TCP/IP e definiu políticas oficiais.

1.6 A reorganização do IAB

No inverno de 1989, tanto a tecnologia TCP/IP quanto a Internet cresceram muito além do projeto de pesquisa inicial, entrando em instalações de produção das quais milhares de pessoas dependiam para os negócios do dia a dia. Não era mais possível introduzir novas ideias alterando algumas instalações da noite para o dia. Em grande parte, as centenas de empresas comerciais que oferecem produtos TCP/IP determinaram se os produtos interoperariam decidindo quando incorporar mudanças em seu software. Os pesquisadores que elaboraram especificações e testaram novas ideias em laboratórios não poderiam mais esperar aceitação e uso instantâneo das ideias. Foi irônico que os pesquisadores que projetaram e observaram o TCP/IP se desenvolver se encontraram sufocados pelo sucesso comercial de sua criação. Resumindo, os protocolos TCP/IP e a Internet tornaram-se uma tecnologia de produção bem-sucedida, e o mercado começou a dominar sua

evolução.

Para refletir as realidades políticas e comerciais do TCP/IP e da Internet, o IAB foi reorganizado no inverno de 1989. Os pesquisadores foram movidos do próprio IAB para um grupo subsidiário conhecido como *Internet Research Task Force* (*IRTF*), e um novo comitê do IAB foi constituído para incluir representantes da comunidade mais ampla. A responsabilidade pelos padrões de protocolos e outros aspectos técnicos passaram para um grupo conhecido como *Internet Engineering Task Force* (*IETF*).

O IETF existia na estrutura original do IAB, e seu sucesso forneceu parte da motivação para a reorganização. Diferentemente da maioria das forças-tarefa do IAB, que eram limitadas a alguns indivíduos que focavam uma questão específica, o IETF era grande antes da reorganização, ele tinha crescido para incluir dezenas de membros ativos que trabalhavam em muitos problemas simultaneamente. Após a reorganização, o IETF foi dividido em mais de 20 *grupos de trabalho*, cada qual focando um problema específico.

Como o IETF era muito grande para um único presidente dirigir, ele foi dividido em aproximadamente dez áreas, cada uma com seu próprio gerente. O presidente do IETF e os gerentes de área constituem o *Internet Engineering Steering* Group (*IESG*), os indivíduos responsáveis por coordenar os esforços dos grupos de trabalho do IETF. O nome "IETF" agora se refere à agência inteira, incluindo o presidente, os gerentes de área e todos os membros dos grupos de trabalho.

1.7 Internet Request for Comments (RFCs)

Dissemos que a tecnologia TCP/IP não pertence a nenhum fornecedor nem a qualquer sociedade profissional ou agência de padrões. Portanto, a documentação dos protocolos, padrões e políticas não pode ser obtida de um fornecedor. Em vez disso, a IETF gerencia o processo de padronização. A documentação sobre protocolos resultante é colocada em repositórios *on-line* e encontra-se disponível sem qualquer cobrança.

A documentação de trabalho na Internet, propostas para protocolos novos ou revisados e padrões de protocolo TCP/IP aparecem em uma série de relatórios técnicos chamados *Internet Requests For Comments*, ou

RFCs, que podem ser curtas ou longas, podem abranger conceitos amplos ou detalhes e podem ser padrões ou simplesmente propostas para novos protocolos. Existem referências a RFCs no decorrer do texto. Embora as RFCs não sejam referidas da mesma maneira que os artigos de pesquisa acadêmicos, elas são revistas e editadas. Durante muitos anos, um único indivíduo, Jon Postel, trabalhou como editor de RFC. A tarefa de editar RFCs agora está com gerentes de área do IETF; o IESG como um todo aprova novas RFCs.

A série RFC é numerada sequencialmente na ordem cronológica em que as RFCs são escritas. Cada RFC nova ou revisada recebe um novo número, de modo que os leitores devem ter cuidado para obter a versão de número mais alto de um documento; um índice de RFCs está disponível para ajudar a identificar a versão correta. Além disso, versões preliminares dos documentos de RFC, conhecidas como *Internet drafts*, também estão disponíveis.

As RFCs e os Internet Drafts podem ser obtidos em: *http://www.ietf.org*.

1.8 O crescimento da Internet

A Internet tem crescido rapidamente e continua a evoluir. Novos protocolos estão sendo propostos; os antigos estão sendo revistos. A demanda mais significativa sobre a tecnologia subjacente não surge de conexões de rede adicionais, mas de tráfego adicional. À medida que novos usuários se conectam à Internet e novas aplicações aparecem, os padrões de tráfego mudam. Por exemplo, quando a *World Wide Web* foi introduzida, tornou-se extremamente popular, e o tráfego da Internet aumentou dramaticamente. Mais tarde, quando o compartilhamento de música tornou-se popular, os padrões de tráfego mudaram novamente. Mais mudanças estão ocorrendo, já que a Internet é usada para telefone, vídeo e redes sociais.

A Figura 1.1 resume a expansão da Internet e ilustra um componente importante do crescimento: grande parte da mudança na complexidade surgiu porque vários grupos agora gerenciam várias partes do todo.

	Número de redes	Número de computadores	Número de usuários	Número de gerentes
1980	10	10^2	10^2	10^0
1990	10^3	10^5	10^6	10^1

2000	10^5	10^7	10^8	10^2
2010	10^6	10^8	10^9	10^3

Figura 1.1 Crescimento da Internet. Além de aumentos no tráfego, a complexidade surgiu devido ao gerenciamento descentralizado.

O número de computadores ligados à Internet ajuda a ilustrar o crescimento. A Figura 1.2 contém um gráfico.

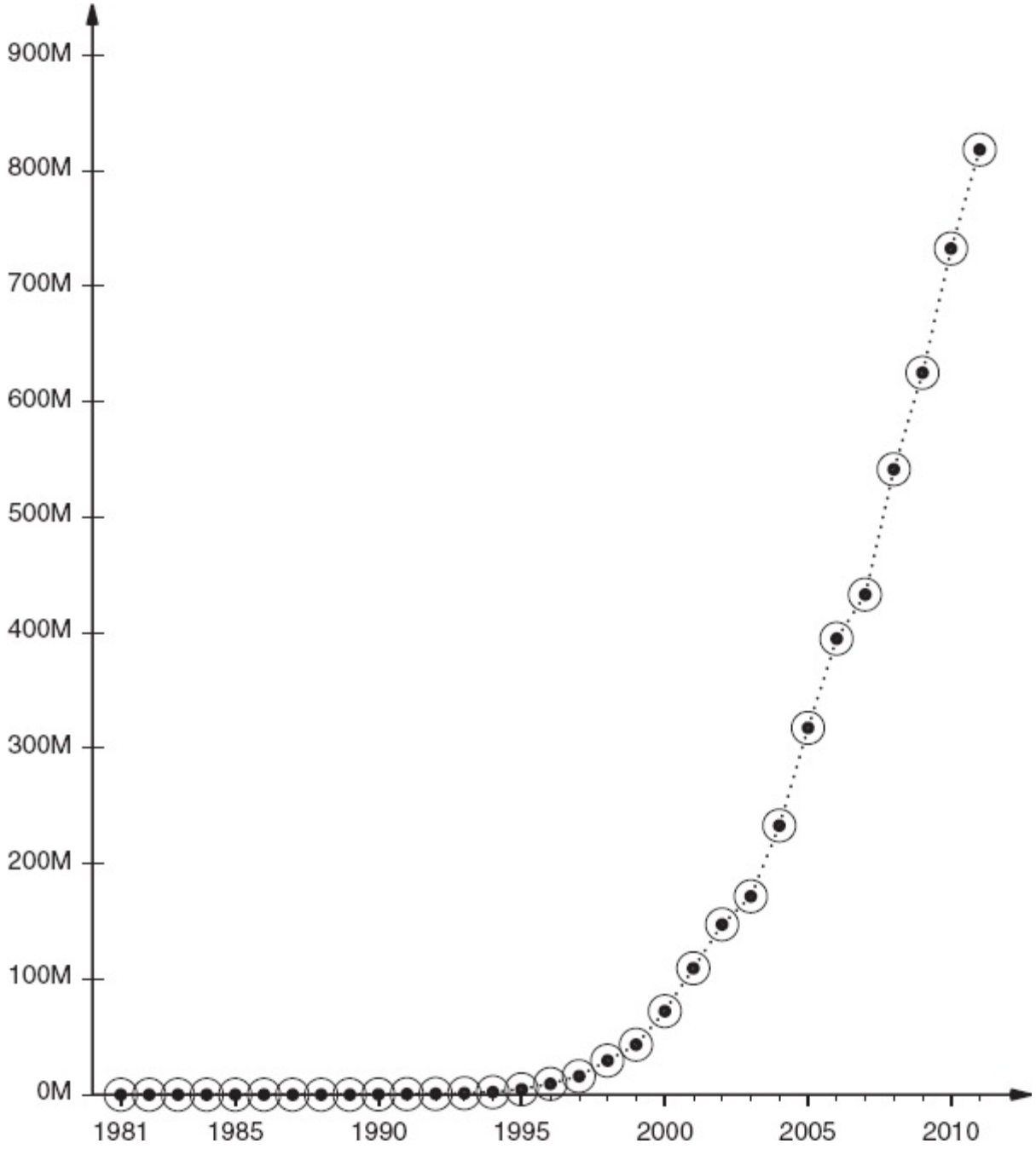


Figura 1.2 Computadores na Internet em função dos anos (escala linear).

O gráfico faz parecer que a Internet não começou a crescer até o final de 1990. No entanto, a escala linear esconde um ponto importante: mesmo na Internet precoce, a taxa de crescimento foi elevada. A Figura 1.3 apresenta os mesmos dados representados graficamente numa escala logarítmica. A figura revela que, embora a contagem de computadores tenha sido muito menor, alguns dos mais rápidos crescimentos ocorreram no final de 1980, quando a Internet cresceu de 1.000 para mais de 10.000 computadores.

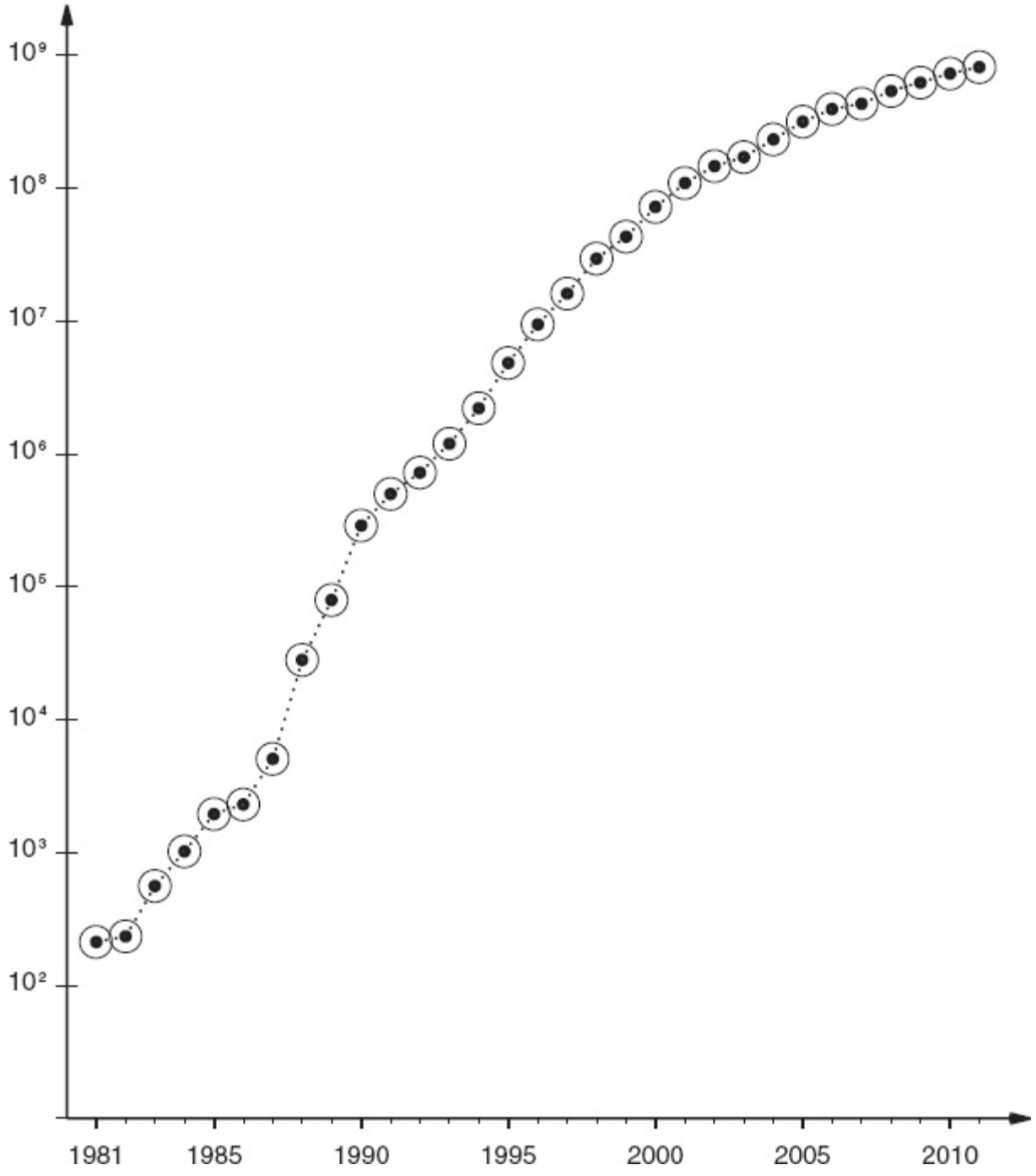


Figura 1.3 Computadores na Internet em função dos anos (escala logarítmica).

O número de computadores não é a única mudança significativa. Porque a tecnologia foi desenvolvida quando uma única pessoa no DARPA tinha o controle de todos os aspectos da Internet, os projetos de muitos subsistemas dependiam do gerenciamento e controle centralizados.

Como a Internet cresceu, a responsabilidade e o controle foram

divididos entre múltiplas organizações. Em particular, como a Internet tornou-se global, as operações e o gerenciamento precisaram abranger múltiplos países. A maior parte do esforço desde o início dos anos 1990 foi direcionada no sentido de encontrar caminhos para expandir o projeto de forma a acomodar um gerenciamento descentralizado.

1.9 Transição para o IPv6

A evolução da tecnologia TCP/IP sempre esteve interligada com a evolução da Internet global. Com bilhões de usuários em sites ao redor do mundo dependendo da Internet global como parte de suas rotinas diárias, pode parecer que tenhamos passado dos estágios iniciais de desenvolvimento e agora tenhamos alcançado instalações de produção estáveis. No entanto, apesar das aparências, nem a Internet nem o conjunto de protocolos TCP/IP são estáticos. Inovações contínuas tais como novos aplicativos são desenvolvidas e novas tecnologias são usadas para melhorar os mecanismos subjacentes.

Um dos esforços mais significativos envolve a revisão do Protocolo Internet, o fundamento de todas as comunicações na Internet. A mudança pode parecer surpreendente, em vista do sucesso da atual versão de IP.

Por que mudar? A versão corrente do Protocolo Internet, IPv4, tem sido memorável. Ela foi a primeira versão de trabalho e permaneceu praticamente sem modificações desde sua concepção, no final dos anos 1970. Sua longevidade mostra que o IPv4 é flexível e poderoso. Desde a época em que foi desenvolvido, o desempenho dos processadores aumentou da ordem de 2000 e a largura das bandas dos links mais velozes na Internet aumentou na ordem de 1.000.000. Surgiu a tecnologia sem fio, e o número de hosts na Internet aumentou de um punhado para centenas de milhões.

Apesar do sucesso do IPv4, críticos começaram a discutir, no início dos anos 1990, que ele seria insuficiente para novas aplicações, tais como voz e vídeo, e que o crescimento da Internet iria rapidamente esgotar o conjunto de endereços disponível. Desde então, duas coisas se tornaram aparentes: aplicações como telefonia digital trabalharam bem com o IPv4, e revisões no mecanismo de endereçamento da Internet produziram endereços suficientes para no mínimo mais uma década. Entretanto, se designarmos um endereço IP para cada dispositivo (por exemplo, cada aparelho inteligente, cada automóvel, cada telefone móvel), o espaço de endereço realmente se esgotará.

1.10 Projeto comitê e a nova versão do IP

Foram necessários muitos anos para a IETF formular uma nova versão de IP. Como a IETF produz padrões *abertos*, representantes de muitas comunidades foram convidados a participar do processo. Fabricantes de computadores, fornecedores de hardware e software, usuários, programadores, companhias de telefonia e a indústria de televisão a cabo, todos especificaram suas solicitações para a próxima versão de IP, e todos fizeram seus comentários a respeito das propostas específicas.

Muitos projetos foram propostos para atender a propósitos ou comunidades em particular. No final, cada grupo produziu um projeto expandido que incluiu as ideias de várias propostas iniciais. A IETF homologou a revisão do IP como versão número *6* e a nomeou *IPv6.**

1.11 Relação entre IPv4 e IPv6

Embora os proponentes quisessem criar uma Internet totalmente nova, o IPv6 herdou muitos dos princípios e características de projeto do IPv4. Em consequência, o IPv6 não pode ser compreendido de forma isolada – precisamos revisar os princípios gerais, entender como eles foram implementados no IPv4 e, então, ver como foram modificados ou expandidos no IPv6. Por exemplo, o IPv6 usa um projeto de endereços hierárquico que foi herdado diretamente do endereçamento classless do IPv4; o uso de máscaras de endereço e até alguma de sua terminologia foram derivados também dele. Na verdade, o IPv6 inclui todos os endereços existentes do IPv4 como um subconjunto do novo conjunto de endereços. Portanto, ao longo do texto, vamos discutir princípios e conceitos, estudar sua implementação no IPv4 e, então, ver as extensões e modificações no IPv6.

Como o IPv6 se difere? Os padrões dispõem que o IPv6 mantém muitas das características que contribuíram para o sucesso do IPv4. Na realidade, os projetistas caracterizam o IPv6 basicamente como o mesmo que o IPv4 com pequenas modificações. Por exemplo, ambos usam o paradigma connectionless delivery, permitindo ao emissor escolher o tamanho dos dados a serem enviados, e também exigem que ele especifique o número máximo de hops que um datagrama pode fazer antes de ser encerrado. O IPv6 mantém muitas outras facilidades do IPv4, tal como fragmentação. O ponto importante é descrito a seguir:

> *Devido ao IPv6 ter herdado muitos dos conceitos, princípios, e mecanismos encontrados no IPv4, não podemos entender o IPv6 sem ter*

entendido o IPv4; ambos são apresentados ao longo do texto.

Apesar das similaridades conceituais, o IPv6 muda a maior parte dos detalhes de protocolo. Ele usa maiores endereços e revisa completamente o formato dos pacotes. As mudanças introduzidas pelo IPv6 podem ser agrupadas em sete categorias descritas a seguir.

- *Endereços maiores.* O novo tamanho de endereço é a mudança mais evidente. O IPv6 quadruplica o tamanho de endereço do IPv4, de 32 bits para 128 bits.
- *Hierarquia de endereço estendida.* O IPv6 usa o maior espaço de endereço para criar níveis adicionais de hierarquia de endereçamento (por exemplo, permite a um ISP alocar blocos de endereços para cada cliente).
- *Novo formato de cabeçalho.* O IPv6 usa um formato de pacote inteiramente novo e incompatível que inclui um conjunto de cabeçalhos opcionais.
- *Melhores opções.* O IPv6 permite que um pacote inclua um controle de informação opcional não disponível no IPv4.
- *Provisão para Protocol Extension.* Em vez de especificar todos os detalhes, a capacidade de extensão do IPv6 permite que o IETF adapte o protocolo a um novo hardware de rede e a novas aplicações.
- *Suporte para autoconfiguração e renumeração.* O IPv6 permite a um site mudar de um ISP para outro pela automação do requisito mudança de endereço.
- *Suporte para alocação de recurso.* O IPv6 inclui um fluxo de abstração e permite serviços diferenciados.

1.12 Migração para o IPv6

Como pode ser feita a mudança de IPv4 para IPv6 na Internet? Os projetistas consideraram cuidadosamente essa questão. Nos anos de 1990, a Internet já tinha crescido muito para simplesmente tirá-la do ar, mudar cada host e roteador e, então, reinicializá-la. Então, os programadores planejaram fazer a integração das alterações gradualmente ao longo do tempo. Usamos o termo *migração para o IPv6* para entender o conceito.

Muitos grupos propuseram planos para a migração para o IPv6. Esses planos podem ser agrupados em três grandes aspectos, que são:

- uma Internet IPv6 separada funcionando em paralelo;
- ilhas IPv6 conectadas por IPv4 até os ISPs instalarem o IPv6;
- gateways que fizessem a tradução entre o IPv4 e o IPv6.

Internets paralelas: conceitualmente, o plano pedia que os ISPs criassem uma Internet paralela funcionando com IPv6. Na prática, IPv6 e IPv4 podem compartilhar muitos dos caminhos subjacentes e equipamentos de rede (desde que os aparelhos estejam atualizados para funcionar com IPv6). Todavia, endereçamento e roteamento usados pelas duas versões de protocolo seriam totalmente independentes. Os proponentes argumentaram que, devido ao fato de o IPv6 oferecer muitas vantagens, todos iriam mudar para ele, de forma que a Internet IPv4 seria rapidamente desativada.

Ilhas IPv6: o plano permitia que organizações individuais começassem a usar o IPv6 antes que todos os ISP o estivessem usando. Cada organização seria uma ilha usando IPv6 no meio de um oceano IPv4. Para enviar um datagrama entre ilhas, o datagrama IPv6 seria embrulhado dentro de datagrama IPv4, enviado pela Internet, e desembrulhado quando chegasse à ilha de destino. Quando as ISPs adotassem o IPv6, os sites poderiam começar a enviar IPv6 a mais e mais destinos, até que toda a Internet estivesse usando IPv6. Alguns entusiastas do IPv6 não gostaram dessa proposta, pois ela não fornecia incentivo econômico suficiente para que as ISPs adotassem IPv6.

Gateways e traduções: a terceira proposta usa dispositivos de rede que fazem a tradução entre o IPv4 e o IPv6. Por exemplo, se um site escolher usar o IPv6, mas seus ISP ainda usarem IPv4, um dispositivo de gateway pode ser instalado entre o site e o ISP para fazer a operação de tradução. O gateway aceita os pacotes de saída IPv6, cria pacotes IPv4 equivalentes e os envia para o ISP encaminhá-los. Da mesma forma, quando um pacote IPv4 chegar do ISP, o gateway vai criar um pacote IPv6 equivalente e enviá-lo para a organização. Assim, os computadores da organização podem trabalhar com IPv6 mesmo que o ISP continue usando IPv4. Alternativamente, um site pode usar IPv4 mesmo que o resto da Internet tenha adotado IPv6.

Cada estratégia de migração tem vantagens e desvantagens. Ao final, surge uma questão central: qual incentivo econômico faz um usuário, empresa ou um ISP mudar?

De forma surpreendente, praticamente não há evidência de que o IPv6 ofereça muito para o consumidor comum, as organizações ou os provedores. Obviamente há excessos. Por exemplo, uma companhia cujo modelo de negócios envolve a venda de informação para anunciantes vai ser muito beneficiada se cada um usar um endereço IP separado, pois vai estar apta a identificar hábitos individuais de forma muito mais precisa do que quando uma família compartilha um computador ou um só endereço. Ao final, cada uma das estratégias de migração está sendo usada em algum lugar, mas nenhuma se sobressaiu como consenso amplamente aceito.

1.13 Sistema de pilha dupla (Dual Stack Systems)

Muitos capítulos neste texto discutem software de protocolo, geralmente conhecido como *protocol stack*. A iminente mudança para o IPv6 afetou o modo como o software de protocolo é projetado, especialmente para computadores individuais. A maior parte dos sistemas operacionais (por exemplo, Linux, Windows, e OS-X) já está classificada como *dual stack*. Ou seja, adicionalmente a todos os softwares necessários para o IPv4, o sistema contém todos os sistemas necessários para o IPv6. Na maioria dos sistemas, as duas versões não interagem. Isso quer dizer que cada um tem um endereço IP e pode mandar e receber pacotes. Entretanto, os endereços são diferentes e nenhum deles pode usar o endereço do outro (ou simplesmente desconhecem a existência um do outro). A ideia do dual stack está intimamente ligada ao conceito de Internet paralela discutido anteriormente.

O sistema dual stack permite que as aplicações escolham se vão usar o IPv4, o IPv6 ou ambos. Aplicações antigas continuam a usar IPv4. Todavia, um mecanismo dual stack possibilita que uma aplicação escolha dinamicamente, fazendo migração automática. Por exemplo, considere um navegador. Se um dado URL mapeia ambos os endereços, IPv4 e IPv6, o navegador pode tentar comunicar-se usando primeiramente o IPv6. Se a tentativa falhar, pode tentar o IPv4. Se o computador estiver conectado a uma rede IPv6, alcançará o destino. A comunicação IPv6 terá sucesso. Senão, o navegador automaticamente voltará e usará o IPv4.

1.14 Organização do texto

O material sobre TCP/IP foi escrito em três volumes. Este volume introduz a tecnologia TCP/IP. Nele são discutidos os fundamentos dos protocolos como TCP e IP, formatos de pacotes atuais e é mostrado como os protocolos

se ajustam na Internet. Em adição ao exame dos protocolos individuais, o texto destaca os princípios gerais em que se baseiam os protocolos de rede, e explica por que o protocolo TCP/IP se adapta facilmente a tantas tecnologias subjacentes de rede física. O texto aborda a arquitetura da Internet global e considera os protocolos que propagam informações de roteamento. Finalmente apresenta exemplos de aplicações de rede e explica como essas aplicações usam os protocolos TCP/IP.

O segundo e o terceiro volumes focam as implementações. O Volume II examina a implementação dos protocolos TCP/IP propriamente. Nesse volume se explica como o software do protocolo é organizado e discute-se a estrutura de dados, assim como as facilidades, tais como gerenciamento do tempo. Nele se apresentam algoritmos e exemplos de usos do código de um sistema de trabalho para ilustrar o conceito. No Volume III são consideradas as aplicações em rede e explica-se como elas usam o TCP/IP para comunicação. Nelas, o foco está no paradigma cliente-servidor, a base para toda a programação distribuída. Discute-se, ainda, a interface entre programas e protocolos,* e mostra-se como os programas cliente e servidor são organizados.

Até aqui, temos falado em termos gerais sobre a tecnologia TCP/IP e a Internet, sumarizando os serviços fornecidos e a história de seu desenvolvimento. O próximo capítulo fornece um breve sumário do tipo de hardware de rede usado na Internet. Seu propósito não é iluminar as nuances de um fornecedor de hardware em particular, mas atentar para as características de cada tecnologia que são de fundamental importância para uma arquitetura de Internet. Capítulos posteriores se aprofundam nos protocolos e na Internet, cumprindo três propósitos: explorar os conceitos gerais e revisar o modelo arquitetônico da Internet; examinar os detalhes dos protocolos TCP/IP; considerar os padrões de serviço das aplicações. Outros capítulos descrevem serviços que abrangem múltiplas máquinas, incluindo a propagação de informação de roteamento, resolução de nome e aplicações como a Web.

No apêndice que segue o texto principal, há uma lista em ordem alfabética das abreviações e termos usados na literatura e no texto. Como iniciantes geralmente acham novas terminologias esmagadoras e difíceis de lembrar, são encorajados a usarem a lista alfabética em vez de procurar nas páginas anteriores do texto.

1.15 Resumo

Uma internet consiste de um conjunto de redes conectadas que agem como um todo coordenado. A vantagem principal de uma internet é que ela

fornece interconexões universais enquanto permite a grupos individuais usarem qualquer hardware de rede que seja mais bem ajustado às suas necessidades. Vamos examinar os princípios básicos da comunicação pela internet em geral e os detalhes de um conjunto de protocolos de internet em particular. Vamos discutir também como os protocolos de internet são usados nela. Nossa tecnologia de exemplo, batizada de TCP/IP por causa dos seus dois protocolos principais, foi desenvolvida pela DARPA (Defense Advanced Research Projects Agency) e fornece as bases para a Internet global, que atualmente alcança mais de 2 bilhões de pessoas em países ao redor do mundo. A próxima versão de Protocolo de Internet (IPv6) inspira-se fortemente em conceitos, terminologia e detalhes da versão atual (IPv4). Portanto, os capítulos ao longo do texto vão examinar ambas as versões.

EXERCÍCIOS

1.1 Faça uma lista de todas as aplicações de Internet que você usa. Quantas são baseadas na web?

1.2 Desenhe o crescimento da tecnologia TCP/IP e acesso de Internet na sua organização. Quantos computadores, usuá rios e redes estavam conectados em cada ano?

1.3 Iniciando em 2000, a maior parte das companhias telefônicas começou a substituir suas redes de comutação convencional para redes com base em IP. A maioria das redes de telefonia vai usar somente protocolos IP. Por quê?

1.4 Descubra quando seu site mudou para IPv6 ou quando ele planeja mudar.

---

* Vamos seguir a convenção usual com inicial maiúscula para *Internet* quando se tratar especificamente de Internet global e minúscula para inter-nets privadas que usam a tecnologia TCP/IP.

** Muitas vezes, a DARPA foi chamada de *Advanced Research Projects Agency* (ARPA).

* IAB era conhecido originalmente como *Internet Activities Board.*

* Para evitar confusões e ambiguidades, a versão número 5 foi omitida; surgiram problemas devido a uma série de erros e mal-entendidos.

* O Volume III está disponível em duas versões: a primeira usa o Linux *socket interface* e a segunda usa o *Windows Sockets Interface*, definido pela Microsoft.

CAPÍTULO 2

Visão geral das tecnologias de rede subjacentes

CONTEÚDOS DO CAPÍTULO

2.1 Introdução

A Internet introduziu uma mudança fundamental em nosso modo de pensar sobre rede de computador. Inicialmente todos os esforços objetivaram a produção de um novo tipo de rede. A Internet, no entanto, introduziu um novo método de interconexão entre redes individuais e um conjunto de protocolos que permitiram aos computadores interagirem através de várias redes. Enquanto hardware de rede representa apenas um papel de menor importância na concepção global, o entendimento da tecnologia da Internet requer que se faça distinção entre os mecanismos de baixo nível fornecidos pelo hardware propriamente dito e as facilidades de alto nível fornecidas pelos protocolos TCP/IP. É também importante entender como as interfaces fornecidas pela tecnologia subjacente de comutação por pacotes afetam nossa escolha de abstrações de alto nível.

Este capítulo introduz conceitos e terminologia básicos de comutação por pacote, e então revisa algumas das tecnologias subjacentes de hardware que têm sido usadas em internets TCP/IP. Capítulos posteriores descrevem como networks físicas são interconectadas e como os protocolos TCP/IP acomodam vastas diferenças no hardware. Enquanto a lista apresentada aqui certamente não é abrangente, ela demonstra claramente a variedade entre redes físicas nas quais TCP/IP opera. O leitor pode pular com segurança alguns dos detalhes técnicos, mas deve tentar entender a ideia de comutação de pacotes e tentar imaginar a construção de um sistema

homogêneo de comunicação usando esses hardwares heterogêneos. Mais importante, o leitor deve atentar para os detalhes dos esquemas de endereçamento que várias tecnologias usam; capítulos posteriores discutirão detalhadamente como protocolos de alto nível usam endereços de hardware.

2.2 Duas abordagens para comunicação por rede

A partir de uma perspectiva de hardware, redes são frequentemente classificadas pelas formas de energia que elas usam e pelo meio no qual a energia viaja (por exemplo, sinais elétricos em fio de cobre, pulsos de luz em fibra óptica e ondas de radiofrequência transmitidas pelo espaço). De uma perspectiva de comunicação, tecnologias de rede podem ser divididas em duas grandes categorias que dependem da interface que fornecem: *conexão orientada* (às vezes chamada *comutada por circuito*) e *sem conexão* (às vezes chamada *comutada por pacotes*)*. Redes por conexões orientadas operam formando uma *conexão* dedicada, ou *circuito*, entre dois pontos. Sistemas telefônicos antigos usavam tecnologia conexão orientada – uma chamada telefônica estabelecia uma conexão desde o telefone de origem até o escritório de comutação local, através de linhas tronco, até um escritório de comutação remoto, e finalmente ao telefone de destino. Enquanto uma conexão estava em andamento, o equipamento enviava sinais de voz do microfone para o receptor. Como era dedicado um caminho na rede para cada par de pontos finais de comunicação, um sistema de conexão orientada podia garantir que a comunicação fosse contínua e sem interrupção. Ou seja, uma vez que o circuito estava estabelecido, nenhuma outra atividade diminuiria a capacidade dele. Uma desvantagem da tecnologia de conexão orientada surgiu devido ao custo: os custos do circuito são fixos, independentemente do uso. Por exemplo, a taxa cobrada para uma chamada permanecia fixa, mesmo durante o tempo em que nenhuma das partes estava falando.

Redes sem conexão, o tipo frequentemente usado para conectar computadores, têm uma abordagem totalmente diferente. Em um sistema sem conexão, os dado a serem transferidos por uma rede dividem-se em pequenos pedaços, chamados *pacotes*, que são multiplexados em conexões de alta capacidade entre máquinas. Um pacote, que normalmente contém somente poucas centenas de bytes de dados, leva uma identificação que habilita o hardware de rede a saber como mandá-lo ao destino especificado. Por exemplo, um arquivo grande a ser transmitido entre duas máquinas

deve ser dividido em alguns pacotes que são enviados através da rede um de cada vez. O hardware de rede entrega os pacotes ao destino especificado, onde um software os remonta em um único arquivo. A principal vantagem da comutação por pacotes é que uma comunicação múltipla entre computadores pode ser feita de forma concomitante, com conexões entre máquinas compartilhadas por todos os pares de computadores que estão se comunicando. A desvantagem, claro, é que, com o aumento da atividade, um dado par de computadores em comunicação recebe menos da capacidade da rede. Ou seja, sempre que uma rede comutada por pacote ficar sobrecarregada, os computadores que a estão usando precisarão esperar para que possam mandar pacotes adicionais.

Apesar do potencial inconveniente de não estarem aptas para garantir a capacidade da rede, redes sem conexão têm se tornado extremamente populares. As motivações principais para adoção da comutação por pacote são custo e desempenho. Como múltiplos computadores podem compartilhar os canais subjacentes da rede, poucas conexões são necessárias e o custo se mantém baixo. Pelo fato de engenheiros serem aptos a construírem hardwares de alta velocidade para comutação por pacote, a capacidade geralmente não é um problema. Tantas interconexões de computador usam rede sem conexão que, durante todo o restante do texto, assumiremos que o termo *rede* se refere a redes com tecnologia sem conexão, a menos que tenha indicação em contrário.

2.3 Redes locais e de longa distância (WAN e LAN)

As redes de dados que cobrem grandes distâncias geográficas (por exemplo, os Estados Unidos continental) são fundamentalmente diferentes daquelas que cobrem curtas distâncias (uma sala, por exemplo). Para ajudar a caracterizar as diferenças entre capacidade e uso pretendido, as tecnologias de comutação de pacotes normalmente são divididas em duas categorias gerais: *redes de longa distância* (*WANs*) e *redes locais* (*LANs*), sendo que ambas não possuem definições formais. Em vez disso, os fornecedores aplicam os termos de forma livre para ajudar os clientes a diferenciar as tecnologias.

Tecnologias WAN, às vezes chamadas *long haul networks* (redes de transporte a longa distância), fornecem comunicação a longas distâncias. A maioria das tecnologias de WAN não limita o leque de distância; uma WAN pode permitir que as extremidades de uma comunicação sejam distanciadas arbitrariamente. Por exemplo, uma WAN pode cobrir um continente ou

ainda unir computadores através de um oceano. Normalmente, as WANs operam em velocidades mais baixas que as LANs, e possuem muito mais atraso entre as conexões. As velocidades típicas para uma WAN variam de 100 Mbps (milhões de bits por segundo) até 10 Gbps (bilhões de bits por segundo). Os atrasos através de uma WAN podem variar de alguns milissegundos até várias dezenas de segundos.**

Tecnologias de LAN fornecem as conexões de mais alta velocidade entre os computadores, mas sacrificam a capacidade de cobrir longas distâncias. Por exemplo, uma LAN típica cobre uma pequena área, como um único prédio ou um pequeno *campus*, e opera entre 100 Mbps e 10 Gbps. Como as tecnologias de LAN cobrem curtas distâncias, elas oferecem menores atrasos que as WANs. O atraso através de uma LAN pode ser tão curto quanto algumas décimos de milissegundos ou tão longo como 10 milissegundos.

2.4 Esquemas de endereçamento de hardware

Vamos ver que os protocolos de Internet devem lidar com um aspecto particular do hardware de rede: esquemas de endereçamento heterogêneo. Cada tecnologia de hardware define um *mecanismo de endereçamento* que os computadores utilizam para especificar o destino para um pacote. Cada computador conectado a uma rede recebe um *endereço* exclusivo, normalmente um inteiro. Um pacote enviado através de uma rede inclui dois endereços: um *endereço de destino*, que especifica o destinatário desejado, e um *endereço fonte*, que especifica o emissor. O endereço de destino aparece no mesmo local em todos os pacotes, possibilitando ao hardware de rede verificar com facilidade o endereço de destino. Um emissor precisa saber o endereço do destinatário desejado, colocando-o no campo de endereço de destino de um pacote antes de transmiti-lo.

A próxima seção analisa quatro exemplos de tecnologias de rede que vêm sendo usadas na Internet:

- Ethernet (IEEE 802.3);
- Wi-Fi (IEEE 802.11);
- ZigBee (IEEE 802.15.4);
- redes ponto a ponto de longa distância (SONET).

Nossa discussão sobre tecnologias vai encobrir a maior parte dos detalhes de hardware, pois o propósito é destacar os caminhos nos quais o

hardware subjacente influenciou escolhas de projeto nos protocolos.

2.5 Ethernet (IEEE 802.3)

Ethernet é um nome dado a uma tecnologia LAN de comutação de pacotes desenvolvida pelo Xerox PARC* no início da década de 1970. Em 1978, a Xerox Corporation, a Intel Corporation e a Digital Equipment Corporation padronizaram a Ethernet: o IEEE lançou uma versão compatível do padrão usando o número de padrão *802.3*. A Ethernet tornou-se a tecnologia de LAN mais popular: agora, ela aparece em praticamente todas as redes corporativas e pessoais, e o formato de pacote Ethernet é usado de vez em quando através de redes de longa distância. As versões atuais da Ethernet são conhecidas como *Gigabit Ethernet (GigE)* e *10 Gigabit Ethernet (10GigE)*, pois elas transferem dados a 1Gbps e 10Gbps, respectivamente. A próxima geração de tecnologias opera a 40 e 100 gigabits por segundo. Uma rede Ethernet consiste em um *comutador Ethernet* ao qual múltiplos computadores são ligados.* Um comutador pequeno pode conectar centenas de computadores. Conexões entre um computador e um comutador consistem de fios de cobre para baixas velocidades ou fibras ópticas para velocidades maiores. A Figura 2.1 ilustra a topologia de uma Ethernet.

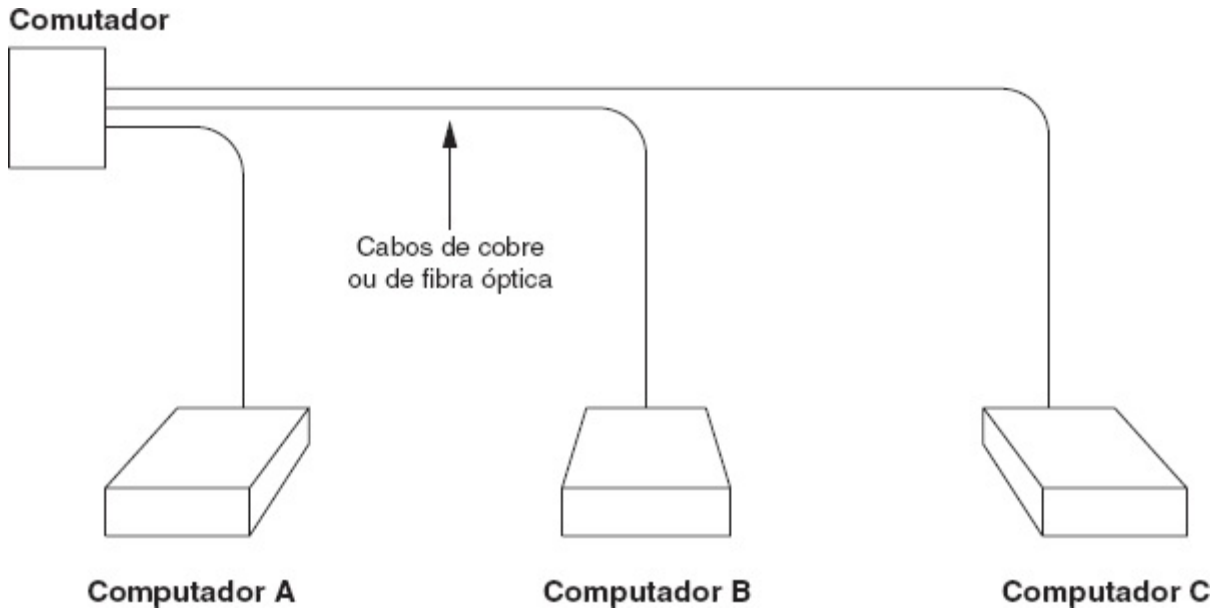


Figura 2.1 Ilustração de conexões usadas com uma Ethernet. Cada computador se conecta a um comutador central.

Um comutador Ethernet é um aparelho eletrônico que normalmente fica em um gabinete de fiação. Quando se usam fios de cobre, a conexão entre um comutador e um computador deve ter menos de 100 metros de distância; conexões ópticas podem exceder em muito essa distância. Cada computador precisa ter uma *Placa de Interface de Rede* (*Network Interface Card - NIC*) que opera como um dispositivo I/O que pode mandar e receber pacotes.

2.5.1 Capacidade da Ethernet

Dissemos que uma *Ethernet Gigabit* transfere dados a 1 gigabit por segundo (1000 megabits por segundo). Então, o nome formal *1000Base-T* é aplicado para versões que usam par de fios de cobre trançados. Uma norma IEEE relacionada conhecida como *1000Base-X* especifica a transmissão Ethernet por fibra óptica. Basicamente, a versão óptica converte um pacote Ethernet em pulsos de luz, que são transferidos por meio de fibra óptica. As principais vantagens das fibras ópticas são: maior capacidade e imunidade à interferência elétrica. A capacidade de uma fibra óptica é suficiente para suportar taxas de bit muito maiores que 10 Gbps. Entretanto, engenheiros estão desenvolvendo tecnologias Ethernet de 40 e 100 Gbps que operam com fibra óptica.

2.5.2 Negociação automática

Um comutador Ethernet moderno não se restringe a uma velocidade. Pelo contrário, o comutador pode operar a 10, 100, 1000 ou mesmo a 10000 Mbps. Um conjunto de velocidades é possível na maioria das placas de interface de rede, assim como nos comutadores. O aspecto importante da Ethernet multivelocidade está na configuração automática. Quando um cabo é plugado, ambos os terminais entram em fase de negociação. A negociação determina o tipo de cabo (reto ou cross-over) e a velocidade máxima que o outro lado da conexão suporta. Os dois lados concordam em operar na máxima velocidade que ambos suportam. A negociação automática fornece retrocompatibilidade – um computador com uma placa de rede velha e lenta pode se ligar sem qualquer alteração a um comutador Gigabit. O formato dos pacotes Ethernet não depende da velocidade do link, o que significa que protocolos TCP/IP podem continuar sem saber da velocidade negociada do link.

2.5.3 Propriedades importantes de uma Ethernet

Capacidade de transmissão. Capacidade de *transmissão* da Ethernet significa que um emissor pode especificar que um dado pacote deve ser entregue a todos os computadores que estejam ligados à rede. Na prática, comutadores normalmente implementam a transmissão fazendo uma cópia do pacote para cada computador. Vamos ver que o TCP/IP depende da transmissão Ethernet.

Best-effort delivery semantics. A Ethernet usa *best-effort delivery semantics*, que significa que a rede tenta entregar os pacotes, o hardware não garante a entrega e não informa ao emissor se o pacote foi ou não entregue. Se acontecer de a máquina de destino estar desligada ou com o cabo desplugado, os pacotes enviados a ela serão perdidos e o emissor não será notificado. Mais importante, se múltiplos computadores tentarem enviar pacotes a um dado computador ao mesmo tempo, um comutador pode ficar sobrecarregado e começar a descartar pacotes. Vamos ver mais adiante que best-effort semantics tem um conceito-chave no projeto de protocolos TCP/IP.

2.5.4 Endereços 48-Bit Ethernet MAC (hardware)

A IEEE define um esquema de endereçamento 48-bit MAC que é usado com Ethernet e com outras tecnologias de rede. A abreviação MAC significa *Media Access Control* (Controle de Acesso à Mídia), e é usada para tornar claro o propósito do endereço. Um endereço MAC é atribuído a cada placa de interface de rede. Para garantir a unicidade, um fornecedor de hardware de Ethernet deve comprar da IEEE um bloco de endereços MAC e atribuir um endereço a cada placa de interface de rede que for fabricada. A atribuição significa que não há dois hardwares de interfaces com o mesmo endereço Ethernet.

Como cada endereço é atribuído a cada dispositivo de hardware, às vezes usamos o termo *endereço de hardware* ou *endereço físico*. Para nossos propósitos, usaremos esses termos indistintamente.

Observe a importante propriedade de um endereço MAC a seguir.

> *Um endereço Ethernet é atribuído a uma placa de interface de rede, não a um computador; transferindo a placa de interface para um novo computador ou substituindo uma placa de interface que tenha falhado,*

o endereço Ethernet do computador muda.

Saber que uma mudança no hardware pode mudar um endereço Ethernet explica por que protocolos de nível mais alto são projetados para acomodarem mudanças de endereços.

O esquema de endereçamento 48-bit MAC propicia três tipos de endereços descritos a seguir.

- *Unicast.*
- *Broadcast.*
- *Multicast.*

Um endereço *unicast* é um valor único atribuído a uma placa de interface de rede, como descrito anteriormente. Se o endereço de destino em um pacote é um endereço unicast, o pacote será entregue a apenas um computador (ou não será enviado, se nenhum computador da rede tiver o endereço especificado).

Um endereço *broadcast* é constituído todo ele só por 1, e é reservado para transmitir para todas as estações simultaneamente. Quando um comutador recebe um pacote só com 1 no campo de endereço de destino, ele entrega uma cópia do pacote a cada computador na rede com exceção do emissor.

Um endereço *multicast* fornece uma forma limitada de transmissão na qual um subconjunto de computadores na rede aceita atender a um dado endereço multicast. O conjunto de computadores participantes é chamado um *grupo multicast*. A placa de interface de rede em um computador deve ser configurada para se juntar a um grupo multicast ou a interface irá ignorar os pacotes enviados ao grupo. Veremos que os protocolos TCP/IP usam multicast e que o IPv6 depende dele.

2.5.5 Formato do frame Ethernet e tamanho de pacote

Devido ao termo *pacote* ser genérico e poder se referir a qualquer tipo de pacote, usaremos o termo *frame* para nos referirmos ao pacote que é definido por tecnologias de hardware.* Os frames Ethernet têm comprimentos variáveis, sem frames menores que 64 octetos** ou maiores

que 1514 octetos (cabeçalho e data). Como esperado, cada frame Ethernet contém um campo que guarda o endereço 48-bit de um destino e outro campo que guarda o endereço 48-bit do emissor. Quando transmitido, o frame inclui também um 4-octet *Cyclic Redundancy Check* (*CRC*), que é usado para verificar erros de transmissão. Como o campo CRC é adicionado pelo hardware de envio e verificado pelo hardware de recebimento, não é visível para o software de camadas mais altas do protocolo. A Figura 2.2 ilustra as partes pertinentes de um frame Ethernet.

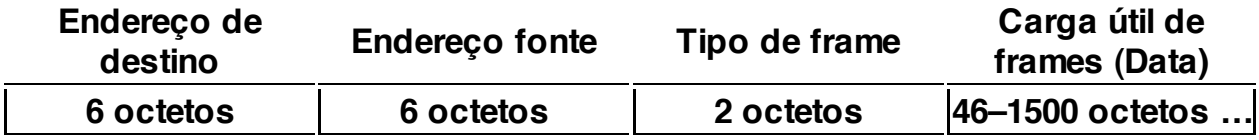


Figura 2.2 Formato de frame Ethernet. Os campos não estão em escala.

Na figura, os três primeiros campos constituem em *cabeçalho* para o frame, e os campos restantes são a *carga útil*. Os pacotes usados na maioria das tecnologias de rede seguem o mesmo padrão: o pacote consiste de um pequeno cabeçalho com campos fixos seguido por um campo de carga útil com tamanho variável. O tamanho máximo de carga útil em um frame Ethernet é 1500 octetos. Como a Ethernet foi adotada de forma universal, a maioria dos ISPs mudou suas redes para se ajustar com a carga Ethernet. Podemos fazer então o resumo a seguir.

O tamanho de carga Ethernet de 1500 octetos se tornou um padrão de fato; mesmo que se usem outras tecnologias, os ISPs tentam projetar suas redes de forma que um pacote possa ter 1500 octetos de dados.

Além dos campos que identificam a fonte e o destino do frame, um frame Ethernet contém um 16-bit inteiro que identifica o tipo de dado que está sendo transportado. A maioria das tecnologias de pacote inclui um campo de tipo. Do ponto de vista da Internet, o tipo do frame é essencial, pois significa que frames Ethernet são *autoidentificáveis*. Quando um frame chega a uma determinada máquina, o software do protocolo define qual módulo de protocolo deve processar esse tipo de frame. As maiores vantagens dos frames autoidentificáveis são que eles permitem que

protocolos múltiplos sejam usados em conjunto em um único computador e múltiplos protocolos sejam misturados na mesma rede física sem interferência.

Por exemplo, pode-se ter um programa de aplicação em um computador usando protocolos Internet, enquanto outra aplicação no mesmo computador usa um protocolo experimental local. O sistema operacional verifica o campo do tipo de cada frame que chega e decide qual módulo deve processar seu conteúdo. Veremos que o campo de tipo pode ser usado para definir protocolos múltiplos na mesma família. Por exemplo, devido aos protocolos TCP/IP incluírem diversos protocolos que podem ser enviados por uma Ethernet, o TCP/IP define vários tipos de Ethernet.

2.6 Wi-Fi (IEEE 802.11)

A IEEE desenvolveu uma série de padrões para redes wireless que estão intimamente ligadas à Ethernet. O mais conhecido tem número-padrão IEEE *802.11*, seguido por um sufixo (por exemplo, *802.11g* ou *802.11n*). O conjunto de normas pode interoperar, o que significa que um dispositivo sem fio pode incluir hardware para vários padrões, inclusive o que dá a velocidade máxima. Um consórcio de fornecedores de equipamentos de rede adotou o termo marketing de *Wi-Fi* para abranger equipamento que utiliza os padrões wireless IEEE, e existem muitos dispositivos Wi-Fi.

Cada um dos padrões Wi-Fi pode ser usado de duas maneiras: como uma tecnologia de acesso em que uma única estação-base (chamada de *ponto de acesso*) se conecta a vários clientes (por exemplo, os usuários com laptops), ou em uma configuração ponto a ponto usada para conectar só dois rádios sem fio. IEEE também definiu uma tecnologia de maior velocidade destinada principalmente para interconexões ponto a ponto. Comercializada como *Wi-MAX* e com o número-padrão *802.16* a ela atribuído, a tecnologia é de maior interesse para os fornecedores de rede ou empresas que precisam conectar dois sites.

2.7 ZigBee (IEEE 802.15.4)

Além de conectar computadores convencionais, uma internet pode ser utilizada por dispositivos incorporados. O conceito de dispositivos de conexão é por vezes referido como uma *Internet das coisas*. Claro, cada dispositivo que se conecta à Internet deve ter um processador embutido e incluir uma interface de rede. A IEEE criou padrão 802.15.4 para uma

tecnologia de rede sem fio de baixa potência destinada para suportar conexões de pequenos dispositivos embarcados. O aspecto de baixa potência torna rádios 802.15.4 atraentes para dispositivos que funcionam com bateria.

Um consórcio de fornecedores escolheu o termo *ZigBee* para se referir a produtos que utilizam o padrão de IEEE 802.15.4 para rádios e executam uma pilha de protocolo específico, que inclui IPv6 mais protocolos que permitem um conjunto de nódulos sem fio para se organizar em uma malha que pode encaminhar pacotes para e a partir da Internet.

A tecnologia IEEE 802.15.4 é um exemplo interessante de extremos para TCP/IP. O tamanho do pacote é de 127 octetos, mas apenas 102 octetos de carga útil estão disponíveis. Além disso, o padrão define dois formatos de endereço: um utiliza endereços MAC de 64 bits e o outro usa endereços MAC de 16 bits. A escolha do modo de endereçamento é negociada na inicialização.

2.8 Optical Carrier e Packet Over SONET (OC, POS)

As companhias telefônicas originalmente conceberam circuitos digitais para conduzir chamadas de voz digitalizadas; só mais tarde é que os circuitos digitais dessas companhias se tornaram importantes para redes de dados. Consequentemente, as taxas de circuitos de dados disponíveis não são potências de dez. Em vez disso, eles foram escolhidos para conduzir múltiplos de 64 Kbps, pois uma chamada de voz digitalizada usa uma codificação conhecida como *Pulse Code Modulation* (*PCM*), que produz 8 mil amostras por segundo, sendo que cada amostra é de 8 bits.

A tabela da Figura 2.3 lista algumas taxas comuns de transmissão de dados usados na América do Norte e na Europa.

Nome	Taxa em bit	Circuitos de voz	Localização
–	0,064 Mbps	1	
T1	1,544 Mbps	24	América do Norte
T2	6,312 Mbps	96	América do Norte
T3	44,736 Mbps	672	América do Norte
T4	274,760 Mbps	4032	América do Norte
E1	2,048 Mbps	30	Europa
E2	8,448 Mbps	120	Europa
E3	34,368 Mbps	480	Europa

E4	**139,264 Mbps**	**1920**	**Europa**

Figura 2.3 Exemplo de taxas de dados disponíveis em circuitos digitais alugados de uma empresa de telefonia.

Circuitos digitais de taxa mais elevada requerem o uso de fibras. Além de normas que especificam a transmissão de altas taxas de dados em cobre, as empresas de telefonia desenvolveram padrões para transmissão das mesmas taxas em fibra óptica. A Figura 2.4 lista exemplos de padrões *Optical Carrier* (*OC*) e a taxa de dados de cada uma. Um sufixo em "OC" denota capacidade.

Norma óptica	Taxa em bit	Circuitos de voz
OC-1	**51,840 Mbps**	**810**
OC-3	**155,520 Mbps**	**2430**
OC-12	**622,080 Mbps**	**9720**
OC-24	**1.244,160 Mbps**	**19440**
OC-48	**2,488 Gbps**	**38880**
OC-96	**4,976 Gbps**	**64512**
OC-192	**9,952 Gbps**	**129024**
OC-256	**13,271 Gbps**	**172032**

Figura 2.4 Exemplo de taxas de dados disponíveis em circuitos digitais de alta capacidade que usam fibra óptica.

O termo refere-se ao *SONET*, um protocolo de framing que permite que um portador possa multiplexar várias chamadas telefônicas digitais de voz em uma única conexão. O SONET é tipicamente usado em conexões OC. Assim, se um ISP aluga uma conexão OC-3, o ISP pode precisar usar SONET framing. O termo de *Packet Over SONET* (*POS*) refere-se a uma tecnologia usada para enviar pacotes usando SONET framing.

2.9 Redes ponto a ponto

Do ponto de vista do TCP/IP, qualquer sistema de comunicação utilizado para transmitir pacotes é classificado como uma *rede*. Se o sistema de comunicação conecta exatamente dois terminais, é conhecido como uma *rede ponto a ponto*. Assim, um circuito dedicado de dados é um exemplo

de uma rede ponto a ponto.

Os puristas se opõem a usar o termo "rede" para descrever uma conexão ponto a ponto, pois reservamos o termo para tecnologias que permitem a comunicação entre um conjunto de computadores. Veremos, no entanto, que a classificação de uma conexão como uma rede ajuda a manter a consistência. Por enquanto, só precisamos observar que uma rede ponto a ponto difere de uma rede convencional em um ponto significativo: devido a apenas dois computadores estarem conectados, não são necessários endereços de hardware. Quando discutimos ligação de endereço internet, a falta de endereços de hardware faz as redes ponto a ponto uma exceção.

A conexão discada fornece um exemplo de uma rede ponto a ponto. No início da Internet, o acesso usava conexões dial-up nas quais um modem dial-up era usado para fazer uma chamada telefônica a outro modem. Uma vez que era feita a conexão, os dois modems podiam usar tons de áudio para enviar os dados. Do ponto de vista do TCP/IP, discar uma chamada telefônica é equivalente à conexão por fio. Uma vez que a chamada foi atendida por um modem na outra extremidade, há uma conexão a partir de um terminal para outro, e a conexão permanece enquanto for necessário. A moderna tecnologia óptica comutada fornece outra forma de rede ponto a ponto. Um gerente da rede pode solicitar que um caminho óptico seja configurado através de uma série de comutadores ópticos. Do ponto de vista do TCP/IP, o caminho é um circuito ponto a ponto de alta capacidade análogo a um circuito dedicado ou uma conexão discada. Capítulos posteriores discutem o conceito tunneling e overlay networks, que fornecem outra forma de conexões ponto a ponto.

2.10 Tecnologia VLAN e domínios de transmissão

Dissemos que um comutador Ethernet forma uma única rede de área local, ligando um conjunto de computadores. Uma forma mais avançada de comutadores, que é conhecida como um comutador virtual de rede de área local – *Virtual Local Area Network* (*VLAN*) –, permite a um gerente configurar o comutador para operar como se fosse vários comutadores menores. Dizemos que o gerente pode criar uma ou mais *redes virtuais*, especificando quais os computadores se conectam a qual VLAN.

Um gerente pode usar VLANs para computadores separados de acordo com suas políticas. Por exemplo, uma empresa pode ter uma VLAN para os funcionários e uma VLAN separada para os visitantes.

Computadores da VLAN dos empregados podem fornecer o privilégio de maior acesso que os computadores da VLAN dos visitantes.

A chave para entender as VLANs e sua interação com protocolos de Internet envolve a forma como um comutador VLAN se relaciona com broadcast e multicast. Dizemos que cada VLAN define um *domínio de transmissão*, o que significa que, quando um computador envia um pacote broadcast, este só é emitido para o conjunto de computadores da mesma VLAN. A mesma definição é válida para multicast. Ou seja, a tecnologia VLAN emula um conjunto de redes físicas independentes. Os computadores em uma determinada VLAN compartilham o acesso broadcast e multicast, mas, assim como em redes físicas separadas, broadcast ou multicast enviados em uma determinada VLAN não se propagam para outras VLANs.

Como os protocolos de Internet lidam com VLANs? A resposta é que protocolos de Internet não fazem distinção entre a VLAN e uma rede física independente. Podemos fazer então o resumo a seguir:

> *Do ponto de vista dos protocolos Internet, uma VLAN é tratada exatamente como uma rede física separada.*

2.11 Bridging (ponte)

Usamos o termo *bridging* para nos referir a tecnologias que transportam uma cópia de um frame a partir de uma rede para outra, e o termo *bridge* para um mecanismo que implementa o bridging. A motivação para o bridging é formar uma única grande rede usando bridges para conectar redes menores.

A ideia-chave é que, quando se transfere uma cópia de um frame, uma bridge não faz qualquer alteração. Em vez disso, o frame é simplesmente replicado e transmitido através da outra rede. Em particular, uma ponte não altera os endereços fonte ou de destino no frame. Assim, os computadores nas duas redes podem se comunicar diretamente. Além disso, os computadores usam exatamente a mesma interface de hardware, formato de frame e endereços MAC quando se comunicam através de uma bridge da mesma forma que ao se comunicarem localmente – os computadores desconhecem por completo que a ponte ocorreu. Para captar o conceito, podemos dizer que a bridge é *transparente* (ou seja, invisível) a computadores usando a rede.

Originalmente, os fornecedores de equipamentos de rede vendiam as

bridges como dispositivos físicos separados. Com o advento das redes comutadas modernas, as bridges não eram mais viáveis. Apesar da mudança na tecnologia, o bridging ainda é usado em muitos sistemas de rede. A diferença é que as bridges estão agora incorporadas em outros dispositivos. Por exemplo, os ISPs que fornecem serviços para residências e empresas usam bridges em equipamentos como *modems a cabo* e em hardware de *Digital Subscriber Line* (*DSL*). Frames Ethernet transmitidos através delas em uma residência são bridged ao ISP, e vice-versa. Computadores na casa usam a Ethernet local como se o roteador do ISP estivesse ligado diretamente; um roteador no ISP se comunica através da Ethernet local como se os computadores dos clientes fossem locais.

Do nosso ponto de vista, o ponto mais importante a entender sobre bridging surge a partir do sistema de comunicação resultante da constatação a seguir:

> *Porque o bridging esconde os detalhes de interconexão, um conjunto de bridged Ethernets age como uma única Ethernet.*

Na verdade, as bridges fazem mais do que reproduzir frames de uma rede para outra: a bridge toma decisões inteligentes sobre quais frames transmitir. Por exemplo, se um usuário tem um computador e uma impressora em sua residência, a bridge em um modem de cabo não vai enviar cópias de frames para o ISP se os frames vão do computador do usuário para a impressora, ou vice-versa.

Como é que uma bridge sabe se é para transmitir frames? Bridges são chamadas de bridges de *adaptação* ou *de aprendizagem*, pois usam o tráfego de pacotes para saber quais computadores estão em cada rede. Recorde-se de que um frame contém o endereço do remetente, assim como o endereço de um receptor. Quando ela recebe um frame, a bridge registra o endereço de origem de 48 bits. Em uma rede típica, cada computador (ou dispositivo) vai enviar pelo menos um frame broadcast ou multicast, o que significa que a bridge vai aprender o endereço MAC do computador. Uma vez que ela aprende os endereços dos computadores, a bridge examina cada frame e verifica a lista antes de encaminhar uma cópia. Se tanto o emissor como o receptor estão na mesma rede, não é necessário encaminhamento.

As vantagens de bridging adaptativo devem ser óbvias. Por ele usar endereços encontrados no tráfego normal, um mecanismo de bridging é

tanto transparente como automático – as pessoas não precisam configurá-lo. Como ele não encaminha o tráfego desnecessariamente, uma bridge ajuda a melhorar o desempenho. Podemos fazer então o resumo a seguir.

> *Uma bridge Ethernet adaptativa conecta duas redes Ethernet, encaminha frames de uma para outra e usa os endereços de origem dos pacotes para saber quais computadores estão em qual Ethernet. Uma bridge usa o local dos computadores para eliminar encaminhamentos desnecessários.*

2.12 Congestionamento e perda de pacote

Na prática, a maioria das tecnologias de rede funciona tão bem que é fácil assumir total confiabilidade. No entanto, a menos que um sistema de pacotes preserve a capacidade antes de cada utilização, o sistema é suscetível a *congestionamento* e *perda de pacotes*. Para entender o porquê, considere um exemplo simples: um comutador Ethernet com apenas três computadores conectados. Suponha que dois computadores enviam dados para um terceiro, como a Figura 2.5 ilustra.

Imagine que cada uma das conexões com o comutador opera a 1 Gbps, e considere o que acontece se os computadores A e B enviarem dados continuamente para o computador C. A e B vão encaminhar os dados a uma taxa agregada de 2 Gbps. Como a conexão de C só pode lidar com metade dessa taxa, o link para *C* vai se tornar *congestionado*.

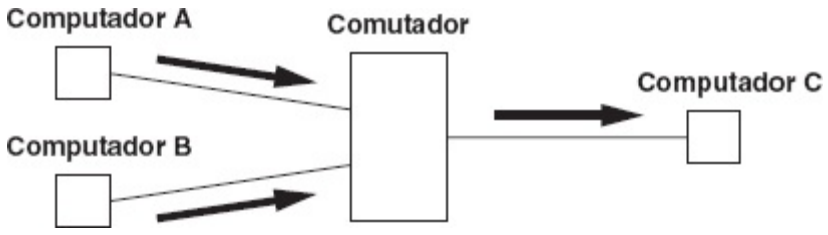


Figura 2.5 Uma Ethernet com setas indicando o fluxo do tráfego.

Para entender o que acontece com o tráfego, lembre-se de que a Ethernet usa best-effort delivery semantics. O comutador não tem como informar a A e B que um link de saída está congestionado, e não há maneira de parar o tráfego de entrada. Internamente, um comutador tem uma quantidade finita de espaço no buffer. Uma vez que todos os buffers são usados, o comutador

deve descartar frames adicionais que chegam. Assim, mesmo com apenas três computadores conectados a uma rede Ethernet, é possível que os pacotes sejam perdidos.

Neste ponto, só o que importa é compreender que o congestionamento e a perda podem realmente ocorrer em redes por pacotes. Um capítulo mais à frente examina o TCP e os mecanismos que ele usa para evitar o congestionamento.

2.13 Resumo

Protocolos de Internet são projetados para acomodar uma grande variedade de tecnologias de hardware subjacentes. Para compreender algumas das decisões de projeto, é necessário estar familiarizado com os conceitos básicos de hardware de rede.

Tecnologias de comutação de pacotes são geralmente divididas nos tipos conexão orientada e sem conexão. Uma rede de comutação de pacotes é classificada mais como uma rede de longa distância ou rede de área local, dependendo de se o hardware suporta a comunicação a longas distâncias ou se limita a curtas distâncias.

Analisamos várias tecnologias utilizadas na Internet, incluindo Ethernet, Wi-Fi, ZigBee e os circuitos digitais dedicados que podem ser usados para longa distância. Consideramos também VLANs e redes bridged. Embora os detalhes das tecnologias de rede específicas não sejam importantes, então uma ideia geral descrita a seguir emergiu.

> *Os protocolos Internet são extremamente flexíveis; uma grande variedade de tecnologias subjacentes de hardware foi usada para transportar o tráfego na Internet.*

Cada tecnologia de hardware define um esquema de endereçamento conhecido como endereços MAC. As diferenças são dramáticas: a Ethernet usa endereços MAC de 48 bits, enquanto as redes 802.15.4 podem usar endereços MAC de 16 bits ou 64 bits. Como o objetivo é interligar hardware de rede arbitrário, a Internet deve acomodar todos os tipos de endereços MAC.

EXERCÍCIOS

2.1 Faça uma lista das tecnologias de rede usadas em seu local.

2.2 Se frames Ethernet são enviados através de um circuito OC-192 dedicado, quanto tempo leva para transmitir os bits do maior frame Ethernet possível? E o menor frame possível? (Nota: você pode excluir o CRC de seus cálculos.)

2.3 No Estudo da tecnologia de comutador Ethernet, qual é o *spanning tree* algorithm e por que ele é necessário?

2.4 Leia a respeito do IEEE 802.1Q. O que faz a identificação da VLAN acontecer?

2.5 Qual o maior tamanho de pacote que pode ser enviado por redes sem fio 4G usado pelos telefones celulares?

2.6 Se seu site usa Ethernet, descubra o tamanho do maior e do menor comutador (por exemplo, o número de portas às quais os computadores podem se conectar). Quantos comutadores estão interconectados?

2.7 Qual a quantidade máxima de dados que podem ser transmitidos em um pacote Wi-Fi?

2.8 Qual a quantidade máxima de dados que podem ser transmitidos em um pacote ZigBee?

2.9 Que característica em um canal de satélite de comunicação é mais desejável? E qual a menos desejável?

2.10 Encontre o limite inferior de tempo que leva uma rede que opera a 100 Mbps, 1000 Mbps e 10 Gbps para transferir um gigabyte de dado.

2.11 O processador, o disco e o barramento interno no seu computador funcionam rápido o suficiente para ler dados de um arquivo no disco e enviá-lo através de uma rede de 10 gigabits por segundo?

2.12 Um roteador sem fio que utiliza a tecnologia Wi-Fi para conectar laptops à Internet tem uma conexão Ethernet e conexões sem fio com vários laptops. Considere os dados fluindo dos laptops para a Ethernet. Se a conexão Ethernet opera a 1 Gbps, quantos laptops devem estar ligados para causar congestionamento na Ethernet? (Dica: o que é a taxa máxima de dados de uma única conexão Wi-Fi?)

---

* Tecnologias híbridas também são possíveis, mas esses detalhes não importam para a nossa discussão.

** Excepcionalmente, longos atrasos podem resultar de redes WAN que se comunicam enviando sinais para satélites em órbita e retornando para outro ponto

na Terra.

* Vamos descrever redes como conexões de computadores, mas elas também podem conectar aparelhos, tais como impressoras, que tenham conexões de rede.

* O termo *frame* deriva da comunicação em linhas seriais nas quais o emissor "frames" os dados adicionando caracteres especiais antes e depois dos dados transmitidos.

** Tecnicamente, o termo byte se refere a um tamanho de caractere dependente de hardware: profissionais de rede usam o termo octeto porque ele se refere a uma quantidade de 8-bit em todos os computadores.

CAPÍTULO 3

Conceito de interligação de redes e modelo de arquitetura

CONTEÚDOS DO CAPÍTULO

3.1 Introdução

Até aqui, vimos os detalhes de transmissão de baixo nível através de redes de dados individuais, a base sobre a qual toda a comunicação por computador é construída. Este capítulo dá um salto conceitual gigante descrevendo um esquema que nos permite agrupar as diversas tecnologias de rede em um todo coordenado. O principal objetivo é um sistema que esconda os detalhes de hardware subjacente de rede, enquanto fornece serviços de comunicação universal. O resultado principal é uma abstração de alto nível que fornece a estrutura para todas as decisões de projeto. Capítulos seguintes mostram como usamos a abstração aqui descrita para construir as camadas de software de comunicação de internet necessárias e como o software omite os mecanismos subjacentes de transporte físico. Capítulos posteriores mostram como os aplicativos usam o sistema de comunicação resultante.

3.2 Interconexão em nível de aplicação

Quando confrontados com sistemas heterogêneos, os primeiros criadores contaram com programas de aplicação especiais, chamados *gateways de aplicação*, para esconder as diferenças subjacentes e fornecer a aparência de uniformidade. Por exemplo, uma das primeiras incompatibilidades surgiu a partir de sistemas de e-mail comerciais. Cada fornecedor projetou

seu próprio sistema de e-mail. Os provedores escolheram um formato para armazenar e-mail, convenções para identificar um destinatário e um método para transferir mensagem de e-mail do remetente ao destinatário. Infelizmente, os sistemas eram totalmente incompatíveis.

Quando era necessária uma conexão entre os sistemas de e-mail, usava-se um gateway de aplicação. O software de gateway era executado em um computador que se conectava a ambos os sistemas de e-mail, como ilustra a Figura 3.1.

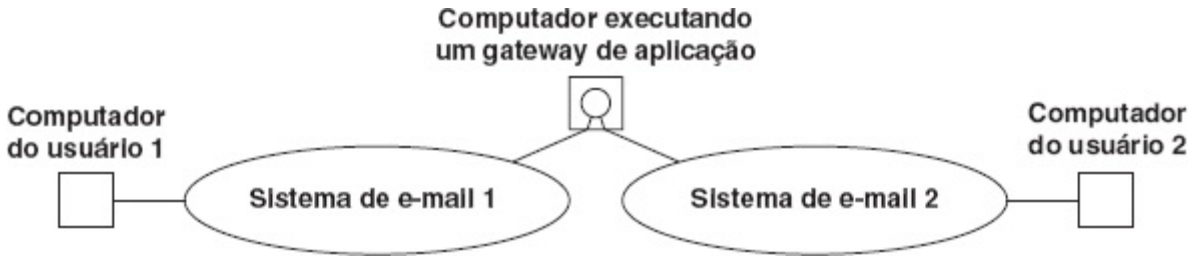


Figura 3.1 Ilustração de um gateway de aplicação usado para acomodar um par de sistemas de e-mail heterogêneos.

O gateway de aplicação deve compreender os detalhes das conexões de rede e os protocolos de mensagens, bem como o formato das mensagens usado nos dois sistemas de e-mail. Quando o usuário 1 envia uma mensagem para o usuário 2, o e-mail do usuário 1 é configurado para enviar a mensagem para o gateway de aplicação. Este deve traduzi-la para a forma utilizada pelo sistema de e-mail 2 e, em seguida, encaminhar a mensagem para o usuário 2.

Usar programas de aplicação para ocultar detalhes da rede pode parecer bastante razoável. Já que tudo pode ser manipulado por um aplicativo, não é necessário nenhum hardware especial. Além disso, os sistemas originais de e-mail nos computadores dos usuários permanecem inalterados. Na verdade, nem os usuários nem o software de e-mail nos computadores dos usuários podem dizer que o outro usuário tem um sistema de e-mail diferente.

Infelizmente, a abordagem no gateway de aplicação é complicada e limitada. A principal desvantagem surge porque um determinado gateway de aplicação só pode lidar com uma aplicação específica. Por exemplo, mesmo que um gateway de e-mail esteja ativo, ele não pode ser usado para transferir arquivos, conectar sessões de chat ou encaminhar mensagens de texto. Uma segunda desvantagem surge quando as diferenças de funcionalidade evitam interoperação. Por exemplo, se o sistema de e-mail 1 permite a um remetente anexar um arquivo a uma mensagem, mas o

sistema de e-mail 2 não, um gateway de aplicação não vai ser capaz de transferir mensagens que incluam arquivos. A terceira desvantagem surge a partir da frequência de atualizações. Sempre que um ou outro fornecedor muda seu software de e-mail, o gateway deve ser atualizado para lidar com essa mudança. Assim, gateways de aplicação devem ser atualizados frequentemente.

Nosso exemplo considera apenas um gateway de aplicação que conecta dois sistemas. Os usuários que têm experiência em rede entendem que, uma vez que o tamanho de um sistema de comunicação cresça o suficiente para se tornar global e existam vários fornecedores cada um criando seu próprio software de aplicação, será impossível manter um conjunto de gateways de aplicação que interliguem todas as redes. Além disso, para evitar a criação de gateways de aplicação para todas as combinações possíveis, o sistema evolui rapidamente a fim de usar um paradigma de comunicação passo a passo em que uma mensagem enviada para o primeiro gateway de aplicação é convertida e enviada para o segundo, e assim por diante. Uma comunicação bem-sucedida requer um correto funcionamento de todos os gateways de aplicação ao longo do caminho. Se qualquer um deles não consegue realizar a tradução correta, a mensagem não será entregue. Além disso, a fonte e o destino podem continuar incapazes de detectar ou controlar o problema. Assim, os sistemas que usam gateways de aplicação não podem garantir uma comunicação confiável.

3.3 Interconexão em nível de rede

A alternativa ao uso de gateways em nível de aplicativo é um sistema baseado na interligação em nível de rede. Ou seja, podemos conceber um sistema que transfere pacotes a partir de sua fonte original até seu destino final sem o uso de programas aplicativos intermediários. Comutação de pacotes em vez de arquivos ou mensagens grandes tem várias vantagens. Primeiro, o sistema mapeia diretamente no hardware de rede subjacente, tornando-se extremamente eficiente. Em segundo lugar, a interligação em nível de rede separa atividades de comunicação de dados de programas aplicativos, permitindo que computadores intermediários lidem com o tráfego de rede sem compreender as aplicações que estão enviando ou recebendo mensagens. Em terceiro lugar, a comunicação em nível de rede mantém a totalidade do sistema flexível, tornando possível a construção de instalações de comunicações de finalidade geral que não estão limitadas a usos específicos. Em quarto lugar, veremos que o sistema permite aos gerentes de rede adicionarem ou alterarem as tecnologias de rede, enquanto os programas de aplicação permanecem inalterados.

A chave para a concepção de interconexão universal em nível de rede pode ser encontrada na de comunicação abstrata, conceito conhecido como *interligação de rede* em um sistema. O conceito de *internet* é extremamente poderoso. Ela separa as noções de comunicação dos detalhes das tecnologias de rede e oculta de usuários e aplicações detalhes de baixo nível. Mais importante, dirige todas as decisões de projeto de software e explica como lidar com endereços físicos e rotas. Depois de analisar as motivações básicas para interligações de redes, vamos considerar de forma mais detalhada as propriedades de uma internet.

Começamos com duas observações fundamentais sobre a concepção de sistemas de comunicação:

- nenhuma tecnologia de hardware de rede única pode satisfazer todas as restrições;
- usuários desejam interconexão universal.

A primeira observação é de ordem tanto econômica quanto técnica. Tecnologias de LAN de baixo custo que proporcionam comunicação de alta velocidade só cobrem curtas distâncias; redes de que se estendem por longas distâncias não podem fornecer comunicação local barata. É possível conseguir apenas dois itens no grupo alta velocidade, longa distância e baixo custo, mas não é possível atingir todos os três. Então, como não há tecnologia de rede única capaz de satisfazer todas as necessidades, somos forçados a considerar várias tecnologias de hardware subjacentes.

A segunda observação é autoexplicativa. Um usuário arbitrário gostaria de ser capaz de comunicar-se com um terminal arbitrário, outro usuário ou um sistema de computador. Dado o desejo de acesso móvel, podemos dizer que um usuário arbitrário gostaria de se envolver em uma comunicação a partir de uma localização arbitrária. Como consequência, desejamos um sistema de comunicação que não seja limitado pelas fronteiras de redes físicas.

O objetivo é construir um sistema unificado e cooperativo de interligação de redes que suporte um serviço de comunicação universal. Cada computador será anexado a uma rede específica, como as descritas no Capítulo 2, e vai usar os meios de comunicação dependentes de tecnologia de rede subjacente. Um novo software, inserido entre os mecanismos de comunicação dependentes de tecnologia e programas de aplicação, irá esconder todos os detalhes de baixo nível e fazer o conjunto de redes parecerem uma única grande rede. Tal esquema de interconexão é chamado *interligação de redes* ou *internet*.

A ideia de construir uma internet segue um modelo-padrão de projeto de sistema: pesquisadores imaginaram uma instalação de alto nível e trabalharam a partir de tecnologias de base disponíveis para realizar o objetivo imaginado. Na maioria dos casos, os pesquisadores criam um software que oferece cada um dos mecanismos necessários. Eles continuam até que produzam um sistema de trabalho que implemente de forma eficiente o sistema previsto. A próxima seção mostra o primeiro passo do processo de concepção, definindo a meta de forma mais precisa. As seções seguintes explicam a abordagem e os capítulos seguintes explicam princípios e detalhes.

3.4 Propriedades da Internet

A noção de serviço universal é importante, mas apenas isso não abrange todas as ideias que temos em mente para uma internet unificada. Na verdade, pode haver muitas implementações de serviços universais. Um dos primeiros princípios em nosso projeto objetiva o encapsulamento: queremos ocultar do usuário a arquitetura de rede subjacente e permitir a comunicação sem que seja necessário ter conhecimento da estrutura da internet. Ou seja, não queremos exigir que os usuários ou programas de aplicação entendam os detalhes das redes subjacentes ou interconexões de hardware para usar a internet. Não queremos também exigir uma topologia de interconexão de rede. Em particular, a inclusão de uma nova rede à internet não deverá significar a conexão a um ponto de comutação centralizado, e nem deverá significar a inclusão de conexões físicas diretas entre o novo hardware e todas as redes existentes. Queremos ser capazes de enviar dados por redes intermediárias, embora elas não estejam conectadas diretamente aos computadores de origem ou destino. Queremos que todos os computadores na rede compartilhem um conjunto universal de identificadores de máquina (que podem ser imaginados como *nomes* ou *endereços*).

Nossa noção de uma rede unificada também inclui a ideia de independência de rede e computador. Ou seja, queremos que o conjunto de operações usadas para estabelecer a comunicação ou transferir dados permaneça independente das tecnologias de rede subjacentes e do computador de destino. Um usuário não deve ter que conhecer sobre redes ou computadores remotos quando utilizar uma aplicação, e um programador não precisará entender de topologia de interconexão de rede ou do tipo de um computador remoto ao criar ou usar programas aplicativos que se comunicam pela internet.

3.5 Arquitetura da Internet

Temos visto como os computadores se conectam a redes individuais. Surge a pergunta: como as redes são interligadas para formar interligação de redes? A resposta tem duas partes. Fisicamente, duas redes não podem ser conectadas em conjunto diretamente. Em vez disso, elas só podem ser ligadas por um sistema de computador que tem o hardware necessário para se conectar a cada uma das redes. A ligação física não fornece a interligação que temos em mente, no entanto, porque essa ligação não garante que um computador vá cooperar com outras máquinas que desejem se comunicar. Para se ter uma internet viável, precisamos de computadores especiais que estão dispostos a transferir pacotes de uma rede para outra. Os computadores que interligam duas redes e passam pacotes de uma para outra são chamados de *roteadores de internet* ou *roteadores IP*.*

Para entender a interconexão, considere um exemplo que consiste de duas redes físicas e um roteador, como mostra a Figura 3.2.

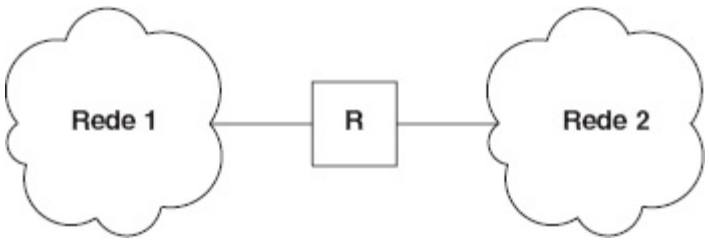


Figura 3.2 Duas redes físicas interconectadas por um roteador IP, *R*.

Na figura, o roteador *R* conecta-se a ambas as redes, rede *1* e rede *2*. Para *R* atuar como um roteador, deve capturar pacotes na rede *1* que são direcionados para a rede *2* e transferi-los. De forma análoga, *R* deve capturar pacotes na rede *2* que são destinados para máquinas na rede *1* e transferi-los.

Na figura, são usadas nuvens para denotar redes físicas, pois o hardware exato não é importante. Cada rede pode ser uma LAN ou uma WAN, e cada uma pode ter muitos ou poucos computadores conectados. O uso de nuvens enfatiza uma diferença significativa entre roteadores e bridges – uma bridge só pode conectar duas redes que usem a mesma tecnologia, mas um roteador pode conectar redes arbitrariamente.

3.6 Interconexão de múltiplas redes com roteadores IP

Embora a Figura 3.2 ilustre a estratégia básica de conexão, ela é extremamente simplista. Uma internet realista incluirá múltiplas redes e roteadores. Nesse caso, cada roteador precisa conhecer as redes além daquelas às quais ele se conecta diretamente. Por exemplo, considere a Figura 3.3, que mostra três redes interconectadas por dois roteadores.

Figura 3.3 Três redes interconectadas por dois roteadores.

No exemplo, o roteador R_1 deve transferir da rede *1* para a rede *2* todos os pacotes destinados a computadores tanto da rede *2* como da rede *3*. De forma similar, o roteador R_2 deve transferir pacotes da rede *3* que são destinados tanto para a rede *2* como para a rede *1*. O importante é que um roteador deve lidar com pacotes para as redes às quais ele não está conectado. Em uma internet grande composta por muitas redes, a tarefa do roteador de decidir sobre para onde enviar os pacotes se torna mais complexa.

A ideia de um roteador parece simples, mas ele é importante, pois fornece uma maneira de interconectar redes, e não só computadores. Na verdade, já descobrimos o principal sobre interconexão usada através de uma internet como descrito a seguir:

> *Em uma internet TCP/IP, um sistema especial de computadores chamados roteadores* **IP** *fornece interconexões entre redes físicas.*

Você pode suspeitar que os roteadores devem cada um saber como encaminhar os pacotes para o seu destino, pois são grandes máquinas com memória primária ou secundária suficiente para armazenar informações sobre cada computador na internet à qual eles estão conectados. Na verdade, os roteadores usados com internet TCP/IP podem ser computadores modestos semelhantes a um desktop PC. Eles não precisam

especialmente de grande armazenamento em disco, nem de uma grande capacidade de memória principal. O truque que permite que os roteadores tenham tamanho razoável reside no conceito a seguir.

Roteadores usam a rede de destino, não o computador de destino, quando estão enviando um pacote.

Como o encaminhamento de pacotes é baseado em redes, a quantidade de informações que um roteador precisa manter é proporcional ao número de redes na internet, não ao número de computadores. Como vimos no Capítulo 1, existem duas vezes menos redes na Internet do que computadores.

Em virtude de eles terem um papel fundamental na comunicação na internet, vamos voltar aos roteadores em capítulos posteriores para discutir os detalhes de como eles funcionam e de como aprendem sobre destinos remotos. Por enquanto, suponhamos que é possível e prático para cada roteador ter rotas corretas para todas as redes em uma internet. Iremos também assumir que apenas roteadores fornecem conexões entre redes físicas em uma internet.

3.7 A visão do usuário

Lembre-se de que uma internet é projetada para fornecer uma interconexão universal entre computadores, independentemente das redes específicas a que se ligam. Queremos que um usuário visualize uma internet como uma única rede virtual à qual todas as máquinas conectam-se, apesar de suas conexões físicas. A Figura 3.4 ilustra a ideia.

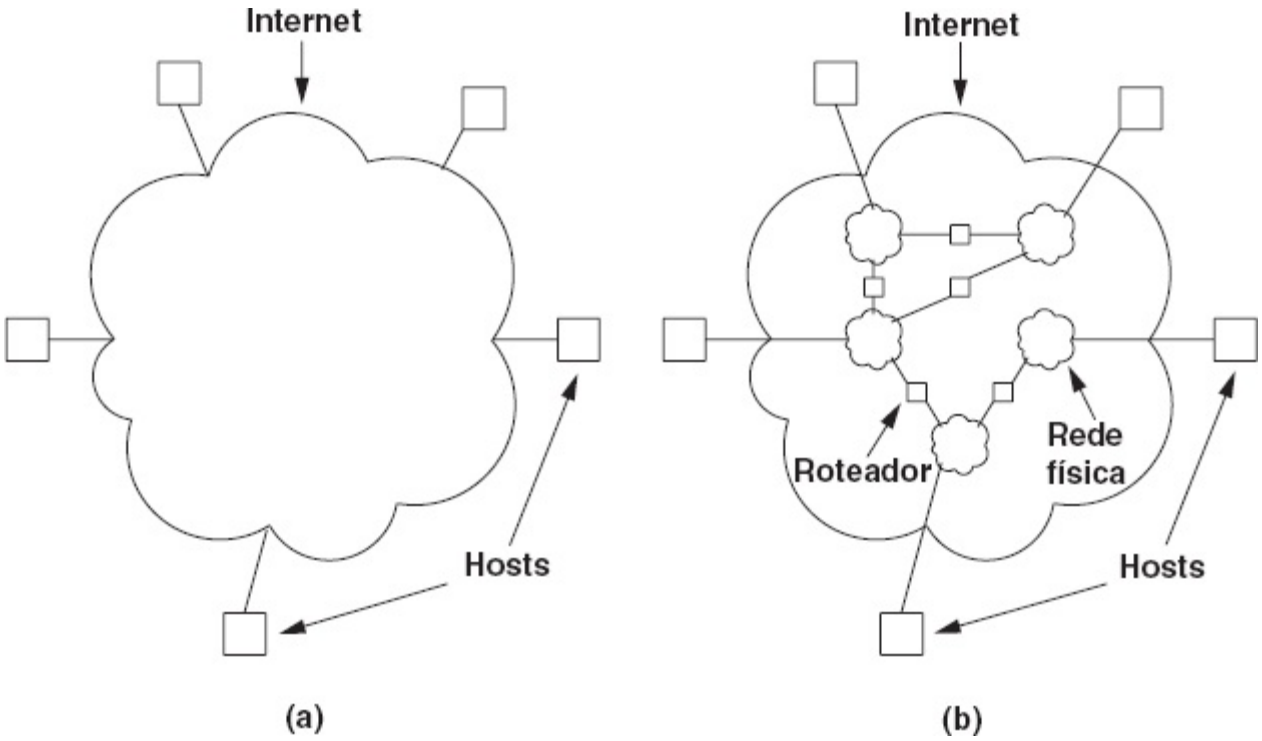


Figura 3.4 (a) A visão do usuário de uma internet TCP/IP em que cada computador parece se conectar a uma única rede grande e (b) a estrutura das redes físicas e dos roteadores que fornecem interconexão.

Na figura, a parte (a) mostra a visão que o usuário tem. Ele pensa na internet como um sistema de comunicação unificado. A visão do usuário simplifica os detalhes e torna mais fácil conceituar comunicação. A parte (b) ilustra as redes constituintes e sua interligação com roteadores. Claro que cada computador que se conecta a uma internet deve executar um software que impõe o ponto de vista de uma única rede física. O software deve ocultar detalhes e permitir a programas aplicativos enviarem e receberem pacotes em locais arbitrários, como se o computador fosse conectado a uma única rede.

A vantagem de fornecer interligação em nível de rede torna-se agora clara. Como os programas aplicativos que se comunicam através da internet não sabem os detalhes de conexões subjacentes, eles podem ser executados sem alterações em qualquer computador. Devido aos detalhes de conexões de rede física de cada máquina estarem escondidos no software de internet, apenas ele precisa reagir quando novas conexões físicas são adicionadas ou conexões existentes são removidas. Por exemplo, um dispositivo portátil pode se conectar a uma rede Wi-Fi em um aeroporto, ser desligado para o voo e, em seguida, ser conectado a uma rede Wi-Fi em outro aeroporto sem

que as aplicações sejam afetadas de maneira alguma. Mais importante, é possível mudar a estrutura interna da internet (por exemplo, adicionando uma rede ou um roteador) enquanto os programas aplicativos estão sendo executados.

A segunda vantagem de ter a comunicação em nível de rede é mais sutil: lembre-se de que os usuários não têm de entender ou especificar como as redes se conectam, o tráfego que carregam ou que aplicações elas suportam. Na verdade, as redes internas não conhecem as aplicações – elas apenas transportam pacotes. Como resultado, os programadores que não conhecem estrutura de internet podem criar aplicativos, e a internet não precisa ser modificada quando uma nova aplicação é criada. Assim sendo, os gerentes de rede são livres para mudar partes interiores da arquitetura subjacente da internet sem qualquer efeito no software aplicativo. Claro que, se um computador se mover para um novo tipo de rede, vai precisar de um novo cartão de interface de rede e do software do driver associado, mas isso é simplesmente uma atualização das capacidades do computador, em vez de uma mudança devido à internet.

A Figura 3.4b ilustra um ponto sobre topologia da internet: os roteadores não fazem conexões diretas entre todos os pares de redes em uma internet. Pode ser necessário que o tráfego que viaja de um computador para outro passe através de vários roteadores quando atravessa redes intermédias. Assim, as redes que participam em uma internet são análogas a um sistema de estradas. Redes locais alimentam o tráfego em redes maiores, assim como as estradas locais ligam-se a rodovias. A maior parte dos ISPs fornece redes que lidam com o trânsito do tráfego, assim como o sistema interestadual nos Estados Unidos forma um backbone de rodovias que lidam com o tráfego por longas distâncias.

3.8 Todas as redes são iguais

O Capítulo 2 analisou exemplos de hardware de rede usados para construir internets TCP/IP e ilustrou a grande diversidade de tecnologias. Nós descrevemos uma internet como uma coleção de redes cooperativas interconectadas. Agora é importante entender um conceito fundamental: do ponto de vista da internet, qualquer sistema de comunicação capaz de transferir pacotes é considerado como uma única rede, independentemente do seu atraso e características de rendimento, tamanho máximo do pacote, ou escala geográfica. Em particular, a Figura 3.4b usa a mesma forma de pequena nuvem para representar cada rede física porque o TCP/IP os trata igualmente, apesar de suas diferenças. O ponto importante é descrito a seguir.

O protocolo TCP/IP de uma internet trata todas as redes de forma igual. Uma rede de área local como uma Ethernet, uma rede de longa distância usada como backbone, uma rede wireless tal como um hotspot Wi-Fi e um link ponto a ponto entre dois computadores, cada uma, conta como uma rede.

Os leitores não acostumados à arquitetura internet podem achar difícil aceitar essa visão simplista de redes. Em essência, o TCP/IP define uma abstração de "rede" que esconde os detalhes de redes físicas. Na prática, os arquitetos de rede devem escolher uma tecnologia apropriada para cada uso. Vamos aprender, entretanto, que a abstração dos detalhes ajuda a fazer os protocolos TCP/IP extremamente flexíveis e poderosos.

3.9 As questões não respondidas

Nosso esboço da internet deixa muitas perguntas sem resposta. Por exemplo, você pode se perguntar sobre a forma exata de endereços de internet atribuídos a computadores, ou como esses endereços se relacionam com os endereços de hardware (por exemplo, endereços Ethernet MAC de 48 bits) descritos no Capítulo 2. Os Capítulos 5 e 6 enfrentam a questão de endereçamento. Eles descrevem o formato dos endereços IP e ilustram como o software em um computador mapeia endereços de internet e endereços físicos. Você também pode querer saber exatamente o que parece um pacote quando ele viaja através de uma internet, ou o que acontece quando os pacotes chegam rápido demais para um computador ou roteador manusear. O Capítulo 7 responde a essas perguntas. Finalmente, você pode se perguntar como vários programas aplicativos que são executados simultaneamente em um único computador podem enviar e receber pacotes para vários destinos sem emaranharem-se nas transmissões um do outro, ou como roteadores de internet aprendem sobre rotas. Todas essas perguntas serão respondidas.

Embora possa parecer vago agora, a direção que estamos seguindo vai nos fazer saber mais sobre a estrutura e o uso de software de protocolo da internet. Vamos examinar cada parte, olhando para os conceitos e princípios, bem como detalhes técnicos. Começamos por descrever a camada de comunicação física na qual uma internet é construída. Cada um dos capítulos seguintes vai explorar uma parte do software da internet até

que entendamos como todas as peças se encaixam.

3.10 Resumo

Uma internet é mais do que uma coleção de redes interconectadas por computadores. Uma interligação de redes implica que os sistemas interligados estejam de acordo com as convenções que permitem que cada host se comunique com qualquer outro host. Em particular, uma internet vai permitir que dois computadores host se comuniquem, mesmo que o caminho de comunicação entre eles passe através de uma rede à qual nenhum dos dois se conecta diretamente. Essa cooperação só é possível quando os computadores host concordam com um conjunto de identificadores universais e um conjunto de procedimentos para a movimentação de dados para o seu destino final.

Em uma internet, interconexões entre as redes são formadas por sistemas de computadores para fins especiais chamados de roteadores IP que se conectam a duas ou mais redes. Um roteador encaminha pacotes entre as redes recebendo-os de uma rede e os enviando para outra.

EXERCÍCIOS

3.1 Representantes comerciais vendem roteadores sem fio para uso em casa. Leia sobre esses roteadores. Que processadores são usados? Com quantos bits por segundo esse roteador deve lidar?

3.2 Aproximadamente quantas redes constituem a parte da Internet em seu site? Aproximadamente quantos roteadores?

3.3 Informe-se sobre o maior roteador usado na sua empresa ou organização. Quantas conexões de rede ele tem?

3.4 Considere a estrutura interna do exemplo internet mostrado na Figura 3.4b. Quais roteadores são mais críticos para o correto funcionamento da internet? Por quê?

3.5 Alterar as informações em um roteador pode ser complicado, porque é impossível mudar todos os roteadores simultaneamente. Examine algoritmos que garantam a instalação de uma mudança em um conjunto de computadores ou em nenhum.

3.6 Em uma internet, roteadores trocam periodicamente as informações de suas tabelas de roteamento, tornando possível para um novo roteador aparecer e começar a rotear pacotes. Estude os algoritmos usados para trocar informações de roteamento.

3.7 Compare a organização de uma internet TCP/IP com o estilo XNS de

internet projetado pela Xerox Corporation.

---

* A literatura original usou o termo *IP gateway*. Entretanto, provedores adotaram o termo *roteador IP*.

CAPÍTULO 4

Camadas de protocolos

CONTEÚDOS DO CAPÍTULO

4.1 Introdução

O capítulo anterior analisa os fundamentos arquitetônicos de internetworking e descreve a interligação de redes com roteadores. Este capítulo considera a estrutura do software encontrado em hosts e roteadores que realizam a comunicação de rede. Apresenta o princípio geral de camadas, mostra como as camadas tornam o software de protocolo mais fácil de entender e construir e traça o caminho que os pacotes seguem através do software de protocolo quando eles atravessam uma internet TCP/IP. Capítulos sucessivos complementam detalhes explicando protocolos em cada camada.

4.2 A necessidade de múltiplos protocolos

Dissemos que os protocolos permitem especificar ou compreender a comunicação sem conhecer os detalhes de hardware de rede de um determinado fornecedor. Eles são para a comunicação por computador o que linguagens de programação são para a computação. A analogia se encaixa bem. Como na linguagem assembly, alguns protocolos descrevem a comunicação através de uma rede física. Por exemplo, os detalhes do formato de frame Ethernet, o significado de campos de cabeçalho, a ordem em que os bits são transmitidos em cabos, e a maneira como são tratados erros de CRC constituem um protocolo que descreve a comunicação em uma rede Ethernet. Veremos que o protocolo de Internet é como uma

linguagem de alto nível que lida com abstrações, incluindo endereços de Internet, formato dos pacotes de Internet, e com a forma como os roteadores encaminham pacotes. Nem protocolos de baixo nível nem de alto nível são suficientes por si só; ambos devem estar presentes.

A comunicação em rede é um problema complexo sob muitos aspectos. Para entender a complexidade, pense em alguns dos problemas que podem surgir quando os computadores se comunicam através de uma rede de dados como os citados a seguir.

- *Falha de hardware.* Um computador ou roteador pode falhar porque o hardware falha ou porque o sistema operacional falha. Um link de transmissão de rede pode falhar ou acidentalmente desligar. O Software de Protocolo precisa detectar tais falhas e, se possível, se recuperar delas.
- *Congestionamento de rede.* Mesmo quando todos os hardwares e softwares funcionam corretamente, as redes têm capacidade finita que pode ser excedida. O Software de Protocolo precisa encontrar uma maneira de detectar o congestionamento e evitar mais tráfego para que a situação não piore.
- *Atraso de pacote ou perda de pacote.* Às vezes, pacotes sofrem atrasos extremamente longos ou são perdidos. O Software de Protocolo precisa identificar as falhas ou adaptarse a longos atrasos.
- *Dados corrompidos.* Interferências elétricas ou magnéticas ou falhas de hardware podem causar erros de transmissão que corrompem o conteúdo dos dados transmitidos; a interferência pode ser especialmente grave em redes sem fio. O Software de Protocolo precisa detectar e se recuperar de tais erros.
- *Duplicação de dados ou chegadas invertidas.* As redes que oferecem várias rotas podem entregar pacotes fora de sequência ou entregar duplicatas de pacotes. O Software de Protocolo precisa reordenar pacotes e remover qualquer duplicata.

Tomados em conjunto, os problemas parecem esmagadores. É impossível escrever uma única especificação de protocolo que irá lidar com todos eles. A partir da analogia com as linguagens de programação, podemos ver como superar a complexidade. A tradução de programa foi dividida em quatro subproblemas conceituais identificados com o software que controla cada subproblema: compilador, assembler, link editor e

carregador. A divisão torna possível para o projetista se concentrar em um subproblema de cada vez, e para o implementador construir e testar cada pedaço de software de forma independente. Veremos que o software de protocolo é dividido de forma semelhante.

Duas observações finais da nossa analogia de linguagem de programação vão ajudar a esclarecer a organização dos protocolos. Em primeiro lugar, deve ficar claro que as peças de software de tradução devem concordar com o formato exato de dados que passam entre elas. Por exemplo, os dados transmitidos a partir de um compilador para um assembler consistem de um programa definido pela linguagem de programação assembly. O processo de tradução envolve múltiplas representações. A analogia continua para o software de comunicação porque vários protocolos definem as representações de dados passados entre os módulos de software de comunicação. Em segundo lugar, as quatro partes do tradutor formam uma sequência linear, em que a saída do compilador torna-se a entrada para o assembler, e assim por diante. O Software de Protocolo também usa uma sequência linear.

4.3 As camadas conceituais de software de protocolo

Pensamos nos módulos de protocolo em cada computador como sendo verticalmente empilhados em *camadas*, como na Figura 4.1. Cada camada tem responsabilidade por tratar de uma parte do problema.

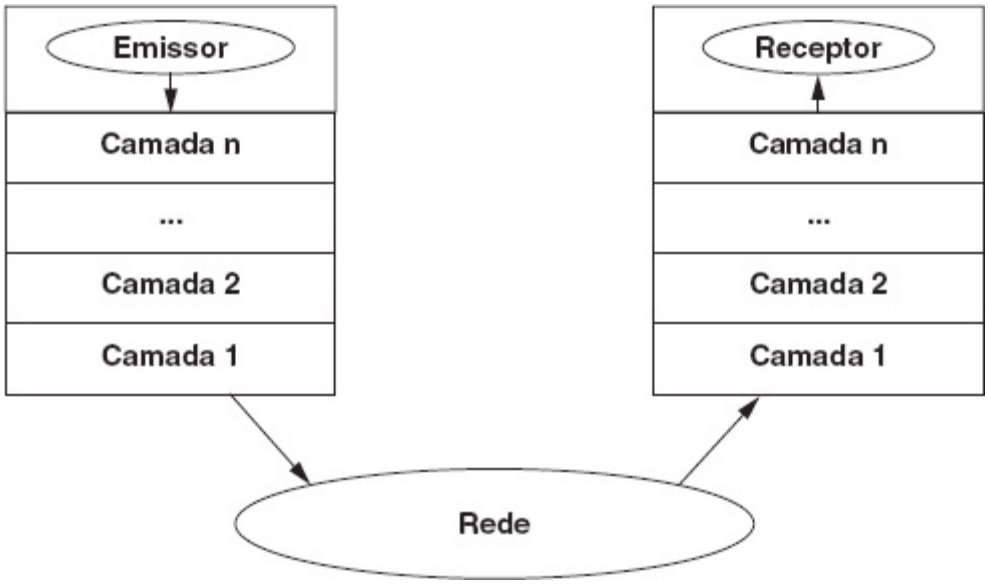


Figura 4.1 A organização conceitual do protocolo em camadas.

Conceitualmente, o envio de uma mensagem de um aplicativo em um computador para um aplicativo em outro, significa transferir a mensagem através de camadas sucessivas de software de protocolo na máquina do remetente, encaminhando a mensagem através da rede e transferindo-a através de camadas sucessivas de software de protocolo na máquina do receptor.

4.4 Funcionalidade das camadas

Uma vez tomada a decisão de dividir o problema de comunicação e organizar o software de protocolo em módulos de forma que cada uma lide com um subproblema, surgem duas questões inter-relacionadas: quantas camadas devem ser criadas e qual funcionalidade deve residir em cada camada? As perguntas não são fáceis de responder por vários motivos. Em primeiro lugar, dado um conjunto de objetivos e restrições que regem um problema de comunicação particular, é possível escolher uma organização que irá otimizar o software de protocolo para esse problema. Em segundo lugar, mesmo quando se consideram serviços gerais em nível de rede, como o transporte confiável, é possível escolher entre abordagens fundamentalmente distintas para resolver o problema. Em terceiro lugar, o projeto de arquitetura da rede (ou internet) e a organização do software de protocolo estão inter-relacionados; um não pode ser concebido sem o outro. Duas abordagens para camadas de protocolo dominam o campo. As duas próximas seções tratarão dessas abordagens.

4.5 Modelo de referência da camada ISO 7

O primeiro modelo de camadas foi baseado em trabalho inicial feito pela Organização Internacional de Normalização (ISO), e é conhecido como *Modelo de Referência de Interconexão Open System*. Ele é muitas vezes referido como o *modelo ISO*. Infelizmente, antecede a trabalhos na Internet e não descreve bem os seus protocolos. Ele contém camadas não utilizadas por protocolos TCP/IP. Além disso, em vez de uma camada dedicada à "internet", o modelo ISO foi projetado para uma única rede e tem uma camada de "rede". Apesar de suas limitações, as divisões de marketing e vendas de fornecedores comerciais ainda se referem ao modelo ISO e introduzem uma confusão ainda maior, afirmando que, de alguma forma, tem sido utilizado no design de seus produtos na internet. O modelo ISO contém 7 camadas conceituais organizadas como mostra a Figura 4.2.

Figura 4.2 O modelo de referência ISO 7. Devido a ele ter sido projetado para descrever protocolos em uma única rede, o modelo não descreve bem a organização dos protocolos TCP/IP.

4.6 X.25 e sua relação com o modelo ISO

Embora tenha sido projetado para fornecer um modelo conceitual e não um guia de implementação, o esquema de camadas ISO tem sido a base para várias implementações de protocolos. Entre os protocolos normalmente associados ao modelo ISO, o conjunto de protocolos conhecido como X.25 provavelmente é o mais reconhecido e o mais usado. O X.25 foi estabelecido como uma recomendação da *International Telecommunications Union* (*ITU*),* uma organização que recomenda padrões para os serviços telefônicos internacionais. O X.25 foi adotado por redes de dados públicas e tornou-se especialmente popular na Europa. Considerar o X.25 ajudará a entender as camadas ISO.

Na visão do X.25, uma rede opera de modo muito semelhante a um sistema telefônico. Considera-se que uma rede X.25 consiste em comutadores de pacotes que contêm a inteligência necessária para roteálos. Os computadores não se conectam diretamente aos fios de comunicação da rede. Em vez disso, cada host se conecta a um dos comutadores de pacotes usando uma linha de comunicação serial. De certa forma, a conexão entre

um host e um comutador de pacote X.25 é uma rede em miniatura consistindo de um link serial. O host precisa seguir um procedimento complicado para transferir pacotes para a rede. O padrão de camadas de protocolo especifica vários aspectos de rede, como se pode ver a seguir.

- *Camada física.* X.25 especifica um padrão para a interconexão física entre computadores e comutadores de pacotes de rede. No modelo de referência, a camada *1* especifica a interconexão física, incluindo características elétricas de voltagem e corrente.
- *Link de camada de dados.* A parte da camada *2* do protocolo X.25 especifica como os dados trafegam entre um computador e o comutador de pacote a que se conecta. O X.25 usa o termo *frame* para se referir a uma unidade de dados que é transferida a um comutador de pacotes. Como o hardware subjacente entrega apenas um fluxo de bits, o protocolo da camada *2* precisa definir o formato dos frames e especificar como as duas máquinas reconhecem os limites do frame. Como os erros de transmissão podem destruir os dados, o protocolo da camada *2* inclui detecção de erros (por exemplo, um contador de frames), assim como um mecanismo de limite de tempo que faz com que o computador reenvie um frame até que ele tenha sido transmitido com sucesso. É importante lembrar que a transferência bem-sucedida na camada *2* significa que um frame foi passado para o comutador de pacote da rede; isso não significa que o comutador de pacote seja capaz de encaminhá-lo ou entregá-lo.
- *Camada de rede.* O modelo de referência ISO especifica que a terceira camada contém funcionalidade que completa a definição da interação entre host e rede. Denominada *camada de rede* ou de *comunicação sub-rede*, essa camada define a unidade básica de transferência pela rede e inclui os conceitos do endereçamento e encaminhamento ao destino. Devido às camadas *2* e *3* serem conceitualmente independentes, o tamanho dos pacotes da camada *3* pode ser maior que os frames da camada *2* (por exemplo, um computador pode criar um pacote da camada *3* e então a camada *2* divide esse pacote em pedaços menores para transferi-los ao comutador de pacotes).

- *Camada de transporte.* A camada 4 fornece confiabilidade fim a fim, fazendo com que o computador de destino se comunique com o computador de origem. A ideia é que, embora as camadas de protocolos inferiores forneçam verificações confiáveis em cada transferência, essa camada faz uma verificação extra, para certificar-se de que nenhuma máquina intermediária falhou.
- *Camada de sessão.* As camadas superiores do modelo ISO descrevem como o protocolo pode ser organizado para lidar com toda a funcionalidade necessária aos programas aplicativos. Quando o modelo ISO foi formado, as redes eram usadas para serem ligadas a um terminal (ou seja, uma tela e um teclado), em um computador remoto. Na verdade, o serviço oferecido inicialmente por redes públicas de dados concentrou-se no fornecimento de acesso a terminal. A camada 5 lida com os detalhes.
- *Camada de apresentação.* A camada ISO 6 destina-se a padronizar o formato de dados de programas de aplicação enviados através de uma rede. Uma das desvantagens de padronizar os formatos de dados é que isso impede a inovação – novas aplicações não podem ser implantadas enquanto o seu formato de dados não for padronizado. Outra desvantagem surge porque determinados grupos reivindicam o direito de padronizar representações adequadas para o seu domínio de aplicação (por exemplo, os formatos de dados para vídeo digital são especificados por grupos que lidam com padrões de vídeo em vez de grupos que padronizam redes). Consequentemente, as normas de apresentação são geralmente ignoradas.
- *Camada de aplicação.* A camada ISO 7 inclui programas de aplicação que usam a rede. Exemplos incluem correio eletrônico e programas de transporte de arquivos.

4.7 Modelo de referência do TCP/IP cinco camadas

O segundo modelo de estratificação principal não surgiu a partir de um órgão formal de normas. Em vez disso, o modelo surgiu de pesquisadores que projetaram a Internet e do conjunto de protocolos TCP/IP. Quando os protocolos TCP/IP se tornaram populares, os defensores do modelo ISO mais antigo tentaram estendê-lo para acomodar o TCP/IP. No entanto, permanece o fato de que o modelo ISO original não fornece uma camada internet, mas camadas de sessão e de apresentação que não são pertinentes

a protocolos TCP/IP.

Uma das principais diferenças conceituais entre os modelos de camadas ISO e Internet surge da maneira como eles foram definidos. O modelo ISO é *normativo* – organismos de normatização convocaram um comitê que escreveu as especificações de como os protocolos deveriam ser construídos. Eles, então, começaram a implementar protocolos. O ponto importante é que o modelo é anterior à implementação. Por outro lado, o modelo de Internet é *descritivo* – os pesquisadores passaram anos entendendo como estruturar os protocolos, construindo protótipos de implementações e documentando os resultados. Depois que os pesquisadores estavam finalmente convencidos de que tinham entendido o projeto, um modelo foi construído. Isso pode ser resumido da forma a seguir.

Diferentemente do modelo ISO, que foi definido por comitês anteriormente à implantação dos protocolos, o modelo de referência da Internet de 5 camadas foi formalizado depois que protocolos tinham sido desenvolvidos e testados.

Protocolos TCP/IP são organizados em cinco camadas conceituais – quatro camadas definem o processamento de pacotes e uma quinta camada define o hardware de rede convencional. A Figura 4.3 mostra as camadas conceituais e enumera o formato de dados que passa entre cada par sucessivo de camadas.

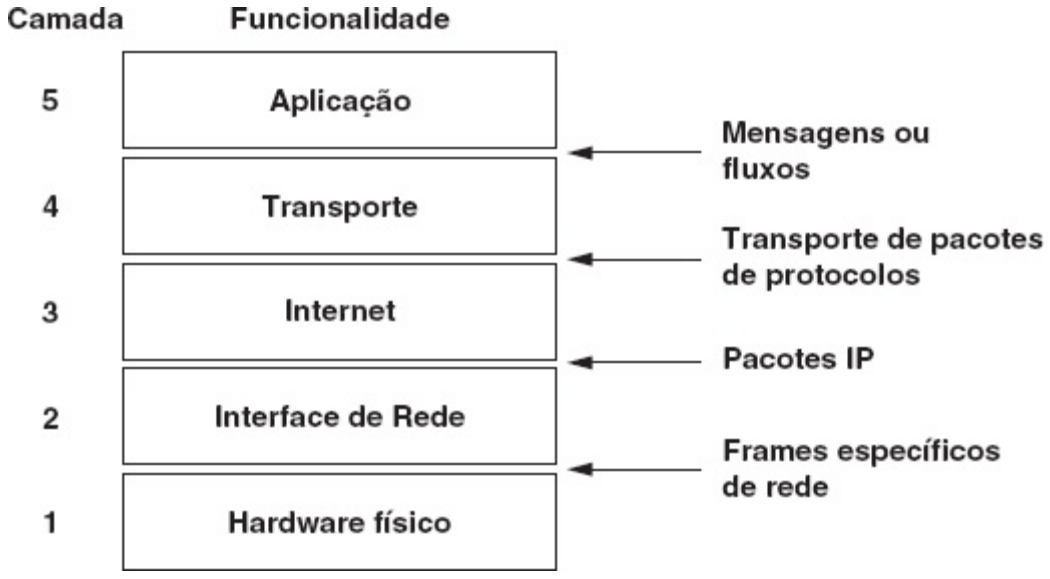


Figura 4.3 O modelo de referência de TCP/IP de cinco camadas mostrando o tipo dos objetos que passam entre camadas.

Os itens a seguir descrevem o propósito geral de cada camada. Capítulos ulteriores darão muitos detalhes e analisarão protocolos específicos em cada camada.

- *Camada de aplicação.* Na camada mais alta, os usuários invocam programas aplicativos que acessam serviços disponíveis através de uma internet TCP/IP. Uma aplicação interage com uma das camadas de protocolos de transporte para enviar ou receber dados. Cada programa de aplicação escolhe o modelo de transporte necessário, que pode ser uma sequência de mensagens individuais ou um fluxo contínuo de bytes. O programa de aplicação passa os dados no formato necessário para a camada de transporte para que seja feita a entrega.
- *Camada de transporte.* O principal dever da camada de transporte é fornecer comunicação de um programa de uma aplicação para outro. Essa comunicação é chamada de *fim a fim*, porque envolve aplicações em dois terminais em vez de aplicações entre roteadores intermediários. Uma camada de transporte pode não só regular o fluxo de informações, mas também fornecer transporte confiável, garantindo que os dados cheguem sem erros e em sequência. Para isso, o software de protocolo de transporte se organiza para ter uma confirmação de recebimento do lado receptor e a retransmissão de pacotes perdidos do lado emissor. O software de transporte divide o fluxo de dados a serem transmitidos em pequenos pedaços (por vezes chamados *pacotes*) e passa cada pacote com um endereço de destino para a próxima camada de transmissão.

Conforme descrito a seguir, um computador de uso geral pode ter vários aplicativos acessando uma internet ao mesmo tempo. A camada de transporte deve aceitar dados de vários aplicativos e enviá-lo para a próxima camada inferior. Para isso, ele acrescenta informação adicional para cada pacote, incluindo valores que identificam qual programa de aplicação enviou os dados e que a aplicação na extremidade de recepção deve receber os dados. Protocolos de transporte também usam um contador para proteção contra erros que causam alteração dos bits. A máquina de recebimento usa a contagem de verificação para identificar se o pacote chegou intacto, e usa as informações de destino para identificar o programa

de aplicação a que deve ser entregue.

- *Camada de internet.* A camada de internet lida com a comunicação de um computador para outro. Ela aceita um pedido para enviar um pacote da camada de transporte, juntamente com uma identificação do computador para o qual o pacote deve ser enviado. O software de Internet encapsula o pacote de transporte em um pacote IP, preenche o cabeçalho e envia o pacote IP tanto diretamente para o destino (se o destino está na rede local) como para um roteador para ser encaminhado pela internet (se o destino é remoto). O software da camada de Internet também lida com pacotes IP recebidos, verificando a sua validade e usando o algoritmo de encaminhamento para decidir se o pacote deve ser processado localmente ou encaminhado. Para os pacotes destinados à máquina local, o software na camada de internet escolhe o protocolo de transporte que irá lidar com o pacote.
- *Camada de interface de rede.* A menor camada de software TCP/IP compreende uma camada de interface de rede, responsável por aceitar pacotes IP e transmiti-los através de uma rede específica. Uma interface de rede pode ser constituída por um driver de dispositivo (por exemplo, quando a rede é uma rede de área local à qual o computador se conecta) ou por um subsistema complexo que implementa um protocolo de link de dados. Alguns profissionais de rede não fazem distinção entre os dois tipos; eles simplesmente usam o termo *camada MAC* ou *camada de link de dados.*

Na prática, o software de protocolo de Internet TCP/IP é muito mais complexo do que o modelo simples da Figura 4.3. Cada camada toma decisões sobre a correção da mensagem e escolhe uma ação apropriada com base no tipo de mensagem ou endereço de destino. Por exemplo, a camada de internet na máquina receptora deve decidir se a mensagem realmente chegou ao destino correto. A camada de transporte deve decidir qual programa aplicativo deve receber a mensagem.

A principal diferença entre o modelo simplista de camadas ilustrado na Figura 4.3 e o software de protocolo em um sistema real é que um computador ou um roteador pode ter várias interfaces de rede e vários protocolos podem ocorrer em cada camada. Para entender algumas das complexidades, considere a Figura 4.4, que mostra uma comparação entre camadas e módulos de softwares.

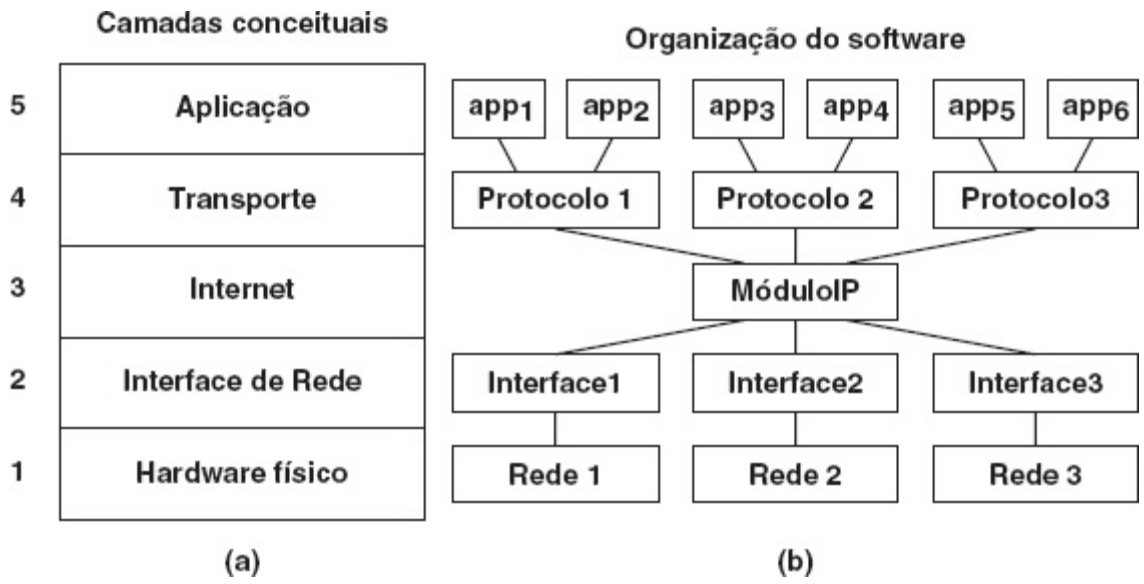


Figura 4.4 Uma comparação entre (a) camada de protocolo conceitual e (b) uma visão mais realista do software de protocolo com múltiplas interfaces de rede e múltiplos protocolos.

O diagrama conceitual na Figura 4.4 (a) mostra cinco camadas de protocolos com uma única caixa que descreve cada camada. A ilustração mais realista do software na Figura 4.4 (b) mostra que a camada 3 pode ser de fato uma camada com um protocolo (Camada 3). No entanto, pode haver múltiplas aplicações na Camada 5, e mais de um aplicativo pode usar um determinado protocolo de transporte. Vamos aprender que os protocolos da Internet têm vários protocolos de transporte na Camada 4, várias redes físicas na Camada 1 e vários módulos de interface de rede na Camada 2.

Profissionais que trabalham com redes usam os termos *ampulheta* (hour glass) e *cintura estreita* (narrow waist) para descrever o papel do protocolo de Internet no conjunto TCP/IP. A Figura 4.4 (b) faz com que a terminologia fique óbvia – embora vários protocolos independentes possam existir acima do IP e várias redes possam existir abaixo do IP, todo o tráfego de saída ou de entrada deve passar pelo IP.

Se um diagrama conceitual de camada não reflete com precisão a organização de módulos de software, por que ele é usado? Apesar de um modelo de camada não captar todos os detalhes, ele ajuda a explicar alguns conceitos gerais. Por exemplo, mesmo que ele não dê os detalhes sobre protocolos específicos, a Figura 4.4 (a) nos ajuda a entender que uma mensagem de saída irá percorrer três camadas intermediárias de protocolo

antes de ser enviada pela rede. Além disso, podemos usar o modelo em camadas para explicar a diferença entre sistemas finais (computadores dos usuários) e sistemas intermediários (roteadores). A Figura 4.5 mostra as camadas utilizadas em uma internet com três redes conectadas por dois roteadores.

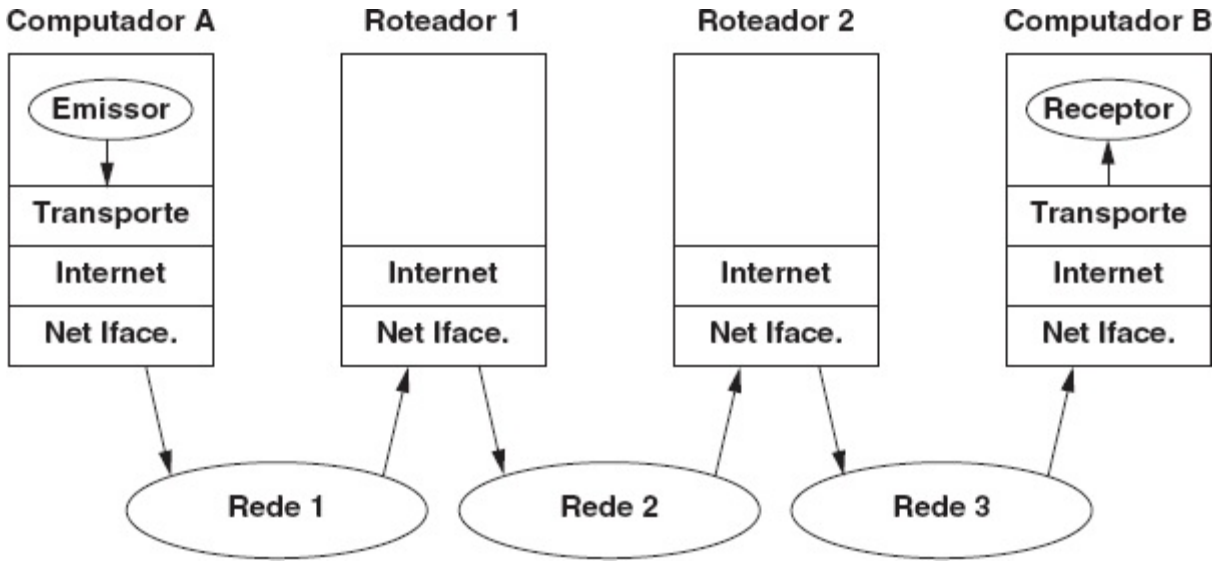


Figura 4.5 Camadas conceituais de protocolos necessários em computadores e roteadores para transferir uma mensagem de um aplicativo no computador A para um aplicativo no computador B.

Na figura, um aplicativo de envio no computador *A* usa um protocolo de transporte para enviar dados para uma aplicação recebedora no computador *B*. A mensagem passa para baixo na pilha de protocolos no computador *A* e é transmitida através da rede para um roteador 1. Quando atinge o primeiro roteador, o pacote passa até a camada internet (Camada 3), que encaminha o pacote através da rede 2 para o roteador 2. Nele, a mensagem passa para a Camada 3, e é encaminhada pela rede 3 para o destino. Quando atinge a máquina de destino final, a mensagem passa à camada de transporte, que a entrega para o aplicativo de recebimento. Capítulos posteriores explicam como IP lida com encaminhamento e mostram por que um pacote em trânsito não usa o protocolo de transporte em um roteador.

4.8 Local de inteligência

A Internet representa um afastamento significativo de projetos de rede anteriores porque grande parte da inteligência está alocada fora da rede nos

sistemas finais (por exemplo, os computadores dos usuários). A rede telefônica original por voz ilustra a diferença. Na rede telefônica analógica, toda a inteligência estava localizada em comutadores telefônicos; os telefones tinham apenas eletrônica passiva (ou seja, um microfone, fone de ouvido, e um mecanismo usado para discar).

Por outro lado, os protocolos TCP/IP requerem computadores conectados para executar protocolos de transporte e aplicações, bem como protocolos das Camadas *3* e *2*. Já mencionamos que os protocolos de transporte implementam a confiabilidade fim a fim por meio da retransmissão de pacotes perdidos. Vamos aprender que os protocolos de transporte são complexos, e que um computador conectado à Internet também deve participar de encaminhamento, pois deve escolher um roteador para usar quando estiver enviando pacotes. Dessa forma, ao contrário do sistema de telefonia analógica, uma internet TCP/IP pode ser vista como um sistema relativamente simples de envio de pacotes para o qual hosts inteligentes se conectam. O conceito é fundamental conforme descrito a seguir.

Protocolos TCP/IP alocam muito da inteligência em hosts – roteadores na Internet encaminham pacotes Internet, mas não participam em serviços de maior nível.

4.9 O princípio de camadas de protocolos

Independentemente do regime de camada particular ou das funções das camadas, a operação de protocolos em camadas é baseada numa ideia fundamental, que é chamada de *princípio de camadas* e pode ser definida de forma sucinta conforme descrito a seguir.

Protocolos em camadas são projetados de modo que a camada n *no destino recebe exatamente o mesmo objeto enviado pela camada* n *na fonte.*

Embora possa parecer óbvio ou mesmo trivial, o princípio de camadas fornece uma base importante que nos ajuda a projetar, implementar e entender protocolos. Especificamente, o princípio de camadas oferece:

- independência de projeto de protocolos;
- definição da propriedade fim a fim.

Independência de projetos de protocolos. Ao colocar uma garantia sobre os itens que passam entre cada par de camadas, o princípio de camadas permite que os designers de protocolo trabalhem em uma camada de cada vez. O designer de protocolo pode se concentrar na troca de mensagens para uma determinada camada com a garantia de que as camadas mais baixas não irão alterar mensagens. Por exemplo, ao criar um aplicativo de transferência de arquivos, um designer só precisa imaginar duas cópias do pedido de transferência de arquivos em execução em dois computadores. A interação entre as duas cópias pode ser planejada, sem pensar em outros protocolos, porque o designer pode assumir que cada mensagem será entregue exatamente como foi enviada. A ideia de que a rede não deve mudar mensagens parece tão óbvia para programadores de aplicativos que a maioria deles não pode imaginar elaborar aplicativos de rede sem considerá-la.

Felizmente, o princípio de camadas também funciona para a concepção de protocolos de camadas mais baixas. Em cada camada, um designer pode depender de que o princípio de camadas seja imposto para camadas inferiores; tudo que um projetista tem de fazer é garantir o princípio de camadas para a próxima camada superior. Por exemplo, quando um desenvolvedor de protocolo trabalha em um novo protocolo de transporte, pode assumir que o módulo de protocolo de transporte na máquina de destino recebe a mensagem que é enviada pelo módulo de protocolo de transporte na máquina emissora. A ideia fundamental é a de que um protocolo de transporte pode ser projetado independentemente de outros protocolos.

Definição da propriedade fim a fim. Informalmente, podemos classificar uma tecnologia de rede como *fim a fim*, se esta permite a comunicação da fonte original com o destino final. A definição informal também é usada com protocolos. O princípio de camadas nos permite sermos mais precisos: dizemos que um protocolo é *fim a fim* se, e somente se, o princípio de camadas se aplica entre a fonte original e destino final. Outros protocolos são classificados como *máquina a máquina*, pois o princípio das camadas se aplica somente através de um salto de rede. A próxima seção explica como o princípio de camadas se aplica a protocolos de Internet.

4.10 O princípio de camada aplicado a uma rede

Para entender como o princípio de camadas se aplica na prática, considere dois computadores conectados a uma rede. A Figura 4.6 ilustra as camadas de software de protocolo em execução em cada computador e as mensagens que passam pelas camadas.

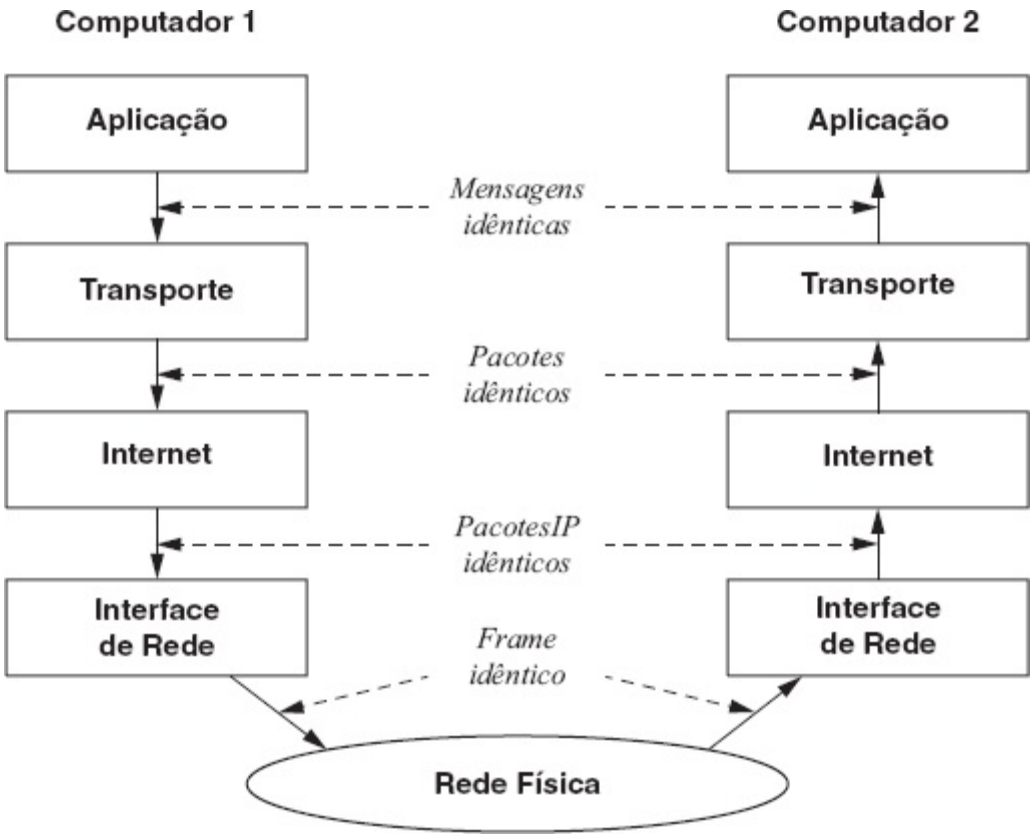


Figura 4.6 O princípio de camadas quando uma mensagem passa através de uma rede de um aplicativo em um computador para um aplicativo em outro.

4.10.1 Camadas em um ambiente Internet TCP/IP

Nossa ilustração do princípio de camadas é incompleta porque o diagrama na Figura 4.6 mostra apenas camadas de dois computadores conectados a uma única rede. Como o princípio de camadas se aplica a uma internet que pode transferir mensagens através de múltiplas redes? A Figura 4.7 responde a essa pergunta mostrando um exemplo em que uma mensagem de um programa de aplicação em um computador é enviada para um programa de aplicação em outro computador através de um roteador.

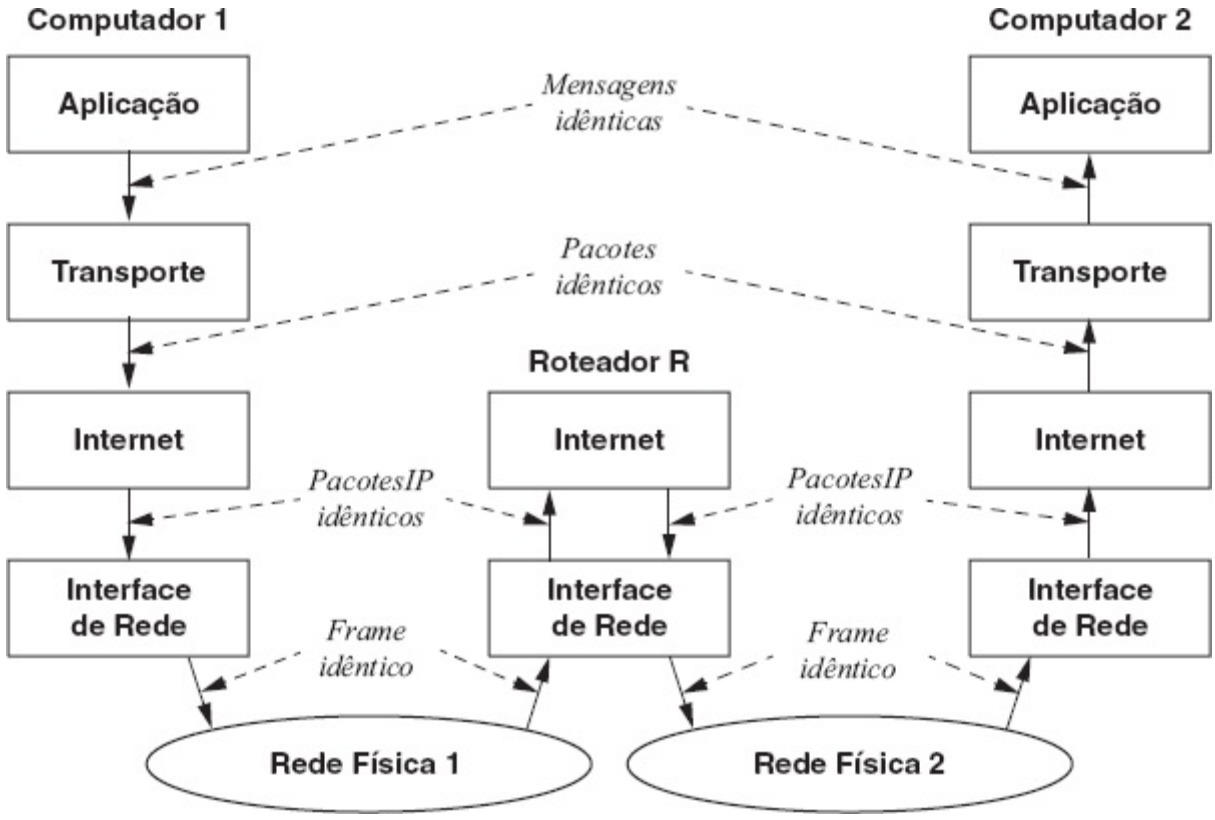


Figura 4.7 O princípio de camadas quando uma mensagem passa de um aplicativo em um computador, através de um roteador, e é entregue a um aplicativo em outro computador.

Como mostra a figura, a entrega da mensagem utiliza dois frames de rede separados, um para a transmissão a partir do computador 1 para o roteador *R* e outro do roteador *R* para o computador 2. O princípio de rede em camadas é estabelecer que o frame emitido para *R* é idêntico ao enviado pelo computador 1 e o frame entregue ao computador 2 é idêntico ao enviado pelo roteador *R*. No entanto, os dois frames certamente vão ser diferentes. Por outro lado, para os protocolos de aplicação de transporte, o princípio de camadas se aplica fim a fim. Ou seja, a mensagem entregue no computador 2 é exatamente a mesma mensagem que o protocolo par no computador 1 enviou.

É fácil entender que, para as camadas mais altas, o princípio de camadas se aplica fim a fim, e que, na camada mais baixa, se aplica à transferência de uma única máquina. Não é tão fácil de ver como o princípio de camadas aplica-se à camada internet. Por um lado, o objetivo do projeto da Internet é apresentar uma grande rede virtual, com a camada de internet enviando pacotes através da internet virtual de forma análoga ao modo que o

hardware de rede envia frames através de uma única rede. Assim, parece lógico imaginar um pacote IP sendo enviado da fonte original por todo o caminho até o destino final, e imaginar o princípio de camadas garantindo que o destino final receba exatamente o pacote IP que a fonte original enviou. Por outro lado, vamos aprender que um pacote IP contém campos, como um contador de *tempo de vida* que deve ser mudado cada vez que o pacote passa por um roteador. Assim, o destino final não receberá exatamente o mesmo pacote IP como a fonte enviou. Concluímos que, embora a maior parte do pacote IP permaneça intacta à medida que passa através de uma internet TCP/IP, o princípio de camadas só se aplica a transferências de pacotes através de máquinas individuais. Portanto, a Figura 4.7 mostra a camada de internet fornecendo um serviço máquina a máquina, em vez de um serviço fim a fim.

4.11 Camadas em redes mesh

As tecnologias de hardware usadas na maioria das redes garante que cada computador conectado possa alcançar outros computadores diretamente. No entanto, algumas tecnologias não garantem conexões diretas. Por exemplo, a tecnologia sem fio ZigBee, descrita no Capítulo 2, usa rádios sem fio de baixa potência que têm alcance limitado. Consequentemente, se os sistemas ZigBee são implantados em vários cômodos de uma residência, a interferência de estruturas metálicas pode significar que um determinado rádio pode ser capaz de chegar a alguns, mas não a todos os outros rádios. Da mesma forma, um grande provedor de acesso pode optar por alugar um conjunto de circuitos ponto a ponto digitais para interligar vários sites. Apesar de cada rádio ZigBee só poder chegar a um subconjunto dos nós e de cada circuito digital só conectar dois pontos, falamos de uma "rede" ZigBee e dizemos que um ISP tem uma "rede". A fim de distinguir tais tecnologias de tecnologias de rede convencionais, usamos o termo *rede mesh* para caracterizar um sistema de comunicação construído a partir de muitos links individuais.

Como uma rede mesh se encaixa em nosso modelo de camadas? A resposta depende de como os pacotes são enviados através dos links. Por um lado, se o encaminhamento ocorre na Camada *2*, toda a rede pode ser modelada como uma única rede física. Usamos o termo *mesh-under* para descrever tal situação. Por outro lado, se o IP lida com encaminhamento, a rede mesh deve ser modelada como redes individuais. Usamos o termo

route-over IP, muitas vezes abreviado para *route-over*, para descrever esses casos.

Route-over. A maioria das redes ISP usa route-over. O ISP utiliza circuitos digitais alugados para interligar roteadores, e um roteador individual vê cada circuito como uma única rede. O IP lida com todos os encaminhamentos, e o roteador usa protocolos de roteamento de Internet padrão (descritos nos capítulos posteriores) para construir as tabelas de encaminhamento.*

Mesh-under. A tecnologia IEEE 802.15.4 usada em redes ZigBee pode ser configurada para funcionar como links individuais ou como uma rede completa. Ou seja, eles podem se organizar em uma única rede mesh, ao concordar em descobrir vizinhos e formar uma rede Camada *2 mesh-under* que encaminha pacotes sem usar IP, ou eles podem formar links individuais e permitir ao IP lidar com encaminhamento. Com relação ao nosso modelo de camada, a única mudança que a abordagem mesh-under introduz é um módulo de software adicionado à interface de rede para controlar o encaminhamento em links individuais. Dizemos que o novo software controla o *encaminhamento intrarrede*. O novo software é muitas vezes referido como uma *subcamada intranet*, conforme a Figura 4.8 ilustra.

Além de uma subcamada intranet que lida com o encaminhamento através do conjunto de links individuais, nenhuma outra alteração é necessária para o esquema geral de camadas para acomodar a abordagem mesh-under. Curiosamente, o ZigBee usa uma pequena modificação das ideias anteriormente descritas. Embora recomende o uso da abordagem route-over, o consórcio ZigBee não recomenda o uso de protocolos de roteamento IP padrão. Em vez disso, a pilha ZigBee usa um protocolo de roteamento especial que aprende sobre destinos na malha ZigBee e depois configura o encaminhamento IP através dos links individuais.

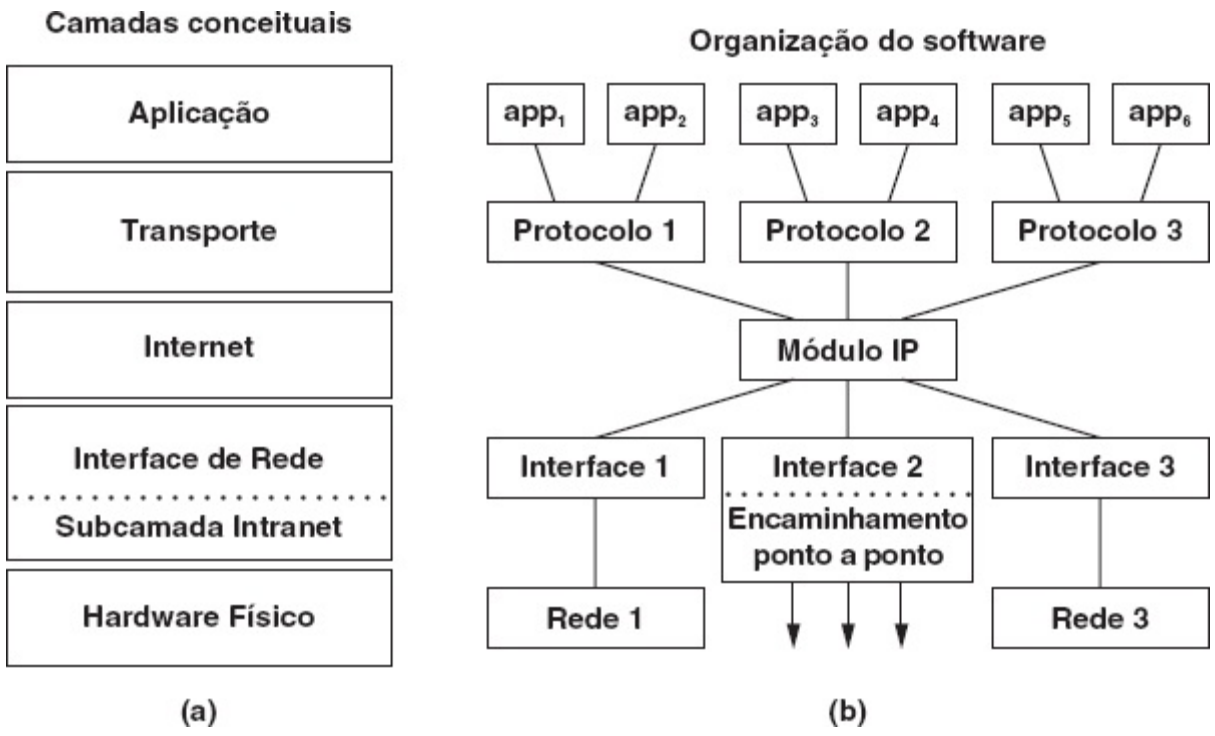


Figura 4.8 (a) posição conceitual de uma subcamada intranet que trata de encaminhamento usando a abordagem mesh-under e (b) a organização de software correspondente.

A desvantagem principal da abordagem route-over é que proliferam muitas rotas na camada IP (um para cada ligação entre duas máquinas), fazendo com que as tabelas de encaminhamento IP sejam maiores do que o necessário. A principal desvantagem da abordagem mesh-under é que ela usa uma tabela de encaminhamento e um protocolo de roteamento separados para atualizar a tabela de encaminhamento. O protocolo de encaminhamento extra significa tráfego adicional, mas também porque a rede mesh é muito menor do que a Internet e pode ser muito mais estática. Um protocolo mesh de roteamento para fins especiais pode ser mais eficiente do que um protocolo de roteamento IP de uso geral. Uma desvantagem final da abordagem mesh-under é que o roteamento intranet evita que o roteamento IP faça com que problemas de roteamento sejam mais difíceis de diagnosticar e reparar.

4.12 Dois limites importantes no modelo TCP/IP

O modelo de camadas inclui dois limites conceituais que podem não ser óbvios: uma fronteira do protocolo de endereço que separa endereçamentos de alto e baixo níveis e um limite de sistema operacional que separa

software de protocolo de programas aplicativos.

A Figura 4.9 ilustra os limites e as próximas seções os explicam.

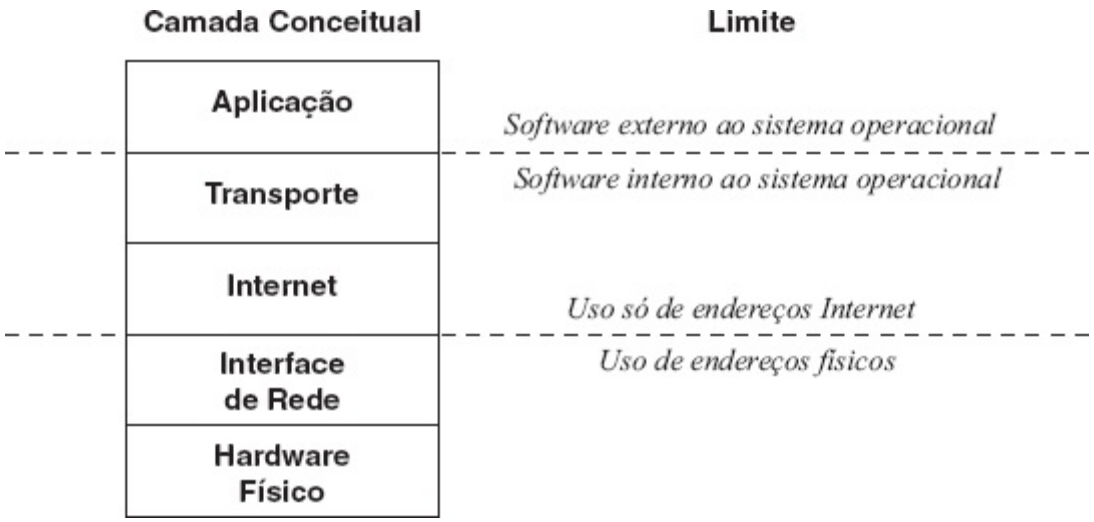


Figura 4.9 Dois limites conceituais no modelo de camadas.

4.12.1 Limite de protocolo de endereço de alto nível

O Capítulo 2 descreve os endereços usados por vários tipos de hardware de rede. Capítulos posteriores descrevem protocolos e endereçamento de Internet. É importante distinguir onde são utilizadas as duas formas de endereçamento, e o modelo de camadas deixa claro: não há uma fronteira conceitual entre Camadas *2* e *3*. Endereços de hardware (MAC) são usados nas Camadas *1* e *2*, mas não acima. Endereços Internet são usados por Camadas de *3* a *5*, mas não pelo hardware subjacente. Podemos fazer então o resumo a seguir.

> *Programas aplicativos e todo software de protocolo da camadas acima do nível internet usa somente endereços Internet; os endereços usados pelo hardware de rede ficam isolados nas camadas inferiores.*

4.12.2 Limite de sistema operacional

A Figura 4.9 ilustra um outro limite importante: a divisão entre software de protocolo que é implementado em um sistema operacional e aplicativos de software que não são. Embora os pesquisadores tenham feito experiências, tornando o TCP/IP parte de um aplicativo, a maioria das implementações

coloca o software de protocolo no sistema operacional onde ele pode ser compartilhado por todos os aplicativos. A fronteira é importante, porque a transferência dos dados entre os módulos dentro do sistema operacional é muito menos dispendiosa do que a passagem de dados entre o sistema operacional e um aplicativo. Além disso, um API especial é necessário para permitir a um aplicativo interagir com o software de protocolo. O Capítulo 21 analisa o limite em mais detalhes e descreve uma interface de exemplo que um sistema operativo fornece a aplicações.

4.13 Otimizações Cross-Layer

Dissemos que estratificação é uma ideia fundamental que fornece a base para o projeto de protocolo. Ela permite que o projetista divida um problema complicado em subproblemas e resolva cada um de forma independente. Infelizmente, o software resultante rigorosamente a partir de camadas pode ser extremamente ineficiente. Como exemplo, consideremos o emprego de camada de transporte. Ela deve aceitar um fluxo de bytes a partir de um programa aplicativo, dividir o fluxo em pacotes e enviar cada pacote através da internet subjacente. Para otimizar a transferência, a camada de transporte deve escolher o maior tamanho de pacote que permita que um pacote viaje em um frame de rede. Em particular, se a máquina de destino se conecta diretamente à mesma rede que a fonte, apenas uma rede física é envolvida na transferência e o remetente pode otimizar o tamanho do pacote para essa rede. Se o software de protocolo preserva estritamente as camadas, entretanto, a camada de transporte pode não saber como o módulo de internet vai encaminhar o tráfego ou as redes a que se ligam diretamente. Além disso, a camada de transporte não vai entender os formatos de pacotes usados por camadas inferiores, nem será capaz de determinar quantos octetos de cabeçalho serão adicionados à mensagem que envia. Assim, a utilização rigorosa de camadas impedirá que a camada de transporte otimize as transferências.

Normalmente, os implementadores relaxam o esquema de camadas rígidas quando da construção de software de protocolo. Eles permitem que as camadas superiores de uma pilha de protocolos obtenham informação, tal como o tamanho máximo do pacote ou a rota a ser usada. Ao alocar os buffers de pacote, os protocolos da camada de transporte podem usar as informações para otimizar o processamento, deixando espaço suficiente para cabeçalhos que serão adicionados por protocolos de camada inferior. Da mesma forma, os protocolos de camada inferior, muitas vezes, mantêm todos os cabeçalhos em um frame de entrada ao passarem o frame para protocolos de camada superior. Essas otimizações podem trazer grandes

melhorias em eficiência, mantendo a estrutura básica de camadas.

4.14 A ideia básica por trás da multiplexação e demultiplexação

Protocolos de comunicação em camadas utilizam um par de técnicas conhecidas como *multiplexação* e *demultiplexação* através da hierarquia de camadas. Ao enviar uma mensagem, o computador-fonte inclui bits extras que armazenam os metadados, tais como o tipo de mensagem, a identidade do programa de aplicação que enviou os dados e o conjunto de protocolos que foram utilizados. Na extremidade receptora, um computador de destino usa os metadados para orientar o processamento.

A Ethernet é um exemplo básico. Cada frame Ethernet inclui um campo de *tipo* que especifica o que o frame carrega. Nos próximos capítulos, veremos que um frame Ethernet pode conter um pacote IP, um pacote ARP, ou um pacote RARP. O remetente define o tipo de campo para indicar o que está sendo enviado. Quando o frame chega, o software de protocolo no computador receptor usa o tipo indicado para escolher um módulo de protocolo para processar o frame. Dizemos que o software *demultiplexa* os frames de entrada. A Figura 4.10 ilustra o conceito.

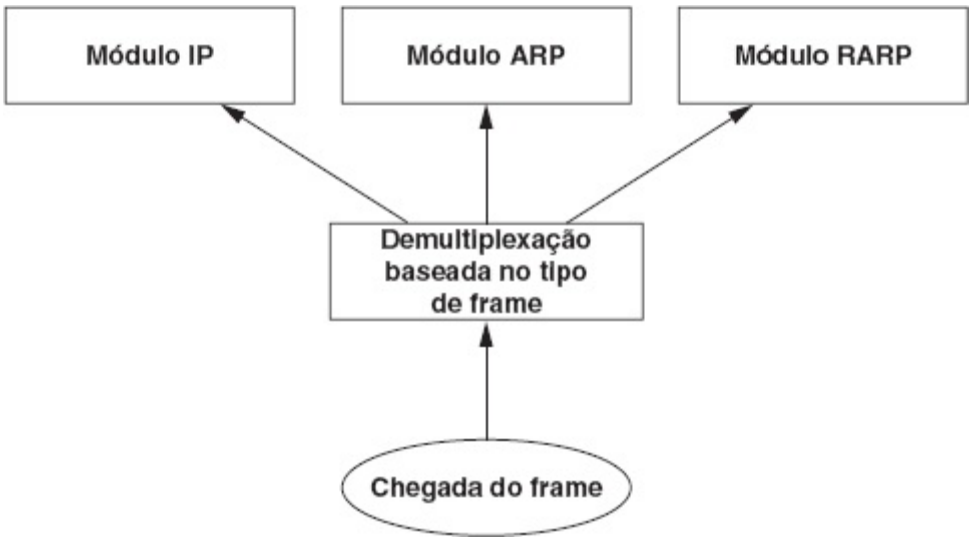


Figura 4.10 Ilustração do frame demultiplexação que usa um tipo de campo no cabeçalho do frame. A demultiplexação é usada com a maioria das redes, incluindo Ethernet e Wi-Fi.

A multiplexação e a demultiplexação ocorrem em cada camada. A demultiplexação, ilustrada na Figura 4.10, ocorre na camada de interface de

rede, Camada *2*. Para entender demultiplexação na Camada *3*, considere um frame que contém um pacote IP. Vimos que um frame demultiplexação vai passar o pacote para o módulo IP para processamento. Uma vez que ele verificou que o pacote é válido (ou seja, na verdade foi entregue para o destino correto), o IP irá demultiplexar ainda mais, passando o pacote para o módulo de protocolo de transporte apropriado.

Como o software IP sabe qual protocolo de transporte o remetente usou? Analogamente a um frame Ethernet, cada pacote IP possui um campo de *tipo* no cabeçalho. O remetente define o campo de tipo de IP para indicar qual foi o protocolo de transporte utilizado. Nos próximos capítulos, vamos aprender sobre TCP, UDP e ICMP, cada um dos quais pode ser enviado em um pacote IP. A Figura 4.11 ilustra como o IP demultiplexa entre os três exemplos.

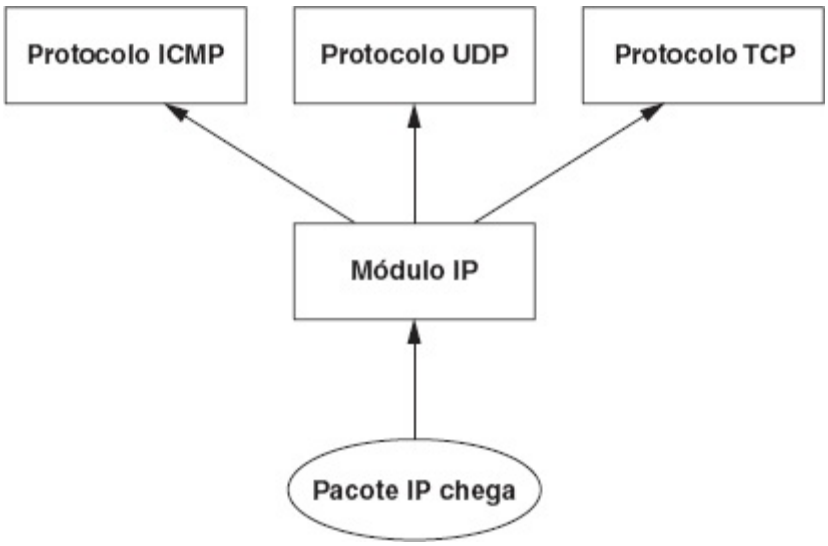


Figura 4.11 Ilustração de demultiplexação de pacotes IP recebidos com base no tipo de campo no cabeçalho IP.

Nossa discussão sobre demultiplexação deixa muitas perguntas sem respostas. Como podemos garantir que o emissor e o receptor concordam com os valores utilizados em um campo de tipo? O que acontece se um pacote de entrada contém um tipo diferente dos tipos que o receptor pode manipular? Capítulos posteriores vão fornecer mais informações sobre demultiplexação, mas podemos dar agora respostas curtas para as perguntas anteriores. Se um receptor não entende o tipo em um pacote que chega, este é descartado. Para garantir um acordo universal sobre os tipos, os organismos de normatização especificaram valores a serem utilizados (por

exemplo, IEEE especifica o conjunto de valores para os tipos de Ethernet e o IETF especifica valores para o protocolo de Internet). Uma vez que remetentes e receptores concordam cada um em seguir as normas, não surgem problemas. Claro, os pesquisadores às vezes conduzem experimentos que usam tipos não atribuídos. A regra de os computadores descartarem pacotes desconhecidos ajuda – mesmo se um pesquisador transmitir um pacote com um tipo experimental, nenhum dano ocorrerá porque os computadores que não entenderem o tipo irão descartar o pacote.

4.15 Resumo

Os protocolos são os padrões que especificam todos os aspectos da comunicação através de uma rede de computadores. Protocolos especificam tanto a sintaxe (por exemplo, o formato de mensagens), como a semântica (por exemplo, como dois computadores trocam mensagens). Protocolos incluem detalhes, como tensões, como os bits são enviados, como são detectados erros e como o emissor e o receptor concordam que uma mensagem foi transferida com sucesso. Para simplificar o projeto e a implementação do protocolo, a comunicação é segregada em subproblemas que podem ser resolvidos de forma independente. Cada subproblema é atribuído a um protocolo separado.

A ideia de camadas é fundamental, pois fornece uma estrutura conceitual para o projeto de protocolo que nos permite dividir o problema em partes gerenciáveis. Em um modelo de camadas, cada camada lida com uma parte do problema de comunicação. Protocolos seguem o princípio de camadas, que afirma que o software que implementa a camada n na máquina de destino recebe exatamente a mensagem enviada pelo software que implementa a camada n na máquina de origem.

Foi analisado o modelo de referência da Internet 5 camadas, bem como o modelo de referência ISO de 7 camadas mais antigo. Em ambos os casos, o modelo de camadas fornece apenas um frame conceitual para o software de protocolo. Na prática, vários protocolos podem ocorrer em cada camada, e software de protocolo usa demultiplexação para distinguir entre vários protocolos dentro de uma determinada camada. A presença de vários protocolos em cada camada faz os softwares de protocolos ficarem mais complexos do que os modelos de camadas sugeridos.

EXERCÍCIOS

4.1 Uma das principais objeções a camadas de protocolos decorre da

aparente sobrecarga – cópias ocorrem em cada camada. Como as cópias podem ser eliminadas?

4.2 Protocolos em camadas escondem todos os detalhes subjacentes de aplicativos. Poderia um software de aplicativo ser otimizado se um aplicativo conhecesse as redes subjacentes que estão sendo usadas? Explique.

4.3 É possível para os protocolos de Internet incluírem uma camada de apresentação que especifica normas para cada tipo de dados (por exemplo, um tipo de imagem gráfica, um tipo de música digital etc.)? Sim ou não? Por quê?

4.4 Construa um caso em que o TCP/IP está se movendo em direção a uma arquitetura de protocolo de seis camadas, que inclua uma camada de apresentação. (Dica: vários programas usam o protocolo XDR, XML e ASN.1.)

4.5 Descubra como um sistema UNIX usa a estrutura *mbuf* para tornar o software de protocolo camadas eficientes.

---

* O ITU ficou formalmente conhecido como *CCITT*.

* Como o Capítulo 9 explica, é possível usar um mecanismo de *link anônimo* que não atribua um prefixo IP para cada link; utilizar ligações não numeradas não altera a estratificação.

CAPÍTULO 5

Endereçamento Internet

CONTEÚDOS DO CAPÍTULO

5.25 Resumo

5.1 Introdução

O Capítulo 3 define uma internet TCP/IP como uma rede virtual construída pela interconexão de redes físicas com roteadores. Este capítulo inicia uma discussão sobre endereçamento, uma parte essencial do projeto que ajuda o software TCP/IP a ocultar detalhes de rede física e torna a internet resultante parecer uma entidade única, uniforme.

Além de discutir o endereçamento Internet tradicional, o capítulo apresenta os endereços IPv6. O esquema de endereçamento tradicional, que foi introduzido com a versão 4 do protocolo Internet, é amplamente utilizado. A próxima versão do Protocolo de Internet, versão 6, já começou a aparecer, e acabará por substituir o IPv4.

5.2 Identificadores universais de host

O TCP/IP usa o termo *host* para se referir a um sistema terminal que se conecta à Internet. Um host pode ser um grande e poderoso computador de uso geral ou um pequeno sistema para fins especiais. Pode ter uma interface que pessoas usam (por exemplo, uma tela e teclado) ou pode ser um dispositivo embutido, como uma impressora de rede. Um host pode usar a tecnologia de rede com ou sem fio. Em suma, a Internet divide todas as máquinas em duas classes: os roteadores e hosts. Qualquer dispositivo que

não é um roteador é classificado como um host. Vamos usar a terminologia em todo o restante do texto.

Um sistema de comunicação se diz ser para fornecer *serviços de comunicação universal* se o sistema permite que um host conectado se comunique com qualquer outro host conectado. Para tornar o nosso sistema de comunicação universal, é necessário um método globalmente aceito de identificar cada host que se conecta a ele.

Muitas vezes, os identificadores são classificados como *nomes*, *endereços* ou *rotas*. Shoch sugere que um nome identifique *o que* um objeto é, um endereço identifique *onde* está e uma rota diga *como* chegar lá.* Apesar de serem intuitivamente atraentes, as definições podem ser enganosas. Nomes, endereços e rotas na verdade se referem a representações de nível sucessivamente inferior de identificadores de host. Em geral, as pessoas preferem usar nomes pronunciáveis para identificar os computadores, enquanto o software funciona de forma mais eficiente com identificadores binários compactos que nós pensamos como endereços ou que poderiam ter sido escolhidos como os identificadores de host TCP/IP.

A decisão foi tomada para padronizar em endereços binários compactos que fazem cálculos, tais como a seleção de um próximo salto eficiente. Por enquanto, vamos apenas discutir endereços binários, adiando as questões de como o mapear entre endereços binários e nomes pronunciáveis, e como usar os endereços para encaminhar pacotes.

Pensamos em uma internet como uma grande rede, como qualquer outra rede física. A diferença, claro, é que uma internet é uma estrutura virtual, imaginada por seus criadores e implementada por software de protocolo em execução em hosts e roteadores. Como uma internet é virtual, seus projetistas são livres para escolher formatos e tamanhos de pacotes, endereços, técnicas de entrega, e assim por diante. Nada é ditado por hardware.

Os projetistas do TCP/IP escolheram um esquema análogo à rede de endereçamento físico, em que a cada host em uma internet é atribuído um endereço único inteiro chamado de *endereço de Protocolo Internet* ou *endereço IP*. A parte inteligente do endereçamento da internet é que os inteiros são cuidadosamente escolhidos para tornar eficiente o encaminhamento. Especificamente, um endereço de IP é dividido em duas partes: um prefixo do endereço identifica a rede a que o host está conectado e um sufixo identifica um host específico na rede. Ou seja, todos os hosts

anexados à mesma rede compartilham um prefixo comum. Veremos mais adiante por que a divisão é importante. Por enquanto, é suficiente lembrar a afirmação a seguir.

> *A cada host em uma internet IPv4, é atribuído um endereço exclusivo que é utilizado em toda a comunicação com o host. Para fazer o encaminhamento eficiente, um prefixo do endereço identifica uma rede e um sufixo identifica um host na rede.*

Os projetistas também decidiram fazer endereços IP com tamanho fixo (foram escolhidos 32 bits para IPv4 e 128 bits para IPv6). Conceitualmente, cada endereço é um par (*netid, hostid*), no qual o *netid* identifica uma rede e o *hostid* identifica um host na rede. Uma vez que as decisões tomadas foram para usar endereços IP de tamanho fixo e dividir cada endereço em uma ID de rede e uma ID de host, surge uma pergunta: que tamanho deve ter cada parte? A resposta depende do tamanho das redes que esperamos na nossa internet. A alocação de muitos bits para o prefixo de rede permite que a nossa internet tenha muitas redes, mas limita o tamanho de cada rede. Alocação de muitos bits para um sufixo de host significa que uma determinada rede pode ser grande, mas limita o número de redes em nossa internet.

5.3 O esquema original de endereçamento Classful IPv4

Esta seção descreve o mecanismo de endereçamento IPv4 original. Embora a maior parte dele não seja mais usada, o apresentamos aqui porque ele explica como o espaço de endereço multicast IPv4 foi escolhido. Também nos ajuda a entender endereçamento de sub-rede, que será tratado na próxima seção e que evoluiu para o atual sistema de endereçamento classless.

Para entender endereçamento, observe que uma internet permite tecnologias arbitrárias de rede, o que significa que ele irá conter uma mistura de redes grandes e pequenas. Para acomodar a mistura, os criadores de software não escolhem uma única divisão de endereço. Em vez disso, eles inventaram um esquema de endereçamento *classful* que permitiu uma determinada rede ser grande, média ou pequena. A Figura 5.1 ilustra como o esquema classful original divide cada endereço IPv4 em duas partes.

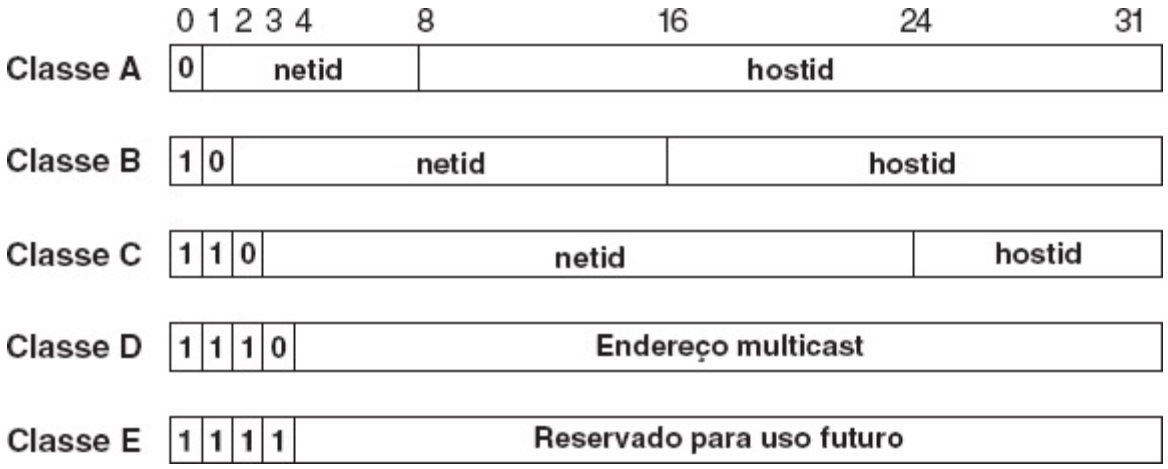


Figura 5.1 As cinco formas de endereços de Internet (IP) utilizadas com o esquema de endereçamento Classful IPv4 original.

No esquema de endereçamento classful, cada endereço é dito *autoidentificável*, pois a fronteira entre o prefixo e o sufixo pode se computada apenas a partir do endereço, sem referência a informação externa. Em particular, a classe de um endereço pode ser determinada a partir dos três bits de ordem superior, com dois bits sendo suficientes para distinguir entre as três classes primárias. Os endereços de classe *A*, usados para o punhado de redes grandes que têm mais de 2^{16} (ou seja, 65.536) hosts, dedica 7 bits para a rede ID e 24 bits para identificação de host. Os endereços de classe *B*, usados para redes de médio porte que tenham entre 2^8 (ou seja, 256) e 2^{16} hosts, aloca 14 bits para a identificação da rede e 16 bits para a identificação de host. Finalmente, os endereços classe *C*, utilizados para redes que têm menos de 2^8 hosts, alocam 21 bits para a identificação da rede e apenas 8 bits para a identificação do host.

5.4 Notação decimal com pontos usada com IPv4

Quando comunicado a pessoas, quer em documentos técnicos ou através de programas aplicativos, os endereços IPv4 são escritos como quatro números inteiros decimais separados por pontos decimais, em que cada inteiro dá o valor de um octeto do endereço.* Assim, o endereço na internet de 32 bits

10000000 00001010 00000010 00011110

é escrito

128.10.2.30

Usaremos notação decimal ao expressar endereços IPv4 em todo o restante do texto. Na verdade, a maioria dos softwares TCP/IP que exibem ou exigem uma pessoa para inserir um endereço IPv4 usa a notação decimal com pontos. Por exemplo, programas de aplicação, como um navegador da web, permitem que um usuário entre com um valor decimal com pontos em vez de um nome de computador. Como um exemplo de decimal com pontos, a tabela na Figura 5.2 resume os valores decimais com pontos para cada classe de endereço.

Endereço	Endereço mais baixo	Endereço mais alto
A	1 . 0 . 0 . 0	127 . 0 . 0 . 0
B	128 . 0 . 0 . 0	191 . 255 . 0 . 0
C	192 . 0 . 0 . 0	223 . 255 . 255 . 0
D	224 . 0 . 0 . 0	239 . 255 . 255 . 255
E	240 . 0 . 0 . 0	255 . 255 . 255 . 254

Figura 5.2 A gama de valores decimais com pontos que correspondem a cada uma das classes originais de endereços IPv4.

5.5 Endereçamento em sub-rede IPv4

No início dos anos 1980, quando redes locais se tornaram amplamente disponíveis, ficou evidente que o esquema de endereçamento classful teria endereços de rede insuficientes, especialmente prefixos classe B. Surgiu a questão: como a tecnologia podia acomodar o crescimento sem abandonar o esquema de endereçamento classful original? A primeira resposta foi uma técnica chamada *endereçamento de sub-rede* ou *sub-rede*. Ela permite que um único prefixo de rede seja usado para várias redes físicas. Embora pareça violar o esquema de endereçamento, a sub-rede tornou-se parte do padrão e foi amplamente utilizada.

Para entender as sub-redes, é importante pensar em sites individuais conectados à Internet. Imagine, por exemplo, que uma universidade começou com uma única rede de área local e obteve um prefixo IPv4. Se essa universidade acrescenta outra LAN, o esquema de endereçamento original exigiria que ela obtivesse um segundo ID de rede para a segunda LAN. No entanto, suponha que a universidade tem apenas alguns computadores. Desde que esconda os detalhes do resto da Internet, ela pode atribuir endereços de host e organizar encaminhamento interno como

quiser. Ou seja, um site pode optar por atribuir e usar endereços IPv4 internamente de forma não usual, desde que:

- todos os hosts e roteadores dentro do site concordem em honrar o esquema de endereçamento do site;
- outros sites na Internet possam tratar endereços como endereços-padrão com um prefixo de rede que pertence ao site.

O endereçamento sub-rede aproveita a liberdade ao permitir a um site dividir a parte do host de seus endereços entre várias redes. A maneira mais fácil de ver como o endereçamento sub-rede funciona é considerar um exemplo. Suponha que a um site tenha sido atribuído um único prefixo de classe *B*, 128.10.0.0. O resto da Internet assume que cada endereço no site tem uma rede física com o ID da rede de 16 bits 128.10. Se o site obtém uma segunda rede física, o local pode usar um endereçamento de sub-rede utilizando uma parte do campo de ID do host para identificar qual a rede física a ser usada. Somente os hosts e roteadores locais vão saber que existem várias redes físicas e como encaminhar o tráfego entre elas; roteadores e hosts no resto da Internet irão assumir que há uma única rede física no local com hosts conectados. A Figura 5.3 mostra um exemplo de uso do terceiro octeto de cada endereço para identificar uma sub-rede.

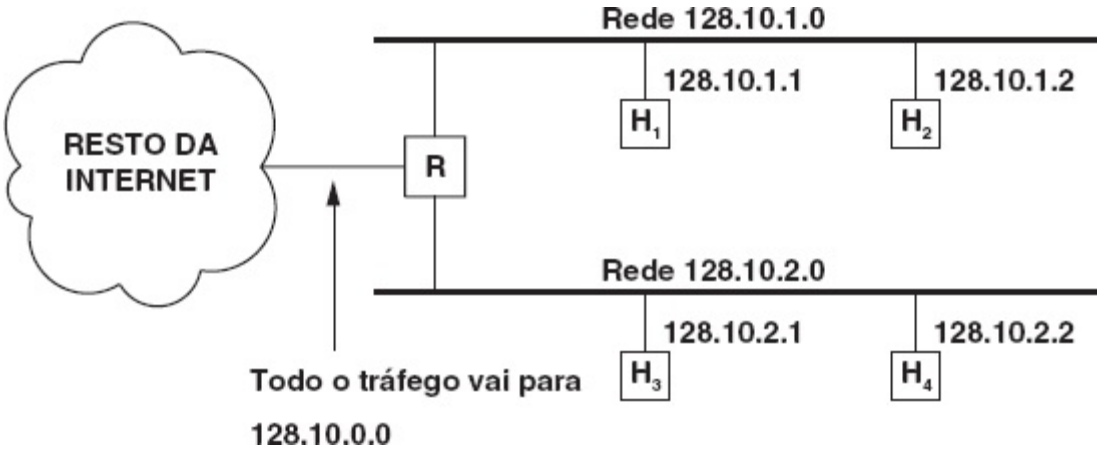


Figura 5.3 Ilustração de endereçamento de uma sub-rede IPv4 usando o terceiro octeto de um endereço para especificar uma rede física.

Na figura, o site decidiu atribuir a suas duas redes os números sub-rede 1 e 2. Todos os hosts da primeira rede têm endereços no formato

128.10.1.*,

onde um asterisco indica um host ID. Por exemplo, a figura mostra dois hosts na rede 1 com endereços 128.10.1.1 e 128.10.1.2. De forma similar, os hosts na rede 2 têm endereços no formato

128.10.2.*.

Quando o roteador *R* recebe um pacote, ele verifica o endereço de destino. Se o endereço começa com 128.10.1, o roteador entrega o pacote para um host na rede 1; se o endereço começa com 128.10.2, o roteador entrega o pacote para um host na rede 2. Vamos aprender mais sobre como roteadores encaminham pacotes. Por agora, é suficiente compreender que um roteador no site pode usar o terceiro octeto de endereço para escolher entre as duas redes.

Conceitualmente, a adição de sub-rede só muda ligeiramente a interpretação de endereços IPv4. Em vez de dividir o endereço IPv4 de 32 bits em um prefixo de rede e um sufixo de host, a sub-rede divide o endereço em uma *parte internet* e uma *parte local*. A interpretação da parte internet continua a ser a mesma que para as redes que não utilizam a sub-rede (isto é, que contém uma identificação de rede). No entanto, a interpretação da parte local de um endereço é deixada para o site (dentro dos limites da norma de endereçamento sub-rede formal). Podemos fazer o resumo a seguir.

Ao usar endereçamento sub-rede, pensamos em um endereço IPv4 de 32 bits como tendo uma parte de internet e uma parte local, onde a parte internet identifica um site, possivelmente com várias redes físicas, e a parte local identifica uma rede física e um host no site.

O exemplo da Figura 5.3 mostra endereçamento sub-rede com um endereço de classe *B* que tem uma parte internet de 2 octetos e uma parte local de 2 octetos. Para tornar eficiente o encaminhamento entre as redes físicas, o administrador do site no nosso exemplo escolheu usar um octeto da parte local para identificar uma rede física e o outro octeto para identificar um host na rede. A Figura 5.4 ilustra como o nosso exemplo divide o endereço IPv4.

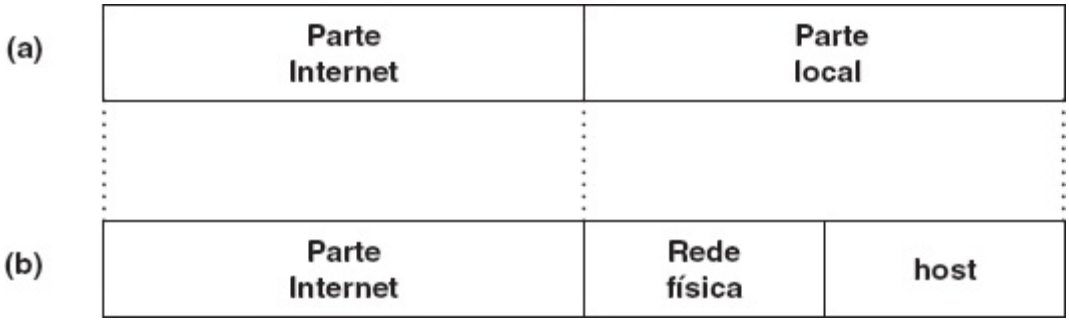


Figura 5.4 (a) A interpretação de um endereço IPv4 de 32 bits a partir da Figura 5.3, quando é usada sub-rede e (b) a parte local dividida em dois campos que identificam uma rede física no site e uma host na rede.

A sub-rede impõe uma forma de *endereçamento hierárquico* que leva a um *roteamento hierárquico*. Os roteadores na Internet usam o nível mais alto da hierarquia para transmitir um pacote para o site correto. Uma vez que o pacote entra no site, os roteadores locais utilizam o octeto da rede física para selecionar a rede correta. Quando o pacote chega à rede correta, um roteador usa a parte host para identificar um host específico.

O endereçamento hierárquico não é novo; muitos sistemas o usaram antes. Por exemplo, o sistema de telefonia dos Estados Unidos divide um número de telefone de dez dígitos em um código de área de três dígitos, três dígitos para intercâmbio, e quatro dígitos para conexão. A vantagem da utilização de endereçamento hierárquico é que esta acomoda um grande crescimento sem a necessidade de roteadores para obter informações sobre destinos distantes. Uma desvantagem é que a escolha de uma estrutura hierárquica é complicada e, muitas vezes, se torna difícil de alterar, uma vez que uma hierarquia foi estabelecida.

5.6 Sub-redes IPv4 de comprimento fixo

No exemplo anterior, foi designado a um site um prefixo de rede de 16-bit e utilizado o terceiro octeto de endereço para identificar uma rede física no site. O padrão TCP/IP para endereçamento sub-rede reconhece que nem todos os sites terão um prefixo de 16 bits e nem todos terão as mesmas necessidades de uma hierarquia de endereço. Assim, a norma permite flexibilidade ao site para escolher como atribuir sub-redes. Para entender por que essa flexibilidade é desejável, considere dois exemplos. A Figura 5.3 representa um exemplo, um site que só tem duas redes físicas. Como outro exemplo, imagine uma empresa que possui vinte grandes edifícios e

implantou vinte LANs em cada edifício. Suponha que o segundo site tem um único prefixo de rede de 16 bits e quer usar sub-redes para todas as suas redes. Como a parte local de 16 bits do endereço deve ser dividida em campos para uma rede física e host?

A divisão mostrada na Figura 5.4 resulta num identificador de rede física de 8 bits e um identificador host de 8 bits. Usar oito bits para identificar uma rede física significa que um gerente pode gerar até 256 números de redes físicas únicas. Da mesma forma, com oito bits para a identificação de host, um gerente pode gerar até 256 identificações de host para cada rede.* Infelizmente, a divisão não é suficiente para a empresa no nosso segundo exemplo, pois esta tem 400 redes, o que excede os 254 números possíveis.

Para permitir a flexibilidade, o padrão de sub-rede não especifica que um site deve sempre usar o terceiro octeto para especificar uma rede física. Em vez disso, um site pode escolher quantos bits da parte local dedicar à rede física e quantos dedicar ao ID do host. Nossa empresa exemplo, com 400 redes, pode escolher a divisão que a Figura 5.5 ilustra, porque um campo de 10 bits permite até 1.022 redes.

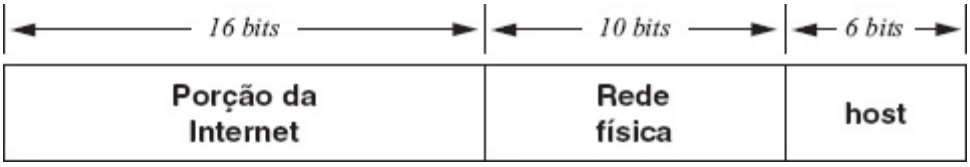


Figura 5.5 Uma divisão da parte local de 16 bits que acomoda 400 redes.

A ideia de permitir a um site a escolha de uma divisão para a parte local de seu endereço e, em seguida, usar a divisão para todo o site é conhecida como *sub-rede de comprimento fixo*. Essa definição é fácil entender porque a sub-rede particiona a parte local de um endereço entre redes e hosts. Em essência, como um gerente escolhe quantas redes o site pode ter, ele também determina o número máximo de hosts em uma determinada rede. A Figura 5.6 ilustra as possíveis escolhas se um site utiliza sub-rede de comprimento fixo com uma parte local de 16 bits.

Bits de Rede	Número de Redes	Hosts por Rede
0	1	65534
2	2	16382
3	6	8190
4	14	4094
5	30	2046
6	62	1022
7	126	510
8	254	254
9	510	126
10	1022	62
11	2046	30
12	4094	14
13	8190	6
14	16382	2

Figura 5.6 As possíveis maneiras de dividir uma parte local de 16 bits de um endereço IPv4 ao utilizar sub-rede de comprimento fixo. Um site deve escolher uma linha na tabela.

Como a figura ilustra, uma organização que adota sub-rede de comprimento fixo deve escolher uma solução de compromisso. Se a organização opta por um grande número de redes físicas, nenhuma delas pode conter muitos hosts; se a organização espera conectar muitos hosts a uma rede, o número de redes físicas deve ser pequeno. Por exemplo, se atribuirmos 3 bits para identificar uma rede, isso resulta em até seis redes que suportam até 8190 hosts cada uma. Atribuir 12 bits resulta em até 4094 redes, mas restringe o tamanho de cada uma a 14 hosts.

Deve ficar claro por que os desenvolvedores não escolheram uma divisão específica para sub-redes: nem uma única partição da parte local do endereço funciona para todas as organizações. Algumas precisam de muitas redes com poucos hosts por rede, enquanto outras precisam de poucas redes com muitos hosts conectados a cada uma. Mais importante, nem todos os sites recebem um prefixo com 16 bits, então o padrão de sub-redes lida com casos em que um site está dividindo menos bits (por exemplo, o

site só tem uma parte local de 8 bits em seus endereços).

5.7 Sub-redes IPv4 de comprimento variável

A maioria dos sites usa sub-redes de comprimento fixo, porque são simples de entender e administrar. No entanto, o comprometimento descrito anteriormente faz com que uma sub-rede de comprimento fixo não seja atraente se um site espera ter uma mistura de redes grandes e pequenas. Quando inventaram as sub-redes, os desenvolvedores perceberam que a sub-rede de tamanho fixo não seria suficiente para todos os sites e criaram um padrão que fornece mais flexibilidade. A norma especifica que uma organização pode selecionar uma partição de sub-rede em uma base perrede. Embora a técnica seja conhecida como *sub-rede de comprimento variável*, o nome é um pouco enganador, porque a partição não varia ao longo do tempo – para uma determinada rede, uma vez que a partição foi selecionada ela nunca muda. Todos os hosts e roteadores conectados a essa rede devem seguir essa decisão; se não o fizerem, os datagramas podem ser perdidos ou extraviados. Podemos fazer o resumo a seguir.

> *Para permitir o máximo de flexibilidade na escolha de como particionar endereços sub-rede, o padrão de sub-rede TCP/IP permite sub-rede de tamanho variável, em que uma partição pode ser escolhida de forma independente para cada rede física. Uma vez que uma partição de sub-rede foi selecionada, todas as máquinas na rede devem honrá-la.*

A principal vantagem de sub-rede de comprimento variável é a flexibilidade: uma organização pode ter uma mistura de redes grandes e pequenas, e pode alcançar um maior aproveitamento do espaço de endereço. No entanto, a sub-rede de comprimento variável tem sérias desvantagens. A desvantagem mais grave surge porque o esquema pode ser difícil de administrar. A partição para cada sub-rede e os valores escolhidos para os números de sub-rede devem ser atribuídos com cuidado para evitar *ambiguidade de endereço*, uma situação em que um endereço é interpretado de forma diferente em duas redes físicas. Em particular, como o campo de rede usado em uma rede física pode ser maior do que o utilizado em outra rede, alguns dos host bits na segunda rede serão

interpretados como bits de rede na primeira. Como resultado, sub-redes de comprimento variável inválidas podem tornar impossível para todos os pares de hosts no site se comunicarem. Além disso, a ambiguidade não pode ser resolvida, a não ser por renumeração. Assim, os gerentes de rede são desencorajados de usar de sub-redes de comprimento variável.

5.8 Implementação de sub-redes IPv4 com máscaras

A tecnologia de sub-rede torna fácil a configuração de sub-redes tanto de comprimento variável como fixo. O padrão especifica que uma *máscara* de 32 bits é usada para especificar a divisão. Assim, um site usando endereçamento de sub-rede deve escolher uma *máscara de sub-rede* de 32 bits para cada rede. A máscara cobre a parte do endereço de internet, bem como a parte de local da rede física. Ou seja, bits na máscara de sub-rede são definidos para *1* se as máquinas na rede tratarem o bit correspondente no endereço IP como parte do prefixo de sub-rede e para *0* se tratarem o bit como parte do identificador de host.

Como exemplo, vejamos a seguinte máscara de sub-rede 32 bits:

11111111 11111111 11111111 00000000

Ela especifica que os três primeiros octetos identificam a rede, e o quarto octeto identifica um host nessa rede. De forma similar, a máscara

11111111 11111111 11111111 11000000

corresponde à partição ilustrada na Figura 5.5, em que a parte da rede física ocupa 10 bits.

Uma característica interessante no endereçamento sub-rede surge porque o padrão original não restringiu as máscaras de sub-rede para selecionar bits contíguos do endereço. Por exemplo, a uma rede pode ser atribuída a máscara

11111111 11111111 00011000 01000000

que seleciona os dois primeiros octetos, dois bits do terceiro octeto e um bit do quarto. Embora o padrão permita organizar atribuições interessantes de endereços, isso torna o gerenciamento de rede quase impossível. Portanto, agora é recomendado que os sites usem apenas máscaras contíguas de sub-rede.

5.9 Representação de máscara de sub-rede IPv4 e notação com barra

Especificar máscaras de sub-rede em binário é tanto desajeitado como propenso a erros. Portanto, a maioria dos softwares permite representações alternativas. Por exemplo, a maioria dos softwares permite que os gerentes usem representação decimal com ponto ao especificar máscaras de sub-rede IPv4. A notação decimal com pontos tem a vantagem de ser familiar, mas a desvantagem de tornar difícil entender padrões de bits. Ela funciona bem se é possível a um site alinhar o limite de sub-rede em limites de octeto. O exemplo na Figura 5.4b mostra como é fácil compreender sub-rede quando o terceiro octeto de endereço é utilizado para identificar uma rede física e o quarto octeto é usado para identificar um host. Em tais casos, a máscara de sub-rede tem a representação decimal com pontos *255.255.255.0*, tornando mais fácil escrever e compreender.

A literatura também contém exemplos de endereços de sub-rede e máscaras de sub-rede representados em chaves como uma 3-tupla:

{ <número rede >, <número sub-rede >, <número host> }

Nesta representação, cada seção pode ser representada em decimal com pontos, e o valor *-1* significa "all ones". Por exemplo, se a máscara de sub-rede para uma rede de classe *B* é *255.255.255.0*, pode ser escrita *{-1, -1, 0}*.

A principal vantagem é que abstrai os detalhes de campos de bits e enfatiza os valores das três partes do endereço. A principal desvantagem é que não especifica com precisão quantos bits são usados para cada parte do endereço. Por exemplo, a 3-tupla

{ 128.10, –1, 0 }

expressa um endereço com um número de rede *128.10*, todos os uns no campo de sub-rede, e todos os zeros no campo host. O endereço pode corresponder a uma sub-rede onde o limite ocorre após o terceiro octeto ou poderia corresponder a uma situação como a mostrada na Figura 5.5, em que o limite aloca 10 bits para a rede e 6 bits para o host.

Para tornar mais fácil para as pessoas expressarem e entenderem máscaras de endereço, o IETF criou uma forma sintática que é conveniente e inequívoca. Conhecida informalmente como *notação com barra (slash)*, a forma especifica como escrever uma barra seguida de um número decimal que indica o número de uns na máscara. Por exemplo, em vez de escrever o

valor decimal 255.255.255.0 pontilhado, um gerente pode escrever / 24. A Figura 5.7 lista cada valor barra possível e a decimal com pontos equivalente. A próxima seção explica como a sub-rede de comprimento variável foi generalizada e como a notação de barra é usada em todos os roteadores.

5.10 O esquema de endereçamento IPv4 Classless atual

Dissemos que o endereçamento sub-rede surgiu na tentativa de conservar o espaço de endereços IPv4. Em 1993, tornouse evidente que sub-redes por si só não impediriam, com o crescimento da Internet, que rapidamente se esgotasse o espaço de endereço. Assim começou o trabalho preliminar sobre a definição de uma versão totalmente nova do IP com endereços maiores. Para acomodar o crescimento até que a nova versão do IP pudesse ser padronizada e adotada, foi criada uma solução temporária.

Conhecido como *endereçamento classless*, o esquema de endereço temporário acaba com a classe de endereços A, B, C.* Em vez das três classes, o novo esquema estende o conceito utilizado no endereçamento de sub-rede para permitir um prefixo de rede de comprimento arbitrário. Capítulos posteriores explicam que, além de um novo modelo de endereçamento, os desenvolvedores modificaram as técnicas de encaminhamento e propagação de rota para lidar com endereços classless. Como resultado, toda a tecnologia tornou-se conhecida como *Classless Inter-Domain Routing (CIDR).*

Para compreender o impacto da CIDR, é preciso conhecer três fatos. Primeiro, o esquema classful não dividiu os endereços de rede em classes de tamanho iguais – embora menos do que dezessete mil números de classe B tenham sido criados, foram criados mais de 2 milhões de números de rede classe C. Segundo, porque os prefixos classe C são apenas suficientes para redes pequenas e não são passíveis de sub-redes, e a demanda por prefixos classe C era muito menor do que a demanda por prefixos de classe B. Em terceiro lugar, os estudos mostraram que, na taxa que os números da classe B estavam sendo atribuídos, se esgotariam rapidamente.

Notação barra	Decimal com ponto equivalente
/0	0 . 0 . 0 . 0
/ 1	128 . 0 . 0 . 0
/ 2	192 . 0 . 0 . 0
/ 3	224 . 0 . 0 . 0

/ 4	240	.	0	.	0	.	0
/ 5	248	.	0	.	0	.	0
/ 6	252	.	0	.	0	.	0
/ 7	254	.	0	.	0	.	0
/ 8	255	.	0	.	0	.	0
/ 9	255	.	128	.	0	.	0
/ 10	255	.	192	.	0	.	0
/ 11	255	.	224	.	0	.	0
/ 12	255	.	240	.	0	.	0
/ 13	255	.	248	.	0	.	0
/ 14	255	.	252	.	0	.	0
/ 15	255	.	254	.	0	.	0
/ 16	255	.	255	.	0	.	0
/ 17	255	.	255	.	128	.	0
/ 18	255	.	255	.	192	.	0
/ 19	255	.	255	.	224	.	0
/ 20	255	.	255	.	240	.	0
/ 21	255	.	255	.	248	.	0
/ 22	255	.	255	.	252	.	0
/ 23	255	.	255	.	254	.	0
/ 24	255	.	255	.	255	.	0
/ 25	255	.	255	.	255	.	128
/ 26	255	.	255	.	255	.	192
/ 27	255	.	255	.	255	.	224
/ 28	255	.	255	.	255	.	240
/ 29	255	.	255	.	255	.	248
/ 30	255	.	255	.	255	.	252
/ 31	255	.	255	.	255	.	254
/ 32	255	.	255	.	255	.	255

Figura 5.7 Máscara de endereço expressa em notação barra e a decimal com ponto equivalente para cada uma.

Um dos primeiros usos de endereçamento classless era conhecido como *supernetting*. A intenção era agrupar um conjunto de endereços de classe C contíguos para ser usado em vez de um endereço de classe B. Para entender como funciona a supernetting, considere uma organização de

médio porte que se junta à Internet. Sob o esquema classful, tal organização iria solicitar um prefixo de classe B. O esquema supernetting permite que um ISP atribua à organização um bloco de endereços de classe C em vez de um único número de classe B. O bloco deve ser grande o suficiente para numerar todas as redes da organização e (como veremos) deve situar-se em um limite que é uma potência de 2. Por exemplo, suponha que a organização espera ter 200 redes. O supernetting pode atribuir à organização um bloco de 256 números de classe C contíguos.

Embora a primeira intenção de uso do CIDR envolvesse blocos de endereços de classe C, os desenvolvedores perceberam que o CIDR poderia ser aplicado em um contexto muito mais amplo. Eles imaginaram um modelo de endereçamento hierárquico em que cada *provedor de serviços de Internet* (*ISP*) comercial poderia receber um grande bloco de endereços de Internet e, então, atribuí-los a assinantes. Como ele permite que o prefixo de rede ocorra em um limite de bit arbitrário, o CIDR permite que um ISP atribua a cada assinante um bloco de endereços apropriados às suas necessidades.

A exemplo do endereçamento de sub-rede, CIDR usa uma *máscara de endereço* de 32 bits para especificar o limite entre o prefixo e sufixo. Bits contíguos na máscara especificam o tamanho do prefixo, e bits 0 na máscara correspondem ao sufixo. À primeira vista, uma máscara de CIDR parece ser idêntica a uma máscara de sub-rede. A principal diferença é que uma máscara de CIDR não é simplesmente conhecida dentro de um site. Em vez disso, esta especifica o tamanho de um prefixo de rede, e o prefixo é conhecido em nível global. Por exemplo, suponha que a uma organização seja atribuído um bloco de 2048 endereços contíguos a partir do endereço 128.211.168.0. A tabela da Figura 5.8 lista os valores binários de endereços no intervalo.

	Decimal com ponto	Binário 32-bit equivalente
Menor	128.211.168.0	10000000 11010011 10101000 00000000
Maior	128.211.175.255	10000000 11010011 10101111 11111111

Figura 5.8 Um exemplo de bloco IPv4 CIDR que contém 2.048 endereços de host. A tabela mostra os menores e maiores endereços no intervalo expressos como decimal com ponto e valores binários.

Como 2048 é 2^{11}, onze bits são necessários para a parte do host de um endereço. Isso significa que a máscara de endereço CIDR terá 21 bits

definidos (ou seja, a divisão entre prefixo de rede e sufixo do host ocorre após o bit 21). Em binário, a máscara de endereço é:

11111111 11111111 11111000 00000000

5.11 Blocos de endereços IPv4 e notação CIDR Slash

Ao contrário dos esquemas originais classless, os endereços CIDR não são autoidentificáveis. Por exemplo, se um roteador encontra o endereço 128.211.168.1, que é um dos endereços no bloco exemplo, ele não pode saber a posição onde fica o limite, a menos que tenha informação externa. Assim, quando estiver configurando um bloco CIDR, um gerente de rede deve fornecer duas informações: o endereço de partida e uma máscara de endereço que informa quais bits estão no prefixo.

Como observado anteriormente, usar decimal binário ou decimal com ponto para uma máscara é inconveniente e propenso a erros. Portanto, CIDR especifica que um gestor deve usar a notação de barra para especificar a máscara. Sintaticamente, o formato, que às vezes é chamado *notação CIDR*, consiste de um endereço inicial em decimal com ponto seguido por um tamanho de máscara em notação barra. Assim, na notação CIDR, o bloco de endereços na Figura 5.8 é expresso:

128.211.168.0 / 21

onde */ 21* designa uma máscara de endereço com 21 bits posicionada em 1.*

5.12 Exemplo de endereçamento IPv4 Classless

A tabela da Figura 5.8 ilustra uma das principais vantagens de endereçamento classless: total flexibilidade na alocação de blocos de vários tamanhos. Ao utilizar CIDR, o ISP pode optar por atribuir a cada cliente um bloco de endereço de um tamanho adequado (isto é, o tamanho das necessidades dos clientes arredondados para a potência de 2 mais próxima). Observe que uma máscara de CIDR de *N* bits define um bloco de endereços de *32-N* endereços de host. Portanto, um bloco de endereço menor tem uma máscara mais longa. Se o provedor tem um bloco CIDR de *N* bits, pode optar por atribuir a um cliente qualquer parte de seu espaço de endereços usando uma máscara maior que *N* bits. Por exemplo, se ao ISP está atribuído 128.211.0.0 / 16, o ISP pode optar por dar a um de seus

clientes 2048 endereços na faixa / 21 que a Figura 5.8 especifica. Se o mesmo ISP também tem um cliente menor com apenas dois computadores, o ISP pode optar por atribuir outro bloco 128.211.176.212 / 30, que abrange o intervalo de endereços que a Figura 5.9 especifica.

	Decimal com ponto	Binário 32-bit equivalente
Menor	128.211.176.212	10000000 11010011 10110000 11010100
Maior	128.211.176.215	10000000 11010011 10110000 11010111

Figura 5.9 Um exemplo de bloco IPv4 CIDR, 128.211.176.212 / 30.

Uma maneira de pensar sobre endereços classless é como se cada cliente de um ISP obtivesse uma sub-rede (de tamanho variável) do bloco CIDR do ISP. Assim, um dado bloco de endereços pode ser subdividido em um limite de bit arbitrário, e um roteador no ISP pode ser configurado para encaminhar corretamente a cada subdivisão. Como resultado, ao grupo de computadores numa dada rede serão atribuídos endereços em uma faixa contígua, mas que não precisa corresponder aos limites das antigas classes A, B e C. Em vez disso, o esquema faz subdivisões flexíveis permitindo que se especifique o número exato de bits que corresponda ao prefixo. Podemos fazer o resumo a seguir.

O endereçamento IPv4 Classes, que agora é usado em toda a Internet, atribui a cada ISP um bloco CIDR e permite que o ISP particione endereços em sub-blocos contíguos, em que o menor endereço em um sub-bloco começa com uma potência de dois e o sub-bloco contém endereços em potências de dois.

5.13 Blocos IPv4 CIDR reservados para redes privadas

Como devem ser atribuídos endereços em uma intranet privada (isto é, em uma internet que não se conecta à Internet global)? Em teoria, podem ser utilizados endereços arbitrários. Por exemplo, na Internet global, o bloco de endereços IPv4 9.0.0.0 / 8 foi atribuído à IBM Corporation. Embora intranets privadas pudessem usar blocos de endereços da IBM, a experiência mostrou que isso é perigoso, porque os pacotes tendem a vazar para a Internet global e parecem vir de fontes válidas. Para evitar conflitos entre endereços usados em intranets privadas e endereços utilizados na Internet global, a IETF reserva vários prefixos de endereço e recomenda

usá-los em intranets privadas. Coletivamente, os prefixos reservados são conhecidos como *endereços privados* ou *endereços não roteáveis*. O último termo é usado porque o IETF proíbe pacotes que usam endereços privados de aparecerem na Internet global. Se um pacote contendo um dos endereços privados for acidentalmente enviado para a Internet global, um roteador irá detectar o problema e descartar o pacote.

Quando o endereçamento classless foi inventado, o conjunto de prefixos IPv4 reservados foi redefinido e ampliado. A Figura 5.10 lista os valores de endereços privados usando a notação CIDR, bem como o valor decimal com ponto dos menores e maiores endereços no bloco. O último bloco de endereços na lista, *169.254.0.0 / 16*, é incomum porque é usado por sistemas que *autoconfiguram* endereços IP. Apesar de a autoconfiguração ser raramente usada com IPv4, um capítulo posterior explica como ele tornou-se parte integrante do IPv6.

Prefixo	Menor endereço	Maior endereço
10.0.0.0 / 8	**10.0.0.0**	**10.255.255.255**
172.16.0.0 / 12	**172.16.0.0**	**172.31.255.255**
192.168.0.0 / 16	**192.168.0.0**	**192.168.255.255**
169.254.0.0 / 16	**169.254.0.0**	**169.254.255.255**

Figura 5.10 Os prefixos reservados para uso com intranets privadas não conectadas à Internet global. Se um datagrama enviado para um desses endereços atingir acidentalmente a Internet, ocorrerá um erro.

5.14 O esquema de endereçamento IPv6

Dissemos que cada endereço IPv6 ocupa 128 bits (16 octetos). O espaço de endereço grande garante que o IPv6 possa tolerar qualquer esquema de atribuição de endereço razoável. Na verdade, se a comunidade decidir mudar o esquema de endereçamento mais tarde, o espaço de endereçamento é suficientemente grande para acomodar uma mudança.

É difícil compreender o tamanho de espaço do endereçamento IPv6. Uma maneira de olhar para ele relaciona a magnitude ao tamanho da população: o espaço de endereços é tão grande que cada pessoa no planeta pode ter endereços suficientes para ter a sua própria internet três vezes tão grande quanto a Internet atual. Uma segunda maneira de pensar no

endereçamento IPv6 é relacioná-lo com o espaço físico disponível: a superfície da Terra tem cerca de $5{,}1 \times 10^8$ quilômetros quadrados, o que significa que há mais de 10^{24} endereços por metro quadrado. Outra forma de entender a dimensão é relacioná-la à exaustão de endereços. Por exemplo, considere o tempo que seria necessário para atribuir todos os endereços possíveis. Um inteiro com 16 octetos pode conter 2^{128} valores. Assim, o espaço de endereço é maior do que $3{,}4 \times 10^{38}$. Se os endereços fossem atribuídos à taxa de um milhão de endereços a cada microssegundo, levaria mais de 10^{20} anos para atribuir todos os endereços possíveis.

5.15 Notação Hexadecimal IPv6 Colon (dois-pontos)

Embora ela resolva o problema de ter capacidade insuficiente, o maior tamanho de endereço apresenta um interessante novo problema: as pessoas que gerenciam a Internet devem ler, inserir e manipular tais endereços. Obviamente, notação binária é indefensável. A notação decimal com pontos usada para IPv4 também não faz endereços IPv6 suficientemente compactos. Para entender o porquê, considere um exemplo de um número de 128 bits expresso em notação decimal com pontos:

104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255

Para ajudar a tornar os endereços ligeiramente mais compactos e mais fáceis de adicionar, os desenvolvedores do IPv6 criaram a notação hexadecimal colon (*colon hexadecimal notation* – abreviada como *hex colon*), em que o valor de cada quantidade de 16 bits é representado em hexadecimal separado por *colon*s. Por exemplo, quando o valor mostrado anteriormente em notação decimal é traduzido para a notação colon hex e impresso usando o mesmo espaçamento, torna-se:

68E6:8C64:FFFF:FFFF:0:1180:96A:FFFF

A notação colon hex tem a vantagem óbvia de requerer menos dígitos e menos caracteres de separação do que a decimal com ponto. Além disso, a notação colon hex inclui duas técnicas que a tornam extremamente úteis. Em primeiro lugar, a notação colon hex permite a *compressão de zeros*, em que uma série de zeros repetidos é substituída por um par de dois-pontos. Por exemplo, o endereço:

FF05:0:0:0:0:0:0:B3

pode ser escrito:

FF05::B3

Para garantir que a compressão de zeros produza uma interpretação inequívoca, as normas especificam que ela pode ser aplicada a qualquer endereço apenas uma vez. A compressão de zeros é especialmente útil porque as atribuições IPv6 criarão muitos endereços que contêm sequências de zeros contíguas. Em segundo lugar, a notação hexadecimal colon incorpora sufixos decimais com ponto; tais combinações se destinam a serem usadas durante a transição do IPv4 para o IPv6. Por exemplo, a seguinte sequência de caracteres é uma notação colon hex válida:

0:0:0:0:0:0:128.10.2.1

Note-se que, embora os números separados por colons especifiquem cada um o valor de uma quantidade de 16 bits, os números na parte decimal com ponto especificam cada um o valor de um octeto. Claro que a compressão de zeros pode ser utilizada com o número anterior para produzir uma cadeia de caracteres hex colon equivalente que se assemelha bastante a um endereço IPv4:

::128.10.2.1

Finalmente, o IPv6 estende a notação tipo CIDR, permitindo que um endereço possa ser seguido por uma barra e um inteiro que especificam um número de bits. Por exemplo,

12AB::CD30:0:0:0:0/60

especifica os primeiros 60 bits do endereço, que é 12AB00000000CD3 em hexadecimal.

5.16 Endereço IPv6 atribuição de espaço

A questão sobre como particionar o espaço de endereço IPv6 tem gerado muita discussão. Há duas questões centrais: como as pessoas gerenciam a atribuição de endereços e como os roteadores lidam com as tabelas de encaminhamento necessárias. A primeira questão centra-se no problema prático de elaboração de uma hierarquia de autoridade. Ao contrário da Internet atual, que usa uma hierarquia de dois níveis de prefixo de rede (atribuído por um ISP) e sufixo de host (atribuído por uma organização), o grande espaço de endereço no IPv6 permite uma hierarquia de vários níveis

ou várias hierarquias. Grandes ISPs podem começar com grandes blocos de endereços e atribuírem sub-blocos para ISPs de segundo nível, que pode, cada uma, atribuir sub-blocos da sua atribuição para ISPs de terceiro nível, e assim por diante. A segunda questão se concentra na eficiência de roteador, e será explicada mais tarde. Por enquanto, é suficiente entender que um roteador deve examinar cada datagrama, por isso a escolha de atribuição pode afetar a forma como os roteadores lidam com encaminhamento.

O espaço de endereço IPv6 foi dividido em blocos de endereços análogos ao esquema classful original utilizado com IPv4. Os primeiros 8 bits de um endereço são suficientes para identificar os tipos básicos. Como o endereçamento IPv4 classful, o IPv6 não particiona o espaço de endereço em seções de tamanhos iguais. A Figura 5.11 lista os prefixos IPv6 e seus significados.

Prefixo binário	Tipo de endereço	Fração do espaço de endereço
0000 0000	Reserved (IPv4 compatibility)	1/256
0000 0001	Unassigned	1/256
0000 001	NSAP Addresses	1/128
0000 01	Unassigned	1/64
0000 1	Unassigned	1/32
0001	Unassigned	1/16
001	Global Unicast	1/8
010	Unassigned	1/8
011	Unassigned	1/8
100	Unassigned	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1/16
1111 0	Unassigned	1/32
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
1111 1110 0	Unassigned	1/512
1111 1110 10	Link-Local Unicast Addresses	1/1024
1111 1110 11	IANA - Reserved	1/1024
1111 1111	Multicast Addresses	1/256

Figura 5.11 Prefixos usados para dividir o espaço de endereço IPv6 em blocos e propósito de cada bloco.

Como mostra a figura, apenas 15% do espaço de endereço foram atribuídos. O IETF irá utilizar as partes restantes conforme a demanda crescer. Apesar da atribuição escassa, os endereços foram escolhidos para fazer um processamento mais eficiente. Por exemplo, o octeto de alta ordem de um endereço distingue entre multicast (todos os bits 1) e unicast (uma

mistura de zeros e uns).

5.17 Endereços IPv4 embutidos em IPv6 para a transição

Para permitir a transição do IPv4 para o IPv6, os desenvolvedores têm alocado uma pequena fração dos endereços no espaço IPv6 para codificar endereços IPv4. Por exemplo, qualquer endereço que começa com 80 bits zero, seguidos por 16 bits todos um, contém um endereço IPv4 na ordem baixa 32 bits. Além disso, um conjunto de endereços é reservado para utilização com o *protocolo de tradução State-less IP/ICMP (SIIT)*. A Figura 5.12 ilustra as duas formas.

Um endereço IPv4 será embutido em um endereço IPv6 durante a transição do IPv4 para o IPv6 por duas razões. Em primeiro lugar, um computador pode optar por atualizar-se do software IPv4 para o IPv6 antes de ter sido atribuído um endereço IPv6 válido. Em segundo lugar, um computador executando um software IPv6 pode precisar se comunicar com um computador que executa apenas o software IPv4.

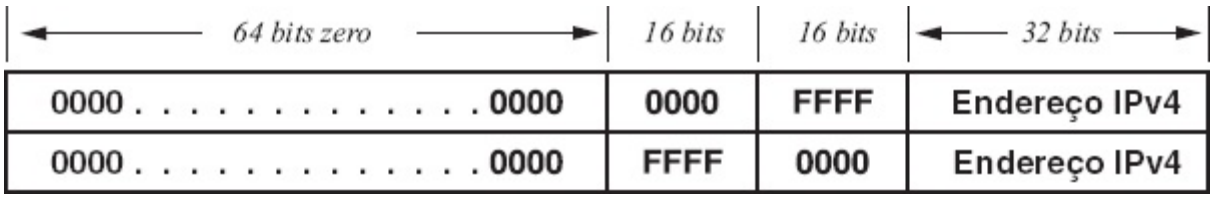


Figura 5.12 Duas formas de incorporar um endereço IPv4 em um endereço IPv6. A segunda forma é usada para tradução Stateless IP/ICMP.

Ter uma maneira de inserir um endereço IPv4 em um endereço IPv6 não resolve o problema de fazer as duas versões interoperarem. Além disso, para o endereço incorporado, é necessário traduzir o pacote para converter entre os formatos de pacotes IPv4 e IPv6. Vamos entender a conversão depois que capítulos posteriores explicarem os dois formatos de pacotes.

Pode parecer que traduzir os endereços de protocolo poderia falhar, pois os protocolos da camada superior verificam a integridade do endereço. Em particular, veremos que os *checksum computations* TCP e UDP usam um *pseudocabeçalho* que inclui os endereços IP de origem e destino. Como resultado, parece que traduzir um endereço invalidaria o checksum. No entanto, os desenvolvedores planejaram cuidadosamente para permitir que o TCP ou UDP em uma máquina IPv4 se comunicasse com o protocolo de transporte correspondente em uma máquina IPv6. Para evitar erros dos

checksum, a codificação IPv6 de um endereço IPv4 foi escolhida de modo que o de 16-bit complemente o checksum para ambos, um endereço IPv4 e um para a versão incorporada IPv6 do endereço de forma que sejam idênticos. O ponto importante é descrito a seguir.

> *Além de escolher os detalhes técnicos de um novo protocolo de Internet, o trabalho da IETF em IPv6 se concentrou em encontrar uma maneira de fazer a transição do protocolo atual para o novo protocolo. Em particular, o IPv6 fornece uma maneira de inserir um endereço IPv4 em um endereço IPv6 de modo que a mudança entre as duas formas não afete o pseudocabeçalho checksum usado por protocolos de transporte.*

5.18 Os endereços IPv6 unicast e /64

O esquema de IPv6 para a atribuir a cada computador host um endereço estende o esquema IPv4. Em vez de dividir um endereço em duas partes (uma ID de rede e uma ID de host), um endereço IPv6 é dividido em três partes conceituais: um prefixo global exclusivo usado para identificar um site, um ID usado para distinguir entre múltiplas redes físicas no site de destino e uma interface ID usada para identificar um computador particular conectado à sub-rede. A Figura 5.13 ilustra o particionamento.

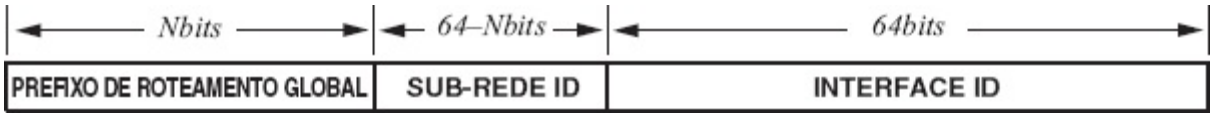


Figura 5.13 A divisão de um endereço unicast IPv6 em três partes conceituais. A interface ID sempre ocupa 64 bits.

Note que a hierarquia de três níveis formaliza a ideia de endereçamento de sub-rede do IPv4. Ao contrário de sub-redes, no entanto, a estrutura de endereço IPv6 não se restringe a um único site local. Em vez disso, a estrutura de endereço é reconhecida globalmente.

5.19 Identificadores de interface IPv6 e endereços MAC

O IPv6 usa o termo identificador de interface (*interface de ID*) em vez de

identificador de host para enfatizar que um host pode ter várias interfaces e vários IDs. Como mostra a próxima seção, IPv4 e IPv6 compartilham o conceito; apenas a terminologia é diferente.

Na Figura 5.13, os 64 bits de baixa ordem de um endereço IPv6 unicast identificam uma interface de rede específica. O sufixo IPv6 foi escolhido para ser grande o suficiente para permitir que um endereço de hardware (MAC) seja usado como o ID único. Como veremos mais tarde, a incorporação de um endereço de hardware em um endereço IPv6 torna trivial encontrar o endereço de hardware de um computador. Claro que, para garantir a interoperabilidade, todos os computadores em uma rede devem usar a mesma representação de um endereço de hardware. Consequentemente, os padrões IPv6 especificam exatamente como representar diversas formas de endereços de hardware. No caso mais simples, o endereço de hardware é colocado diretamente nos bits de baixa ordem de um endereço IPv6; alguns formatos usam transformações mais complexas.

Dois exemplos ajudarão a esclarecer o conceito. O IEEE define globalmente um formato-padrão exclusivo de endereço MAC de 64 bits conhecido como *EUI-64*. A única mudança necessária para utilizar um endereço EUI-64 em um endereço IPv6 consiste em inverter o bit 6 no octeto de alta ordem do endereço. O bit 6 indica se o endereço é conhecido como globalmente único. É necessária uma mudança mais complexa para um endereço Ethernet convencional de 48 bits, como ilustra a Figura 5.14.

Como mostra a figura, bits do endereço MAC original não são contíguos em um endereço IPv6. Em vez disso, são inseridos no meio 16 bits com valor hexadecimal $FFFE_{16}$. Além disso, o bit *6*, que indica se o endereço tem escopo global, é alterado de *0* para *1*. Os bits restantes do endereço, incluindo o grupo de bits (rotulados *g*), o ID da empresa que fabricou a interface (rotulado *c*) e a extensão do fabricante, são copiados como mostrado.

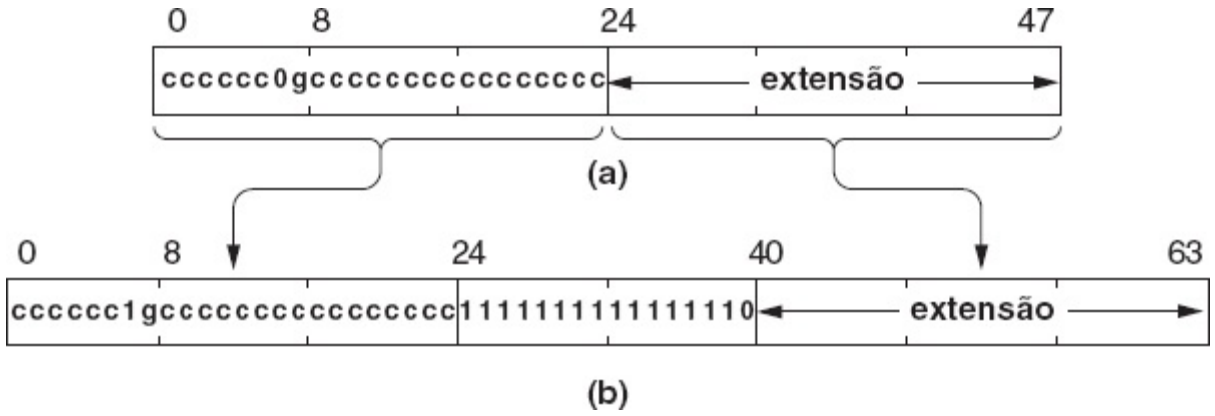


Figura 5.14 (a) O formato de um endereço Ethernet de 48 bits, com bits que identificam um fabricante e extensão, e (b) o endereço, quando colocado em um endereço unicast IPv6 64 bits de baixa ordem.

5.20 Endereços IP, hosts e conexões de rede

Para simplificar a discussão do início do capítulo, dissemos que um endereço IPv4 identifica um host. No entanto, a descrição não é exata. Considere um roteador que se conecta a duas redes físicas. Como podemos atribuir um único endereço IP se cada endereço inclui um identificador de rede, bem como um identificador de host? Na verdade, não podemos. Uma situação similar acontece para um computador convencional, que tem duas ou mais conexões de rede física (tais computadores são conhecidos como *multi-homed hosts*). A cada uma das conexões de rede do computador deve ser atribuído um endereço que identifica uma rede. A ideia a seguir é fundamental tanto em endereçamento IPv4 como IPv6.

> *Como um endereço IP identifica uma rede, bem como um host na rede, um endereço não especifica um computador individual. Em vez disso, um endereço identifica uma ligação a uma rede.*

Um roteador que se conecta a n redes tem n endereços IP distintos; um para cada conexão de rede. O IPv6 faz uma distinção clara, utilizando o termo *endereço de interface* (isto é, um endereço é atribuído para a interface de um computador a uma rede). Para o IPv6, a situação é ainda mais complexa do que várias conexões de rede: para lidar com a migração de um provedor para outro, o IPv6 especifica que uma determinada interface pode ter vários endereços ao mesmo tempo. Por enquanto, só

precisamos ter em mente que cada endereço especifica uma conexão de rede. O Capítulo 18 vai considerar a questão mais a fundo.

5.21 Endereços especiais

Tanto o IPv4 como o IPv6 têm interpretações especiais para alguns endereços. Por exemplo, um endereço internet pode se referir a uma rede assim como a um host. A próxima seção descreve como as duas versões lidam com endereços especiais.

5.21.1 Endereço de rede IPv4

Por convenção, um host IPv4 ID *0* nunca é atribuído a um host individual. Em vez disso, é usado um endereço IPv4 com zero na parte que se refere à rede propriamente dita.

> *Um endereço IPv4 que tem um host ID* zero *é reservado para se referir à rede.*

5.21.2 Endereço IPv4 broadcast direcionado

O IPv4 inclui um *endereço de broadcast direcionado* que às vezes é chamado de um *endereço de rede broadcast*. Quando usado como um endereço de destino, refere-se a todos os computadores em uma rede. A norma especifica que um hostid com tudo *um* é reservado para broadcast direcionado.*

Quando um pacote é enviado para tal endereço, uma única cópia é transferida através da internet da origem para o destino. Roteadores ao longo do caminho usam a parte de rede do endereço, sem olhar para a parte de host. Uma vez que o pacote chega a um roteador conectado à rede final, ele examina a parte do host do endereço e, se encontrar tudo com *um*, transmite o pacote para todas as máquinas da rede.

Em algumas tecnologias de rede (por exemplo, Ethernet), o hardware subjacente suporta broadcasting. Em outras, o software implementa a transmissão através do envio de uma cópia individual de cada host na rede. O ponto é que ter um endereço IP broadcast dirigido não garante que a entrega será eficiente. Podemos fazer o resumo a seguir.

O IPv4 suporta broadcast direcionado em que um pacote é enviado para todos os computadores em uma rede específica; se disponível, um hardware de transmissão é usado. Um endereço de broadcast direcionado tem uma porção de rede válida e uma parte hostid com tudo um.

Endereços broadcast direcionados fornecem um mecanismo poderoso e perigoso, pois um remetente arbitrário pode transmitir um único pacote que será transmitido na rede especificada. Para evitar possíveis problemas, muitos sites configuram os roteadores para rejeitarem todos os pacotes de transmissão dirigida.

5.21.3 Endereço IPv4 broadcast limitado (rede local)

Além de endereços broadcast específicos de rede descritos acima, o IPv4 suporta *broadcasting limitado*, às vezes chamado de broadcast em rede local. A difusão limitada significa que *um* pacote é transmitido através da rede local. O endereço de broadcast local é composto por trinta e dois números um (daí, é às vezes chamado de endereço broadcast "tudo *1*"). Como veremos, um host pode usar o endereço de broadcast limitado na inicialização antes que seja atribuído ao host seu endereço IP ou o endereço IP da rede. Uma vez que o host descobre o endereço IP correto para a rede local, o broadcast direcionado é o preferido.

Podemos então fazer o resumo a seguir.

Um endereço broadcast limitado IPv4 é composto por trinta e dois bits 1. Um pacote enviado para o endereço broadcast limitado será transmitido através da rede local e pode ser usado durante a inicialização antes que o computador tenha seu endereço IP atribuído.

5.21.4 Endereço IPv4 broadcast de sub-rede

Se um site usa sub-rede, o IPv4 define um *endereço sub-rede broadcast* correspondente. Um endereço sub-rede broadcast consiste em um prefixo de rede, um número de sub-rede e tudo um no campo host.

Um endereço sub-rede broadcast é usado para transmissão em uma rede única, em um site que usa sub-rede. O endereço contém um prefixo de rede e de sub-rede e tem tudo 1 no campo host.

5.21.5 Endereço IPv4 com fonte tudo zero (All-0s Source Address)

Um endereço que consiste em trinta e dois bits com zero é reservado para os casos em que um host precisa se comunicar, mas ainda não conhece seu endereço IP (ou seja, na inicialização). Em particular, veremos que, para obter um endereço IP, um host envia um datagrama para o endereço de broadcast limitado e usa o endereço *0* para se identificar. O receptor entende que o host ainda não tem um endereço IP e usa um método especial para enviar uma resposta.

No IPv4, um endereço com trinta e dois bits com zero é usado como um endereço temporário na inicialização antes de o host aprender seu endereço IP.

5.21.6 Endereços de IPv4 multicast

Além de *entrega unicast*, em que se entrega um pacote a um único computador, e *entrega broadcast*, em que um pacote é entregue a todos os computadores de uma determinada rede, o esquema de endereçamento *IPv4* suporta uma forma especial de entrega multi, conhecida como *multicasting*, na qual um pacote é entregue a um subconjunto específico de hosts. O Capítulo 15 discute endereçamento e entrega multicast em detalhes. Por enquanto, basta entender que qualquer endereço IPv4 começando com três bits 1 é usado para multicasting.

5.21.7 Endereço IPv4 loopback

O prefixo de rede 127.0.0.0/8 (um valor da faixa de classe original A) é reservado para *loopback* e destina-se para uso em testes de TCP/IP e para comunicação entre processos no computador local. Por convenção, os programadores usam 127.0.0.1 para testes, mas qualquer valor de host pode

ser usado porque o software TCP/IP não examina a parte do host.

Quando um aplicativo envia um pacote para um endereço 127, o software de protocolo no computador aceita o pacote de saída e imediatamente alimenta o pacote de volta para o módulo que lida com pacotes de entrada, como se o pacote tivesse chegado. O loopback é restrito a um sistema operativo local; um pacote com um endereço 127 nunca deve aparecer na Internet.

O IPv4 reserva 127.0.0.0 / 8 para testes de loopback; um pacote destinado a qualquer host com prefixo 127 fica dentro do computador e não viaja através de uma rede.

5.21.8 Resumo das convenções de endereço especial do IPv4

A Figura 5.15 resume os endereços especiais usados no IPv4. Como as notas da figura fazem menção, o endereço com tudo *zero* nunca é usado como um destino, e só pode ser utilizado como um endereço de origem durante a inicialização. Uma vez que o computador aprende seu endereço IP, a máquina não deve usar tudo *zero* como fonte.

5.21.9 Endereços IPv6 multicast e anycast

Em teoria, a escolha entre multicast e broadcast é irrelevante porque uma pode ser simulada com a outra. Ou seja, multicast e broadcast são duais uma da outra que fornecem a mesma funcionalidade. Para entender por que motivo, considere como simular uma com a outra. Se o broadcast estiver disponível, um pacote pode ser entregue a um grupo, por broadcasting a todas as máquinas e organizar por software em cada máquina para decidir se aceita ou descarta o pacote de entrada. Se o multicast está disponível, um pacote pode ser entregue a todas as máquinas, organizando para que todas as máquinas ouçam o *all nodes* do grupo multicast.

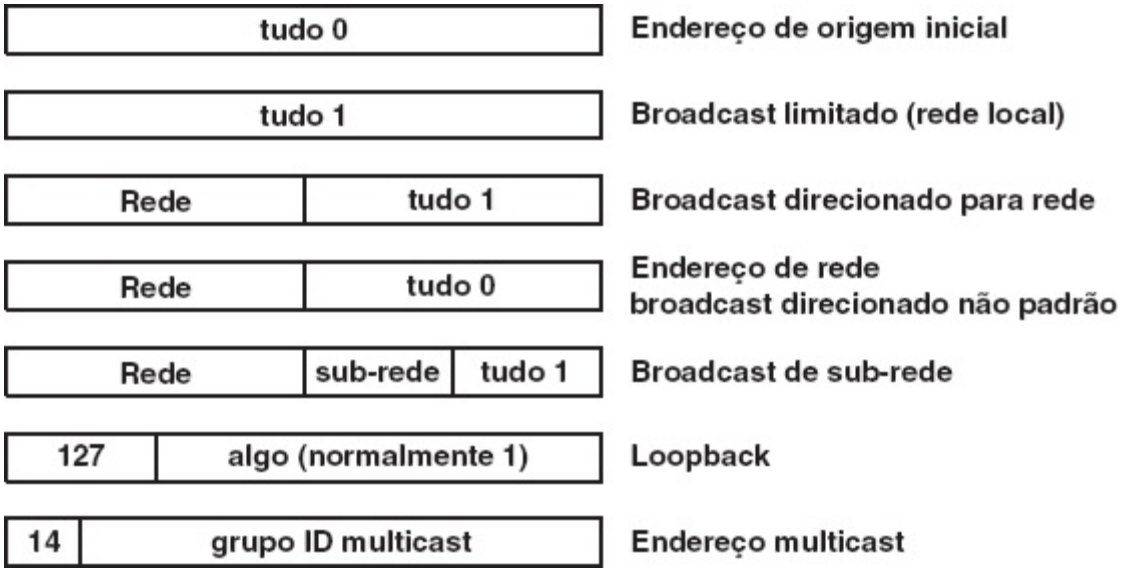


Figura 5.15 Sumário de endereços especiais IPv4.

Sabendo que broadcasting e multicasting são duais teóricos um do outro e que isso não ajuda escolher entre eles, desenvolvedores IPv6 decidiram evitar o broadcast e usar apenas o multicast. Portanto, o IPv6 define vários conjuntos reservados de grupos multicast. Por exemplo, se um host IPv6 quer transmitir um pacote que irá atingir os roteadores na rede local, o host envia o pacote para *todos os roteadores* do grupo multicast. O IPv6 também define um grupo multicast *all hosts* (o pacote é entregue a todos os hosts da rede local) e um grupo multicast *all nodes* (o pacote é entregue a todos os hosts e todos os roteadores).

Para entender por que os desenvolvedores do IPv6 escolheram multicasting como a abstração central, em vez de broadcast, considere os pedidos em vez de olhar para o hardware subjacente. Uma aplicação precisa se comunicar com uma única aplicação assim como com um grupo de aplicações. A comunicação direta é tratada melhor via unicast; comunicação em grupo pode ser tratada tanto por multicast como por broadcast. Na Internet, um grupo de membros não está relacionado com (ou restrito a) uma única rede – os membros do grupo podem residir em locais arbitrários. Usando broadcast para todo o grupo, as comunicações não se adaptam a toda a Internet global, de modo que o multicast é a única opção. Ironicamente, até mesmo os esforços para implementar o multicast na Internet global falharam até agora. Assim, pouco tem sido feito pelo IPv6 multicast.

Além de multicast, o IPv6 introduz um novo tipo de endereço conhecido

como endereço *anycast*.* O endereçamento anycast é projetado para lidar com a replicação do servidor. Um provedor pode implementar um conjunto de servidores idênticos em locais arbitrários na Internet. Todos os servidores no conjunto devem oferecer exatamente o mesmo serviço, e todos recebem o mesmo endereço anycast. O encaminhamento está configurado de forma que um pacote enviado para o endereço anycast vai para o servidor mais próximo.

5.21.10 Endereços IPv6 link-local

O IPv6 define um conjunto de prefixos para endereços unicast que não são globalmente válidos. Em vez disso, os prefixos são ditos ser *de escopo local* ou ter *escopo link-local*, isto é, os pacotes enviados para os endereços são restritos para viajar através de uma rede única. A norma define qualquer endereço IPv6 que comece com o prefixo binário de 10 bits

```
1111 1110 10
```

como um endereço link-local. Por exemplo, quando um computador é inicializado, ele forma um endereço IPv6 combinando o prefixo link-local com um endereço de interface MAC, como descrito anteriormente.

Roteadores honram as regras de escopo link-local. Um roteador pode responder a um pacote de link-local enviado através de uma rede local, mas nunca encaminha um pacote que contenha um endereço link-local fora do escopo especificado (ou seja, nunca fora da rede local).

Endereços link-local fornecem uma maneira para um computador conversar com os seus vizinhos (por exemplo, durante a inicialização) sem perigo de pacotes serem encaminhados através da Internet. Vamos ver, por exemplo, que um nó IPv6 usa um endereço link-local na inicialização, incluindo o endereço de um roteador para descobrir seus vizinhos. Computadores conectados a uma *rede isolada* (isto é, uma rede que não tem roteadores conectados) podem usar os endereços link-local para se comunicarem.

5.22 Pontos fracos no endereçamento Internet

Embutir informações de rede em um endereço internet tem algumas desvantagens. A desvantagem mais óbvia é que esses endereços se referem a conexões de rede, não ao computador host.

Se um host passar de uma rede para outra, seu endereço internet deve mudar.

O IPv6 tenta aliviar o problema, tornando-o mais fácil de alterar um endereço. No entanto, o problema básico permanece. O Capítulo 18 discute como o esquema de endereçamento torna difícil a *mobilidade*.

A fraqueza do sistema IPv4 surge da ligação inicial – uma vez que o tamanho do prefixo é escolhido, o número máximo de hosts da rede está fixado. Se a rede cresce além do limite original, um novo prefixo deve ser selecionado e todos os hosts da rede devem ser renumerados. Enquanto a *renumeração* pode parecer um problema menor, alterar os endereços de rede pode ser extremamente demorado e difícil de depurar. O IPv6 resolve o problema do crescimento da rede por meio da atribuição de um número absurdo de bits (64) para um sufixo que identifica um host (ou para ser mais preciso, um interface de rede).

A falha mais importante no esquema de endereçamento internet se tornará aparente quando analisarmos encaminhamento. Porém, sua importância garante uma rápida introdução aqui. Sugerimos que o encaminhamento será baseado em endereços Internet de destino. Especificamente, um roteador usará um prefixo do endereço que identificar como rede de destino. Considere um host com duas conexões de rede. Sabemos que tal host deve ter dois endereços IP, um para cada interface. A afirmação a seguir é verdadeira.

Como o encaminhamento utiliza a parte de rede do endereço IP, o caminho tomado pelos pacotes que trafegam para um host com vários endereços IP depende do endereço usado.

As implicações são surpreendentes. Os humanos pensam em cada host como uma única entidade e querem usar um único nome. Eles normalmente ficam surpresos ao ver que precisam descobrir mais de um nome, e ainda mais surpresos ao ver que os pacotes enviados usando vário s nomes podem se comportar de formas diferentes.

Outra consequência surpreendente do esquema de endereçamento internet é que simplesmente conhecer um endereço IP para um destino

pode não ser suficiente. Se uma rede está desconectada, pode ser impossível atingir o destino usando um endereço específico.

Para entender, considere o exemplo na Figura 5.16.

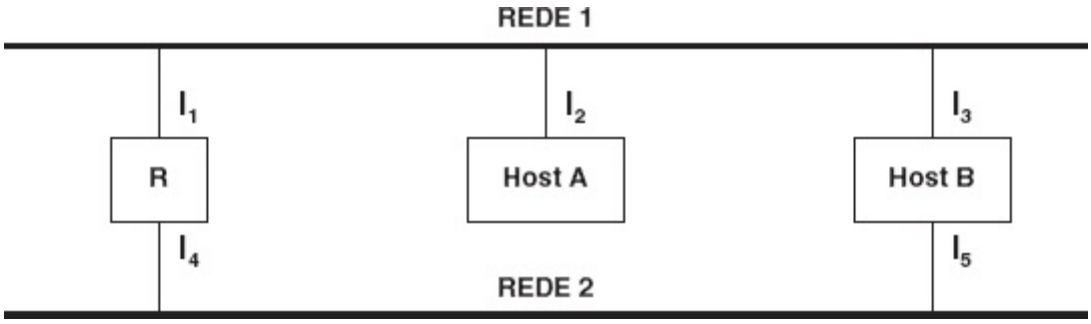


Figura 5.16 Um exemplo de duas redes com conexões a um roteador R, host, A, e um host multi-homes, B.

Na figura, hosts *A* e *B* se conectam à rede 1. Então, normalmente esperaríamos que *A* transmitisse para o endereço de *B* nessa rede. Suponha, entretanto, que a conexão de *B* com a rede 1 falhe (ou seja, que a interface I_3 se desconecte). Se A tentar usar o endereço de *B* na rede 1 (ou seja, o endereço para a interface I_3), *A* irá concluir que *B* está desconectado, pois nenhum pacote seguiu. Surpreendentemente, se *A* enviar ao endereço de interface I_5, pacotes serão encaminhados através do roteador *R* e chagarão a *B*. Ou seja, existe um caminho alternativo de *A* para *B*, mas este não será usado a menos que esse endereço alternativo esteja especificado. Vamos discutir o problema em capítulos posteriores, quando considerarmos o encaminhamento e o vínculo de nome.

5.23 Atribuição de endereço Internet e delegação de autoridade

Cada prefixo de rede usado na Internet global deve ser exclusivo. Para garantir unicidade, todos os prefixos são atribuídos por uma autoridade central. Originalmente, a *Internet Assigned Number Authority* (*IANA*) tinha controle sobre os números atribuídos e definia a política. Desde a época em que a Internet iniciou até ao primavera de 1998, um único indivíduo, Jon Postel, dirigia a IANA e os endereços atribuídos. No final de

1998, depois da morte prematura de Jon, uma nova organização foi criada para tratar da atribuição de endereços. Com o nome *Internet Corporation For Assigned Names* and Numbers (*ICANN*), a organização define a política e atribui valores para nomes e outras constantes usadas em protocolos e também endereços.

A maior parte dos sites que precisam de um prefixo Internet nunca interage com a autoridade central diretamente. Em vez disso, uma organização normalmente contrata com um Provedor de Acesso a Internet (*Internet Service Provider-ISP*) local. Além de fornecer uma conexão física, um ISP obtém um prefixo de endereço válido para cada uma das redes do cliente. Na verdade, muitos ISP locais são clientes de ISP maiores – quando um cliente requisita um prefixo de endereço, o ISP local simplesmente obtém um prefixo a partir de um ISP maior. Assim, somente os ISP maiores precisam contatar um dos registradores locais de endereço que o ICANN autorizou para administrar blocos de endereços (*ARIN*, *RIPE*, *APNIC*, *LATNIC* ou *AFRINIC*).

Observe como a delegação de autoridade repassa a hierarquia ISP quando endereços são atribuídos. Fornecendo um bloco de endereços a um registrado regional, a ICANN delega autoridade para sua atribuição. Quando fornece um sub-bloco par um ISP maior, um registrado delega autoridade para atribuição. No nível mais baixo, quando um ISP fornece parte de suas alocações para uma organização, concede autoridade à organização para subdividir a alocação dentro dela. O ponto é que, quando um bloco de endereços é alocado dentro da hierarquia, o recipiente recebe autoridade para subdividir ainda mais o bloco.

5.24 Um exemplo de atribuição de endereços IPv4

Para esclarecer o esquema de endereçamento IPv4, considere um exemplo de duas redes em um site. A Figura 5.17 mostra a arquitetura conceitual: duas redes conectadas a um ISP. O exemplo mostra três redes e os números de rede classless que lhes foram atribuídos: à rede do Internet Service Provider foi atribuído (9.0.0.0/8), a uma Ethernet no site foi atribuído (128.10.0.0/16) e a uma rede Wi-Fi, (128.210.0.0/16).

A Figura 5.18 ilustra as mesmas redes com os computadores host conectados a elas e os endereços da Internet atribuídos a cada conexão de rede. A figura mostra três hosts, rotulados *Merlin*, *Lancelot* e *Guenevere*. A figura também mostra dois roteadores: R_1, que conecta as redes Ethernet

e Wi-Fi, e R_2, que conecta o site a um ISP.

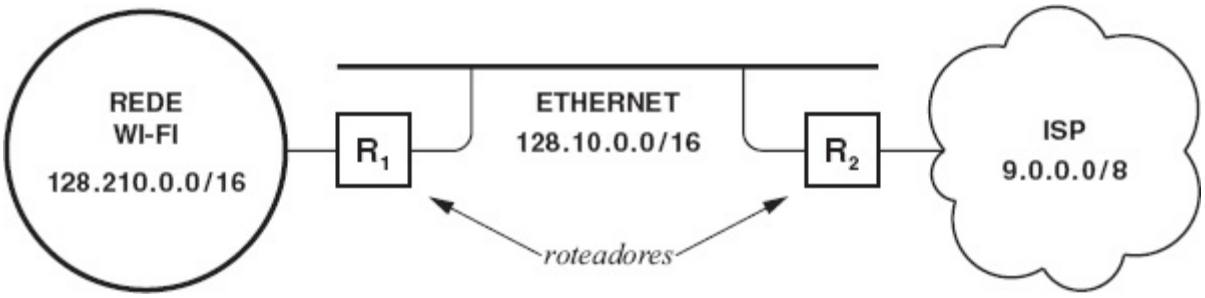


Figura 5.17 Exemplo de arquitetura de um site com prefixos de endereço IPv4 atribuídos às duas redes.

O host *Merlin* possui conexões com a rede Ethernet e Wi-Fi, de modo que pode alcançar destinos em qualquer uma das redes diretamente. A distinção entre um roteador (por exemplo, R_1) e um host multi-homed (por exemplo, *Merlin*) surge na configuração: um roteador é configurado para encaminhar pacotes entre as duas redes, enquanto um host pode usar qualquer rede, mas não encaminha pacotes.

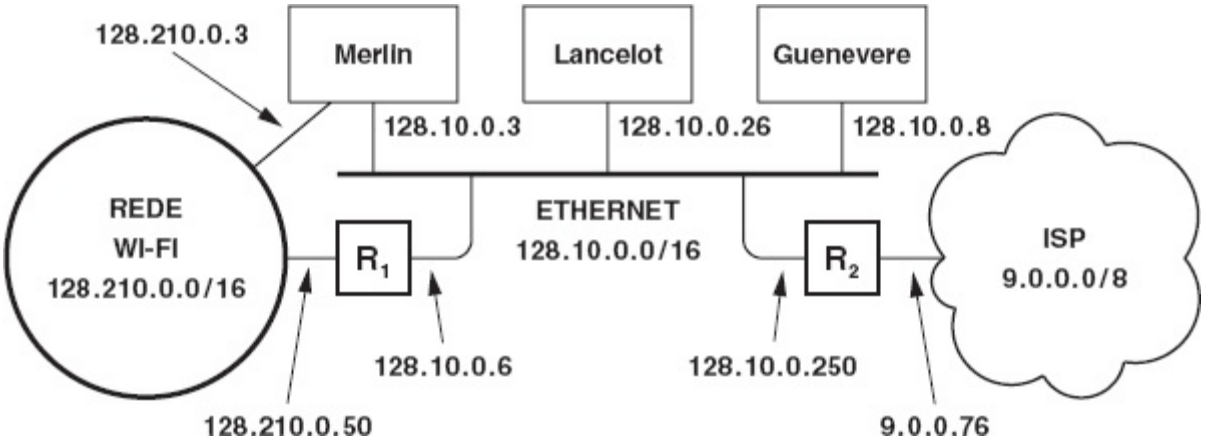


Figura 5.18 A rede do exemplo da Figura 5.17 com endereços IPv4 atribuídos a ambos, os host e os roteadores.

Como vemos na figura, um endereço IP precisa ser atribuído a cada interface de rede. *Lancelot*, que se conecta apenas à Ethernet, recebeu 128.10.2.26 como seu único endereço IP. Por ser um dual-homed host, *Merlin* tem o endereço 128.10.0.3 para sua conexão com a Ethernet e

128.210.0.3 para sua conexão com a rede Wi-Fi; quem fez a atribuição de endereços escolheu o mesmo valor para o número host em cada endereço. O roteador R1 também tem dois endereços, 128.10.0.6 e 128.210.0.50. Note que as partes host dos dois endereços são não relacionadas. Os protocolos IP não se importam se qualquer um dos bytes na forma decimal com ponto dos endereços de um computador for igual ou diferente. Porém, os técnicos de rede, gerentes e administradores podem ter de usar endereços para manutenção, teste e depuração. A escolha de fazer com que todos os endereços de uma máquina terminem com o mesmo valor facilita que as pessoas se lembrem ou adivinhem o endereço de uma interface em particular.

5.25 Resumo

A cada computador em uma internet TCP/IP é atribuído um endereço binário único chamado endereço de Protocolo de Internet ou endereço IP. O IPv4 usa endereços de 32 bits, que são divididos em duas partes principais: um prefixo, que identifica a rede à qual o computador se conecta, e um sufixo, que fornece um identificador exclusivo para um computador na rede. O esquema de endereçamento IPv4 original é conhecido como classful; um prefixo pertence a uma das três classes primárias. Variações posteriores estenderam o mecanismo de endereçamento IPv4 com endereçamento sub-rede e endereçamento classless. Endereçamento IPv4 classless usa uma máscara de bits para especificar quantos bits correspondem a um prefixo.

Para tornar os endereços mais fáceis para humanos compreenderem, foram inventadas formas sintáticas. Os endereços IPv4 são escritos em notação decimal com pontos, em que cada octeto é escrito em decimal, com os valores separados por pontos decimais. Os endereços IPv6 são escritos em notação colon hex, com octetos representados em hexadecimal separado por dois pontos.

Endereços IP referem-se a conexões de rede em vez de hosts individuais. Portanto, um roteador ou multi-homed host tem vários endereços IP.

O IPv4 e o IPv6 incluem endereços especiais. O IPv4 permite rede específica, sub-rede específica e broadcast local específico, bem como multicast. O IPv6 tem endereços link-local e anycast, bem como multicast. Um conjunto de prefixos IPv4 foi reservado para uso em intranets privadas.

EXERCÍCIOS

5.1 Quantas redes de classe *A*, *B* e *C* podem existir? Quantos hosts uma rede pode ter em cada classe? Lembre-se de permitir o broadcast e também os endereços de classes *D* e *E*.

5.2 Se o seu site usa IPv4, descubra qual o tamanho da máscara de endereço usada. Quantos hosts ela permite que seu site tenha?

5.3 Seu site permite pacotes IPv4 broadcast direcionados? (Pense em uma maneira de teste usando ping.)

5.4 Se seu site usa IPv6, tente enviar um ping para os endereços multicast all-nodes. Quantas respostas foram recebidas?

5.5 Se seu site usa IPv6, descubra quando o IPv6 foi pela primeira vez instalado.

5.6 Qual é a principal diferença entre o esquema de endereçamento IP e o esquema de números de telefone dos Estados Unidos?

5.7 Os registradores de endereço no mundo inteiro cooperam para distribuir blocos de endereços IP. Descubra como eles garantem que nenhum ISP receberá endereços que se sobrepõem àqueles dados a outro ISP.

5.8 Quantos endereços IP seriam necessários para atribuir um endereço IP exclusivo a cada casa no seu país? E no mundo? O espaço de endereços IP é suficiente?

5.9 Suponha que cada pessoa no planeta tenha um telefone inteligente, um laptop, e dez outros dispositivos com um endereço IPv6 cada um. Que porcentagem do espaço de endereço IPv6 seria necessária?

---

* J. F. Shoch, *Internetwork Naming, Addressing, and Routing, Proceedings of COMPCON 1978.*

* A seção posterior discute a noção colon hex utilizada para endereços IPv6.

* Na prática, o limite é de 254 sub-redes e de 254 hosts por sub-rede porque o padrão reserva todos os endereços 1 e 0 de sub-rede e hosts.

* Endereçamento Classless preserva os endereços classe D, que são usados para multicast IPv4.

* A tabela da Figura 5.7, na página 53, resume todos os valores possíveis utilizados na notação de barra.

* Uma versão inicial do código TCP/IP que acompanhou o UNIX Berkeley usou incorretamente um hostid com tudo zero para transmissão. Porque o erro ainda sobrevive, o software TCP/IP, muitas vezes, inclui uma opção que permite a um

site usar um hostid com tudo zero para transmissão dirigida.

* Endereços anycast foram originalmente conhecidos como endereços de *cluster*.

CAPÍTULO 6

Mapeando endereços internet em endereços físicos (ARP)

CONTEÚDOS DO CAPÍTULO

6.1 Introdução

O capítulo anterior descreveu os esquemas de endereçamento IPv4 e IPv6 e mostrou que uma internet se comporta como uma rede virtual, usando apenas os endereços atribuídos ao enviar e receber pacotes. O Capítulo 2 reviu várias tecnologias de hardware de rede e observou que duas máquinas em determinada rede física *só podem se comunicar se conhecerem o endereço físico uma da outra*. O que não mencionamos é como um host ou um roteador mapeia um endereço IP para o endereço físico correto quando precisa enviar um pacote por uma rede física. Este capítulo aborda esse mapeamento, mostrando como ele é implementado nos IPv4 e IPv6.

6.2 O problema da resolução de endereços

Considere duas máquinas A e B que se conectam à mesma rede física. Cada uma possui um endereço IP atribuído I_A e I_B e um endereço de hardware (MAC, H_A e H_B). Por fim, a comunicação precisa ser executada enviando frames pela rede usando endereço de hardware que os equipamentos da rede reconhecem. Nosso objetivo, porém, é permitir que aplicativos protocolos de alto nível trabalhem apenas com endereços de Internet. Ou seja, queremos elaborar software que esconda o endereço de hardware em uma pilha de protocolos de nível baixo. Por exemplo, considere que a

máquina A precise enviar um pacote IP para a máquina B pela rede à qual ambas se conectam, mas A só conhece o endereço Internet de B, I_B. Surge a pergunta: como A mapeia o endereço Internet de B no endereço de hardware de B, H_B? Existem duas respostas, o IPv4 normalmente usa uma e o IPv6 usa a outra. Vamos considerar cada uma.

É importante notar que o mapeamento de endereço deve ser realizado a cada passo ao longo do caminho desde a origem até o destino final. Em particular surgem dois casos. Primeiro, no último passo da entrega de um pacote IP, ele deve ser enviado por uma rede física até o destino final. A máquina que envia o datagrama (normalmente um roteador) precisa mapear o endereço Internet do destino final para o endereço hardware do destino antes que a transmissão seja possível. Segundo, em qualquer ponto ao longo do caminho da origem ao destino, fora o passo final, o pacote precisa ser enviado a um roteador intermediário. Veremos que o software de protocolo sempre usa um endereço IP para identificar o próximo roteador ao longo do caminho. Assim, o emissor deve mapear o endereço Internet do roteador para o endereço de hardware.

O problema de mapear os endereços de alto nível em endereços físicos é conhecido como *problema de resolução de endereço* e foi resolvido de várias maneiras. Alguns conjuntos de protocolos mantêm tabelas em cada máquina, contendo pares de endereços de alto nível e endereços físicos. Outros protocolos resolvem o problema codificando endereços de hardware em endereços de alto nível. O uso de qualquer uma das técnicas de forma exclusiva torna o endereçamento de alto nível, na melhor das hipóteses, desajeitado. Este capítulo discute duas técnicas para resolução de endereços usadas pelos protocolos TCP/IP, e mostra quando cada uma é apropriada.

6.3 Dois tipos de endereços de hardware

Existem dois tipos básicos de endereços de hardware: os que são maiores do que a parte host de um endereço IP e os que são menores. Como ele dedica 64 bits para a parte do host de um endereço, o IPv6 acomoda todos os tipos de endereços de hardware. Portanto, a distinção só é importante para IPv4. Comecemos por considerar a técnica utilizada para IPv6 e para IPv4 quando os endereços são suficientemente pequenos. Vamos então considerar uma técnica que usa IPv4 quando os endereços são grandes.

6.4 Resolução por meio de mapeamento direto

O IPv6 usa uma técnica conhecida como *mapeamento direto*. A ideia básica é simples: usar o endereço de hardware de um computador como parte do host do seu endereço de Internet. O IPv4 pode usar mapeamento direto quando os endereços são suficientemente pequenos. A Figura 6.1 ilustra o conceito.

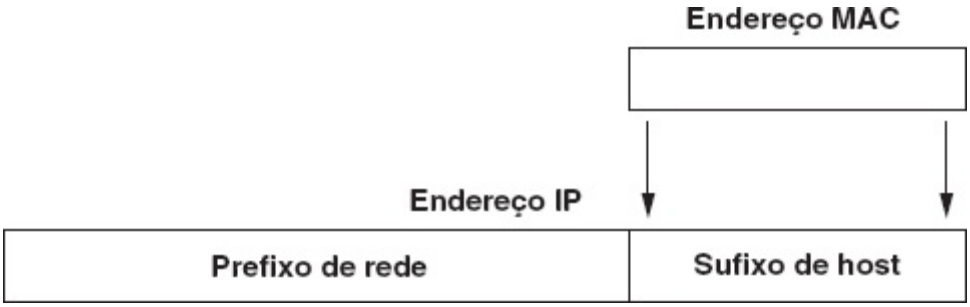


Figura 6.1 Uma ilustração do esquema de um mapeamento direto no qual um endereço de hardware de um computador é inserido em seu endereço IP.

Para ver como mapeamento direto trabalha com IPv4, é importante saber que alguns hardwares usam, como endereços de hardware, inteiros pequenos, configuráveis. Sempre que um novo computador é adicionado a essa rede, o administrador do sistema escolhe um endereço de hardware e configura a placa de rede do computador. A única regra importante é que dois computadores não podem ter o mesmo endereço. Para tornar a tarefa fácil e segura, um administrador geralmente atribui endereços sequencialmente: ao primeiro computador ligado à rede, é atribuído o endereço 1, ao segundo computador, o endereço 2, e assim por diante.

Enquanto um gerente tem a liberdade de escolher os dois, um endereço IP e um endereço de hardware, o par de endereços pode ser selecionado de tal forma que o endereço de hardware e a parte do host do endereço IP sejam idênticos. O IPv6 faz essa atribuição trivial – o endereço de hardware sempre se ajusta na área do endereço usado para um ID de interface. Para o IPv4, considere um exemplo em que à rede foi atribuído o prefixo IPv4:

192.5.48.0 / 24

O prefixo de rede ocupa os três primeiros octetos, deixando um octeto para o ID de host. Ao primeiro computador da rede é atribuído um endereço de hardware 1 e o endereço IP 192.5.48.1, ao segundo computador é atribuído o endereço de hardware 2 e o endereço IP 192.5.48.2, e assim por diante.

Ou seja, a rede está configurada de tal forma que o octeto de baixa ordem de cada endereço IP é o mesmo do endereço de hardware do computador. Claro, o exemplo só funciona se os endereços de hardware estiverem entre 1 e 254.

6.5 Resolução em uma rede mapeada diretamente

Se o endereço IP de um computador inclui seu endereço de hardware, a resolução de endereço é trivial. Dado um endereço IP, o endereço de hardware do computador pode ser obtido da parte do host. No exemplo anterior, se o software de protocolo fornece o endereço IP de um computador na rede (por exemplo, 192.5.48.3), o endereço de hardware correspondente pode ser computado simplesmente extraindo o octeto 3 de baixa ordem. Como o nome *mapeamento direto* indica, ele pode ser feito sem qualquer referência de dados externos. Na verdade, o mapeamento é extremamente eficiente, pois só requer poucas instruções de máquina. O mapeamento direto tem a vantagem de que novos computadores podem ser adicionados à rede sem mudar as atribuições existentes e sem propagar novas informações aos computadores existentes.

Matematicamente, o mapeamento direto significa selecionar uma função f que mapeia o endereço IP de redes físicas. Resolver endereços IP I_A significa computar

$$H_A = f(I_A)$$

Embora seja possível escolher mapeamentos diferentes desse exemplo, queremos que o cálculo de f seja eficiente e também que as escolhas sejam fáceis para um humano entender. Assim, é preferível um esquema em que o relacionamento entre o endereço IP e o endereço de hardware seja óbvio.

6.6 Resolução de endereço IPv4 através de vínculo dinâmico

Embora eficiente, o mapeamento direto não pode ser usado com IPv4 se endereços de hardware forem maiores que o endereço IPv4. Especificamente, um endereço Ethernet MAC não pode ser mapeado diretamente em um endereço IPv4 porque tem comprimento de 48 bits, e o endereço IPv4 só tem 32 bits. Além disso, devido ao fato de ser atribuído quando um dispositivo é fabricado, um endereço Ethernet MAC não pode ser mudado.

Os projetistas dos protocolos TCP/IP descobriram uma solução criativa para o problema de resolução de endereço para redes como Ethernet, que possuem capacidade de broadcast. A solução permite que novos hosts ou roteadores sejam acrescentados à rede sem recompilar código, e não exige manutenção de um banco de dados centralizado. Para evitar a manutenção de um banco de dados centralizado, os projetistas escolheram usar um protocolo de baixo nível para vincular endereços dinamicamente. Chamado *Address Resolution Protocol* (*ARP*), o protocolo oferece um mecanismo que é razoavelmente eficiente e que não requer que um administrador configure tabelas manualmente.

A ideia por trás da tradução dinâmica com ARP é simples: quando ele quer traduzir o endereço IP I_B, um host envia por broadcast uma requisição de pacote ARP que pede ao host com endereço IP I_B para responder com seu endereço de hardware H_B. Todos os hosts, incluindo B, recebem a requisição, mas somente o host B reconhece seu endereço IP e envia uma resposta que contém seu endereço de hardware. O ARP só é usado quando um host precisa mandar um pacote IP. Então, quando ele recebe uma resposta ao seu requisito, o host que fez a requisição usa a informação para enviar pacotes diretamente para B. Podemos fazer o resumo a seguir.

> *O Address Resolution Protocol, ARP, permite que um host encontre o endereço físico de um host alvo na mesma rede física, a partir somente do endereço IP do host alvo.*

A Figura 6.2 ilustra o protocolo ARP mostrando uma solicitação do host A para B, e a resposta de B . Note que, embora a solicitação seja broadcast, a resposta não é.

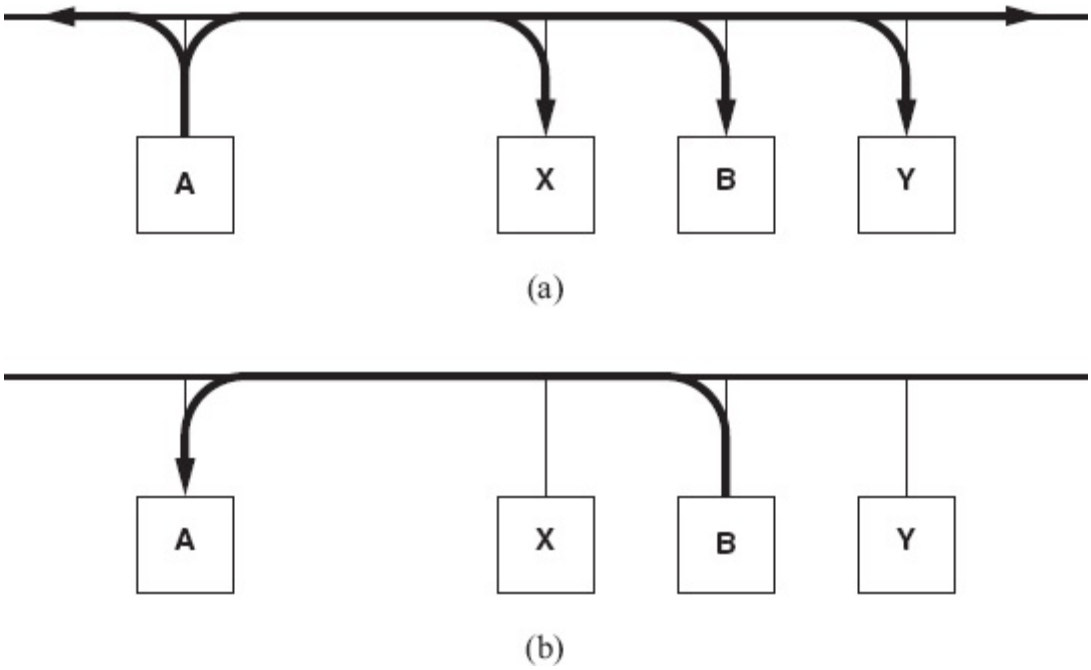


Figura 6.2 Ilustração de ARP onde (a) host *A* envia uma solicitação ARP contendo *IB*, e (b) host *B* responde com uma resposta ARP que especifica seu endereço de hardware *HB*.

6.7 O cache ARP

Pode parecer tolice que, para *A* enviar um pacote para *B*, ele primeiro envia um broadcast que consome tempo em cada host. Ou então pode parecer tolice ainda maior que *A* transmita por broadcast a pergunta "como posso alcançar você?" em vez de apenas transmitir o pacote que deseja entregar. Mas existe um motivo importante para a troca. O broadcasting é muito mais dispendioso de ser usado toda vez que uma máquina precisa transmitir um pacote a outra, pois cada máquina na rede deve receber e processar o pacote de broadcast. Então, o software ARP em *A* usa uma otimização: ele grava a resposta e reusa a informação para transmissões sucessivas.

A norma especifica que o software ARP deve manter um *cache* de ligações de endereço IP a hardware recém-adquiridas. Ou seja, sempre que um computador envia uma solicitação ARP e recebe uma resposta ARP, ele salva o endereço IP e informações de endereço de hardware correspondentes em seu cache temporariamente. Isso reduz os custos globais de comunicação de forma dramática. Ao transmitir um pacote, um computador sempre olha em seu cache antes de enviar uma solicitação ARP. Se ele encontrar a ligação desejada, não precisa enviar um pedido.

Assim, quando dois computadores em uma rede se comunicam, eles começam com um pedido e resposta ARP e, em seguida, repetidamente transferem pacotes sem usar ARP para cada pacote.

Experiências mostram que, devido ao fato de a maioria das comunicações de rede envolverem mais de uma transferência, mesmo um cache pequeno é útil.

6.8 ARP cache timeout

O cache ARP oferece um exemplo de *estado flexível*, uma técnica normalmente usada nos protocolos de rede. O nome descreve uma situação em que a informação pode se tornar "velha" sem aviso. No caso do ARP, considere dois computadores, *A* e *B*, ambos conectados a uma Ethernet. Considere que *A* tenha enviado uma requisição ARP e *B* tenha respondido. Além disso, considere que, depois da troca, *B* falhe. O computador *A* não receberá qualquer notificação da falha. Além do mais, por já ter informação de vínculo de endereço para *B* em seu cache ARP, o computador *A* continuará a enviar pacotes para *B*. O hardware Ethernet não fornece indicação de que *B* não está *on-line*, pois a Ethernet não tem entrega garantida. Assim, *A* não tem como saber quando as informações em seu cache ARP se tornaram incorretas.

Em um sistema que usa o estado flexível, a responsabilidade pela exatidão é do proprietário do cache. Normalmente, os protocolos que implementam o estado flexível utilizam timers que são configurados quando informação é adicionada ao cache, e quando o timer expira, a informação é deletada. Por exemplo, sempre que a informação de vínculo é colocada em um cache ARP, o protocolo exige que um timer seja definido, com um tempo limite (timeout) típico de 20 minutos. Quando o timer expira, a informação deve ser removida. Depois da remoção, existem duas possibilidades: se nenhum outro pacote for enviado ao destino, nada ocorrerá; se um pacote tiver de ser enviado ao destino e não houver um vínculo presente no cache, o computador seguirá o procedimento normal de transmitir uma requisição ARP por broadcast e obter o vínculo. Se o destino ainda for alcançável, o vínculo novamente será colocado no cache ARP. Se não, o emissor descobrirá que o destino está *off-line*.

O uso do estado flexível no ARP tem vantagens e desvantagens. A principal vantagem surge pela autonomia. Primeiro, um computador pode

determinar quando a informação em seu cache ARP deve ser invalidada independente dos outros computadores. Segundo, um emissor não precisa da comunicação bem-sucedida com o receptor ou um terceiro para determinar que um vínculo se tornou inválido; se um destino não responder a uma requisição ARP, o emissor o declarará como parado. Terceiro, o esquema não conta com o hardware da rede para fornecer transferência confiável ou informar a um computador se outro computador está *on-line*. A principal desvantagem do estado flexível surge pelo atraso – se o intervalo do timer for de *N* minutos, um emissor não detectará que um receptor falhou antes que *N* minutos tenham passado.

6.9 Refinamentos ARP

Vários refinamentos ARP foram incluídos no protocolo que reduz a quantidade de tráfego de rede e automatiza a recuperação após uma mudança de endereço de hardware.

- Em primeiro lugar, observe que o host *A* só transmite uma solicitação ARP broadcast para *B* quando se tem um pacote de Internet pronto para enviar para *B*. Como a maioria dos protocolos de Internet envolvem um intercâmbio bilateral, há uma alta probabilidade de que o host *B* irá enviar um pacote de Internet de volta para *A*, em um futuro próximo. Para antecipar a necessidade de *B* e evitar o tráfego extra de rede, ARP solicita que *A* inclua o seu endereço de IP para hardware de ligação ao enviar uma solicitação a *B*, que extrai a ligação de *A* a partir da solicitação e salva a ligação em seu cache ARP. Assim, quando enviar um pacote de Internet para *A*, *B* encontrará a ligação já em seu cache.
- Em segundo lugar, perceber que, devido às solicitações serem transmitidas via broadcast, todas as máquinas da rede recebem uma cópia da solicitação. O protocolo especifica que cada máquina extraia o endereço de ligação IP para hardware do remetente a partir da solicitação, e utilize as informações para atualizar a ligação em seus cache. Por exemplo, se *A* transmite uma solicitação broadcast, as máquinas da rede vão atualizar suas informações para *A*. Máquinas

que ainda não têm uma entrada para *A* em seu cache não adicionarão informações de *A*; a norma especifica apenas a atualização do endereço de hardware em entradas existentes. A ideia é que, se uma máquina já teve comunicação com *A*, seu cache não deve ser obstruído com uma entrada inútil.

- Em terceiro lugar, quando um computador tem a sua interface host substituída, (por exemplo, porque o hardware falhou), seu endereço físico muda. Os outros computadores da rede que tem armazenada uma ligação em seu cache ARP precisam ser informados para que possam alterar a entrada. O computador pode avisar aos outros sobre um novo endereço, transmitindo uma *solicitação broadcast ARP gratuita*. Alterar um endereço MAC requer a substituição de uma NIC, que ocorre quando um computador está desligado. Como um computador não sabe se o seu endereço MAC mudou, a maioria dos computadores transmite um ARP gratuito durante a inicialização do sistema. O ARP gratuito tem um objetivo secundário: ver se alguma outra máquina está usando o mesmo endereço IP. A máquina de inicialização envia uma solicitação ARP para o seu próprio endereço IP; se receber uma resposta, deve haver um erro de configuração ou um problema de segurança em que um computador é intencionalmente imitado.

A seguir se resume a ideia-chave de atualizações automáticas de cache.

O endereço de ligação IP-para-hardware do remetente está incluído em cada transmissão ARP broadcast; receptores usam as informações para atualizar suas informações de ligação de endereço. O destinatário usa a informação para criar uma nova entrada de cache na expectativa de uma resposta.

6.10 Relação de ARP com outros protocolos

Como vimos, como o IPv6 usa mapeamento direto, não precisa de ARP. Assim, ARP fornece apenas um possível mecanismo de mapeamento de um endereço IP para um endereço de hardware. Curiosamente, ARP e outros mecanismos de ligação de endereço seriam completamente desnecessários

se pudéssemos redesenhar todo o hardware de rede para reconhecer endereços IP. Assim, do nosso ponto de vista, a ligação de endereço só é necessária para ocultar os endereços de hardware subjacentes. Conceitualmente, impusemos nosso novo esquema de endereçamento IP acima de qualquer mecanismo de endereço de baixo nível que o hardware usa. Portanto, vemos ARP como um protocolo de baixo nível associado ao hardware, em vez de uma parte fundamental dos protocolos TCP/IP, que são executados acima do hardware. A ideia pode ser resumida:

> *ARP é um protocolo de baixo nível que esconde o endereçamento subjacente usado pelo hardware de rede, que nos permite atribuir um endereço IP arbitrário para cada máquina. Pensamos em ARP como associado com o sistema de rede física, e não como parte dos protocolos da Internet.*

6.11 Implementação do ARP

Funcionalmente, o ARP é dividido em duas partes. A primeira fornece resolução de endereços para pacotes de saída: dado o endereço IP de um computador na rede, ele encontra o endereço de hardware do computador. Se um endereço não está no cache, ele envia uma solicitação. A segunda parte lida com pacotes ARP de entrada. Ele atualiza o cache, responde às solicitações de outros computadores na rede e checa se uma resposta coincide com uma solicitação pendente.

A resolução de endereços para os pacotes que saem parece ser direta, mas pequenos detalhes complicam uma implementação. Dado um endereço IP de um computador para o qual um pacote deve ser enviado, o software consulta seu cache ARP para ver se contém o mapeamento do endereço IP para o endereço hardware. Se estiver no cache, o software extrai o endereço hardware, insere no endereço de destino em um frame de saída e o envia. Se não estiver no cache, duas coisas devem acontecer. Primeiro, o host deve guardar o pacote de saída para que possa ser enviado assim que o endereço estiver resolvido. Segundo, o software ARP deve enviar uma solicitação ARP.

A coordenação entre a parte da ARP que envia pedidos e a parte que recebe respostas pode se tornar complicada. Se uma máquina de destino está desligada ou muito ocupada para aceitar o pedido, nenhuma resposta será recebida (ou a resposta pode ser atrasada). Além disso, como a

Ethernet é um sistema de entrega de melhor esforço, o pedido de transmissão broadcast ARP inicial ou a resposta pode ser perdida. Portanto, um remetente deve retransmitir o pedido pelo menos uma vez, o que significa que um temporizador deve ser utilizado e o lado de envio deve cancelar o temporizador se uma resposta chega. Mais importante, surge a questão: enquanto ARP está resolvendo um determinado endereço IP, o que acontece se outro aplicativo tentar enviar para o mesmo endereço? O ARP pode optar por criar uma fila de pacotes de saída, ou simplesmente pode optar por descartar pacotes sucessivos. Em qualquer caso, as principais decisões de design envolvem o acesso concorrente. Outras aplicações podem prosseguir enquanto o ARP resolve o endereço? Se outro aplicativo tentar enviar para o mesmo endereço, as aplicações devem ser bloqueadas ou o ARP deve simplesmente criar uma fila de pacotes de saída? Como pode ser desenvolvido o software ARP para se prevenir que se transmita desnecessariamente um segundo pedido para um computador?

Um detalhe final distingue a gestão de cache ARP da gestão de um cache típico. Em um cache típico, os tempos de espera são usados para eliminar entradas inativas. Assim, o registro data/hora de uma entrada é reiniciado cada vez que a entrada é utilizada. Quando se precisar de espaço, a entrada com o registro data/hora mais antigo é removido do cache. Para uma cache ARP, no entanto, o momento em que uma entrada foi referenciada pela última vez é irrelevante – o ARP pode continuar a usar a entrada, mesmo que o computador de destino tenha falhado. Assim, é importante limitar o tempo de entrada, mesmo que ela ainda esteja sendo usada.

Dissemos que a segunda parte do software ARP trata de pacotes ARP que chegam da rede. Quando um pacote ARP chega, o software primeiro extrai o endereço IP do emissor e o par de endereços de hardware e examina o cache local a fim de ver se já tem uma entrada para o emissor. Se existir uma entrada de cache para determinado endereço IP, o handler atualiza essa entrada sobrescrevendo o endereço físico com o endereço físico obtido do pacote. Então, o receptor processa o restante do pacote ARP.

Para processar o resto de um pacote ARP, o receptor verifica a operação. Se o pacote de entrada é uma solicitação ARP, a máquina de recepção verifica se ele é o alvo do pedido (ou seja, alguma outra máquina transmitiu broadcast um pedido e “Eu” sou o alvo do pedido). Se assim for, o software ARP acrescenta o par de endereços do remetente à seu cache (se o par já não está presente), forma uma resposta e a envia diretamente de volta para o solicitante. Se o endereço IP mencionado em um pedido não corresponde ao endereço IP local (ou seja, o pedido é para outro computador), o pacote ARP de entrada é descartado.

Se o pacote de entrada é uma resposta ARP, o receptor tenta corresponder a resposta com um pedido anterior. Se a resposta não corresponder a um pedido, o pacote é descartado. Caso contrário, a ligação endereço é conhecida (a entrada no cache já terá sido atualizada no primeiro passo acima). Portanto, o software ARP examina a fila de pacotes IP de saída (pacotes que estavam à espera de resposta). O software ARP coloca cada pacote IP em um frame, usa as informações de ligação de endereços do cache para preencher o endereço de destino no frame e envia o pacote.

6.12 Encapsulamento e identificação do ARP

Quando as mensagens ARP trafegam de uma máquina para outra, elas precisam ser transportadas em frames de rede. A Figura 6.3 mostra que a mensagem ARP é transportada na parte de dados de um frame (ou seja, é tratada como dado).

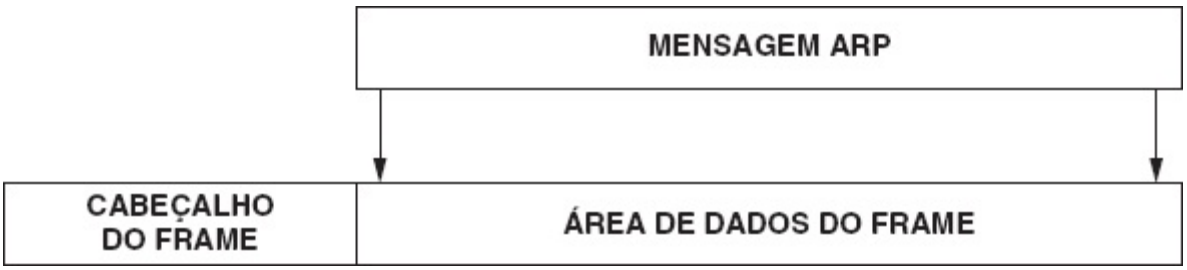


Figura 6.3 Uma mensagem ARP encapsulada em um frame de rede.

Para identificar o frame como transportando uma mensagem ARP, o emissor atribui um valor especial ao campo de tipo no cabeçalho do frame. Quando um frame chega a um computador, o software de rede utiliza o tipo de frame para determinar seu conteúdo. Na maioria das tecnologias, um único valor de tipo é usado para todos os frames que transportam uma mensagem ARP – o software de rede no receptor precisa examinar melhor a mensagem ARP para distinguir entre requisições ARP e respostas ARP. Por exemplo, em uma Ethernet, os frames transportando mensagens ARP têm um campo de tipo 0x0806 onde o prefixo *0x* indica um valor hexadecimal. O tipo de frame para ARP foi padronizado pela IEEE (que é dona das normas Ethernet). Assim, quando ARP trafega por qualquer Ethernet, o tipo é sempre 0x0806. Outras tecnologias de hardware podem utilizar outros valores.

6.13 Formato de mensagem ARP

Diferentemente da maioria dos protocolos, uma mensagem ARP não possui um cabeçalho de formato fixo. Em vez disso, para tornar o ARP útil para diversas tecnologias de rede, os projetistas optaram por tornar o tamanho do campo de endereço hardware dependente dos endereços usados pela rede subjacente. De fato, os projetistas não restringiram ARP para endereços IPv4. Em vez disso, o tamanho do campo de protocolo de endereço em uma mensagem ARP depende do tipo de protocolo de endereço de alto nível usado. Ironicamente, com apenas poucas exceções tais como experiências de pesquisa, o ARP é sempre usado com protocolo de endereço IPv4 de 32 bits e endereço de hardware Ethernet de 48 bits. O ponto importante é descrito a seguir.

> *O projeto permite ao ARP mapear um protocolo de endereço de alto nível arbitrário para um endereço de hardware de rede arbitrária. Na prática, a ARP só é usado para mapear os endereços IPv4 de 32 bits para endereços Ethernet de 48 bits.*

O exemplo na Figura 6.4 mostra o formato da mensagem ARP em 28 octetos quando usado com protocolo de endereço IPv4 e em endereço de hardware Ethernet. O protocolo de endereço tem tamanho de 32bits (4 octetos), e o endereço de hardware tem tamanho de 48 bits (6 octetos).

Figura 6.4 O formato de mensagem ARP quando usado para mapear um endereço IPv4 em um endereço Ethernet.

A Figura 6.4 mostra uma mensagem ARP com 4 octetos por linha, um

formato que é padrão por todo este texto e para os padrões TCP/IP. Infelizmente, o endereço Ethernet 48 bits significa que os campos na mensagem ARP não se alinham bem em limites de 32 bits, tornando o diagrama difícil de ser lido. Por exemplo, o endereço de hardware do emissor, rotulado como HARD EMISSOR, ocupa 6 octetos contíguos, de modo que ocupa duas linhas no diagrama, a terceira linha e metade a quarta linha.

O campo TIPO DE HARDWARE especifica um tipo de interface de hardware para o qual o emissor busca uma resposta; ele contém o valor *1* para especificar que o endereço de hardware é um endereço MAC Ethernet. De modo semelhante, o campo TIPO PROTOCOLO especifica o tipo do endereço de protocolo de alto nível que o emissor forneceu; ele contém o 0x0800 (hexadecimal) para especificar que o protocolo de endereço é IPv4. O formato da mensagem ARP permite que um computador possa interpretar qualquer mensagem ARP, mesmo que o computador não reconheça o tipo de endereço de protocolo ou tipo de endereço de hardware. A interpretação é possível porque os campos fixos perto do início da mensagem rotulados HLEN e PLEN especificam o tamanho de um endereço de hardware e o tamanho de um endereço de protocolo.

O campo OPERAÇÃO especifica uma requisição ARP (*1*), resposta ARP (*2*), requisição RARP* (*3*) ou resposta RARP (*4*). Quando se envia um pacote ARP, um emissor coloca seu endereço de hardware nos campos HARD EMISSOR. E seu endereço IPv4, se conhecido, em EMISSOR IPv4.

Os campos HARD DESTINO e IPv4 DESTINO fornecem o endereço de hardware e o protocolo de endereço da máquina destino, se conhecido. Para a mensagem resposta (ou seja, resposta), a informação do destino pode ser extraída da mensagem de pedido. Quando se envia uma mensagem de pedido, o emissor sabe o endereço IPv4 do destino, mas não sabe o endereço de hardware. Portanto, em um pedido, o endereço de hardware de destino contém zeros. Podemos então fazer o resumo a seguir.

> *Uma resposta ARP carrega o endereço IPv4 e o endereço de hardware do solicitante original, bem como o endereço IPv4 e o endereço de hardware do remetente. Em um pedido, o endereço de hardware de destino é definido como zero, porque é desconhecido.*

6.14 Revalidação automática de cache ARP

É possível usar uma técnica que evita a introdução de *jitter* (ou seja, variação nos tempos de transferência de pacotes). Para entender como pode ocorrer o jitter, imagine um computador enviando um fluxo constante de pacotes para outro computador. Sempre que um timer ARP expirar, o ARP removerá o cache de entrada e o próximo pacote de saída vai disparar o ARP. O pacote vai sofrer um atraso até que o ARP possa enviar um pedido e receber uma resposta. O atraso dura o dobro do tempo que ele leva para transmitir um pacote. Embora esse atraso possa parecer insignificante, eles introduzem jitter, especialmente para dados em tempo real, tais como chamadas telefônicas por voz.

A chave para evitar o jitter surge da *revalidação antecipada*. Ou seja, a implementação associa dois contadores a cada entrada no cache ARP: o timer tradicional e um timer de revalidação. O timer de revalidação é ajustado para um valor ligeiramente menor do que o timer tradicional. Quando o timer de revalidação expira, o software examina a entrada. Se os datagramas tiverem usado a entrada recentemente, o software envia uma requisição ARP e continua a usar a entrada. Ao receber a requisição, a estação de destino responde, e os dois timers são reiniciados. Claro que, se nenhuma resposta chegar, o timer tradicional expirará, e o ARP vai continuar a tentar obter uma resposta. Na maioria dos casos, porém, uma revalidação pode reiniciar o timer sem interrupção do fluxo de pacotes.

6.15 Resolução de endereços reversos (RARP)

Vimos que o campo de operação em um pacote ARP pode especificar uma mensagem *Reverse Address Resolution* (*RARP*). A RARP não é mais importante na Internet, mas já foi um protocolo essencial usado para a efetuar o boot de sistemas que não tinham armazenamento estável. O paradigma é simples: na inicialização, um sistema transmite um pedido RARP para obter um endereço IP. O pedido contém o endereço Ethernet do emissor. Um servidor na rede recebe a solicitação, procura o endereço Ethernet em um banco de dados, extrai o endereço IPv4 correspondente do banco de dados e envia uma resposta RARP com a informação. Uma vez que a resposta chega, o sistema sem disco continua a inicialização, e utiliza o endereço IPv4 para todas as comunicações. Curiosamente, a RARP utiliza o mesmo formato de pacotes como ARP.** A única diferença é que usa Ethernet tipo 0x8035.

A RARP não é mais importante para os dispositivos sem disco, mas tem

uma aplicação interessante em centros de dados em nuvem. Em um centro de dados, quando uma máquina virtual migra de um PC para outro, a VM mantém o mesmo endereço Ethernet que estava usando antes. Para permitir que o comutador Ethernet subjacente saiba que houve movimento, a VM deve enviar um frame (o endereço de origem no frame fará com que o interruptor atualize suas tabelas). Que frame deve uma VM enviar? Aparentemente, a RARP foi escolhida porque tem a vantagem de atualizar a tabela de endereços MAC no comutador sem causar processamento adicional. Na verdade, depois de atualizar a tabela de endereços, o comutador vai simplesmente descartar o pacote RARP.

6.16 Caches ARP nos comutadores da camada três

Um comutador Ethernet é classificado como um *comutador de camada 3* se ele compreende os pacotes IP e pode examinar os cabeçalhos IP no momento de decidir como processar um pacote. Alguns comutadores de camada 3 têm uma implementação incomum de ARP que pode ser confusa para alguém que está tentando entender o protocolo.

A implementação surge de um desejo de reduzir o tráfego ARP. Para ver por que a otimização é útil, pense sobre o tráfego gerado por ARP. Suponha que um comutador tenha 192 portas que se conectam a computadores. Se cada computador implementa ARP timeouts de cache, o computador irá periodicamente fazer entradas de cache com timeout e, em seguida, transmitir uma solicitação ARP. Mesmo que o computador use a revalidação automática, o comutador vai receber transmissões periódicas que devem ser enviadas para todos os computadores.

Como pode um comutador reduzir o tráfego de broadcast para computadores? Observamos três coisas. Primeiro, um switch pode assistir tráfego ARP e manter um registro de ligações entre endereços IP e endereços Ethernet. Em segundo lugar, se ele tem as informações necessárias, um comutador pode responder a uma solicitação ARP sem transmitir o pedido. Em terceiro lugar, um endereço Ethernet só pode mudar se um computador é desligado, e um comutador pode dizer se um computador foi desligado. Portanto, um comutador pode criar seu próprio cache de informações ARP e pode responder a solicitações. Por exemplo, se o computador *A* envia uma solicitação ARP para o computador *B*, o comutador pode interceptar a solicitação, procurar em seu cache e criar uma resposta ARP, *como se a resposta tivesse vindo de B.*

Para um ambiente de produção, a otimização descrita anteriormente

funciona bem. O computador *A* parece transmitir um pedido ARP para *B*, e *A* recebe uma resposta válida. Ele reduz o tráfego extra, e não requer nenhuma modificação no software em execução nos computadores. No entanto, para qualquer um que esteja testando protocolos de rede, pode ser confuso. Um computador transmite uma solicitação que não é recebida por qualquer outro computador na rede! Além disso, o computador que enviou o pedido recebe uma resposta fantasma que nunca foi enviada pela fonte!

6.17 Proxy ARP

Intranets às vezes usam uma técnica conhecida como *proxy ARP* para implementar uma forma de segurança. Vamos primeiro examinar o proxy ARP e, em seguida, ver como ele pode ser usado.

No início da história da Internet, foi desenvolvida uma técnica que permitia um único prefixo IPv4 para ser usado em duas redes. Originalmente chamada de *O ARP Hack*, a técnica tornou-se conhecida pelo termo mais formal *proxy ARP*. Ele se baseia em um computador que possui duas conexões de rede e executa o software ARP para fins especiais.

A Figura 6.5 mostra um exemplo de configuração no qual o proxy ARP pode ser usado.

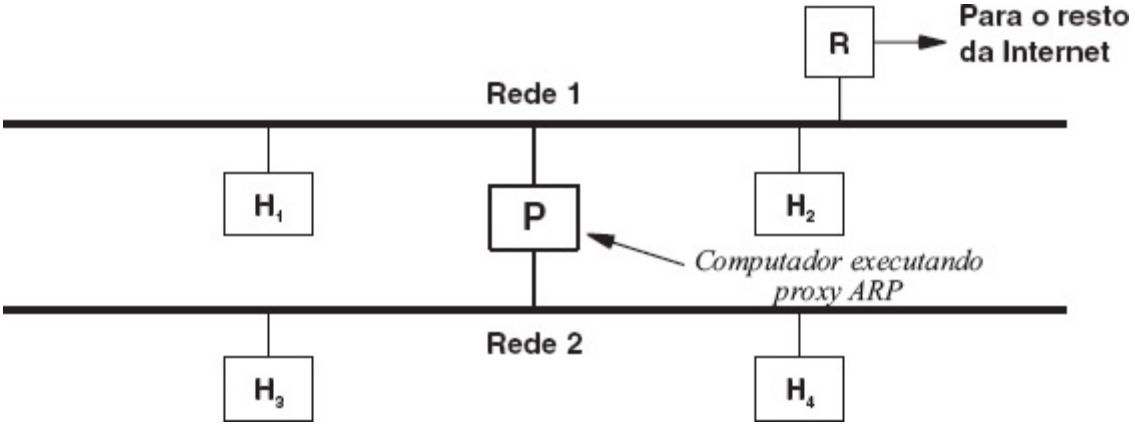


Figura 6.5 Ilustração de duas redes usando proxy ARP.

Na figura, o computador denominado *P* executa o software proxy ARP. O computador *P* tem um banco de dados que contém o endereço IPv4 e o endereço Ethernet MAC de cada uma das outras máquina na rede 1 e na rede 2. O roteador e todos os outros hosts executam padrão ARP; eles não

sabem que o proxy ARP está sendo usado. Mais importante, todos os outros hosts e roteadores são configurados como se estivessem em uma única rede.

Para entender a interação proxy ARP, considere o que acontece quando o roteador R recebe um pacote a partir da Internet que é destinado para o endereço IPv4 que está sendo usado. Antes de poder entregar o pacote de entrada, R deve usar ARP para encontrar o endereço de hardware do computador. R transmite uma solicitação ARP. Há dois casos a considerar: se o destino está na rede 1 ou na rede 2. Considere o primeiro caso (por exemplo, suponha que o destino é o host H_1). Todas as máquinas na rede 1 recebem uma cópia do pedido de R. O computador P olha em seu banco de dados, descobre que H_1 é uma rede 1 e ignora o pedido. O Host H_1 também recebe uma cópia do pedido e responde normalmente (ou seja, envia uma resposta ARP).

Considere agora o segundo caso, em que R transmite uma solicitação para uma máquina na rede 2 (por exemplo, host H_4). ARP só foi concebido para ser utilizado em uma única rede, de modo que a transmissão para um computador em outra rede parece ser uma violação do protocolo. No entanto, R está se comportando corretamente, porque não sabe que existem duas redes. Todos os computadores na rede 1 receberão uma cópia da transmissão, incluindo P. O computador P consulta seu banco de dados, descobre que H_4 está na rede 2 e envia uma resposta ARP que especifica o endereço de Ethernet de R

Como sendo o endereço de hardware, R vai receber a resposta, colocá-la no cache ARP e enviar um pacote IP para P (porque P imitou H_4). Quando recebe um pacote Internet, P examina o endereço de destino no pacote e o encaminha para H_4.

O proxy ARP também lida com a imitação e encaminhamento quando um computador na rede 2 envia a um computador na rede 1. Por exemplo, quando H_4 forma um pacote de Internet e precisa enviá-lo para o roteador R, H_4 transmite um pedido ARP para R. P receberá uma cópia do pedido, consultará seu banco de dados e enviará uma resposta ARP que se faz passar por R.

Como o proxy ARP pode ser usado para segurança? O proxy ARP pode ser usado para um firewall ou em uma conexão VPN. A ideia é que como uma máquina proxy ARP representará máquinas na rede secundária, todos os pacotes devam transitar pela máquina proxy ARP, onde podem ser verificados. Na Figura 6.4, por exemplo, um site poderia colocar todos os hosts na rede 2 e colocar o software de firewall na máquina *P*. Sempre que um pacote chega a partir da Internet, passa por *P* (onde o pacote pode ser examinado e o firewall pode ter suas regras aplicadas) em seu caminho para o host de destino.

6.18 Descoberta de vizinho IPv6

O IPv6 usa o termo *vizinho* para descrever outro computador na mesma rede. IPv6's *Neighbor Discovery Protocol* (*NDP*) substitui o ARP e permite a um host mapear entre um endereço IPv6 e um endereço de hardware.* No entanto, NDP inclui muitas outras funções. Ele permite a um host encontrar o conjunto de roteadores em uma rede; determinar se um determinado vizinho ainda é alcançável; aprender o prefixo da rede que está sendo usada; determinar as características do hardware de rede (por exemplo, o tamanho máximo do pacote); configurar um endereço para cada interface e verificar que nenhum outro host na rede o está usando; e encontrar o melhor roteador para usar para um determinado destino.

Em vez de criar um protocolo análogo ao ARP para lidar com a descoberta de vizinho, os desenvolvedores do IPv6 escolheram usar ICMPv6.** Assim, ICMPv6 inclui mensagens que um computador usa para encontrar os seus vizinhos na inicialização e para verificar periodicamente o status de um vizinho.

Uma diferença fundamental entre ARP e NDP surge da forma como cada um lida com o status dos vizinhos. O ARP usa uma abordagem de fim de ligação com o estado flexível. Ou seja, O ARP espera até que um datagrama seja enviado para um vizinho antes de tomar qualquer ação. Depois, executa uma troca, armazenando a ligação em seu cache e, em seguida, enviando os pacotes IP para o vizinho sem verificar o status deste até que o cache ARP temporizador expire. O atraso pode durar muitos minutos. NDP usa ligação inicial e tem uma abordagem proativa para a manutenção do estado. Em vez de esperar até que um datagrama seja enviado, um nó IPv6 utiliza NDP para descobrir vizinhos na inicialização. Além disso, um nó IPv6 verifica continuamente o estado de vizinhos. Assim, a transmissão de um datagrama IPv6 para um vizinho pode prosseguir sem demora e não

envolve transmissão broadcast.

6.19 Resumo

O software de protocolo de Internet utiliza endereços IP. Para enviar um pacote através de uma rede, no entanto, devem ser usados os endereços de hardware. Portanto, o software de protocolo deve mapear o endereço Internet de um computador para um endereço de hardware. Se os endereços de hardware são menores do que os endereços IP, um mapeamento direto pode ser estabelecido por ter endereço de hardware da máquina embutido em seu endereço IP. Como o IPv6 tem endereços grandes, ele sempre usa mapeamento direto. O IPv4 usa o mapeamento dinâmico quando o endereço de hardware é maior do que a parte do host de um endereço IPv4. O Address Resolution Protocol (ARP) executa a resolução dinâmica de endereço, utilizando apenas o sistema de comunicação de rede de baixo nível. O ARP permite que um computador resolva endereços sem o uso de um banco de dados de ligações e sem a necessidade de um gerente para configurar o software.

Para encontrar o endereço de hardware de outro computador na mesma rede, uma máquina transmite uma solicitação ARP. O pedido contém o endereço IPv4 da máquina de destino. Todas as máquinas em uma rede recebem um pedido ARP, e apenas a máquina de destino responde. A resposta ARP contém o endereço IPv4 do remetente e o endereço de hardware. As respostas são enviadas unicast; elas não são broadcast.

Para tornar ARP eficiente, cada máquina armazena no cache o endereço de ligações IP para hardware. Usar um cache evita o tráfego desnecessário de broadcast; revalidação inicial pode ser usada para eliminar o jitter.

Um protocolo mais antigo relacionado a ARP, RARP, está sendo usado em centros de dados em nuvem. A técnica de proxy ARP pode ser usada em sistemas de segurança, como uma VPN ou um firewall que é transparente para os roteadores e hosts. O IPv6 substituiu ARP com um Neighbor Discovery Protocol (NDP). Ao contrário do ARP, NDP verifica continuamente o estado do vizinho para determinar se este continua acessível.

EXERCÍCIOS

6.1 Dado um pequeno conjunto de endereços de hardware (inteiros positivos), você pode descobrir uma função f e uma atribuição de endereços IP de modo que f mapeie os endereços IP 1-para-1 nos

endereços físicos e o cálculo de f seja eficiente? (Dica: consulte literatura sobre hashing perfeito.)

6.2 Em que casos especiais um host conectado a uma Ethernet não precisa usar ARP ou um cache ARP antes de transmitir um datagrama IP? (Dica: consulte literatura sobre multicast).

6.3 Um algoritmo comum para gerenciar o cache ARP substitui a entrada usada menos recentemente quando acrescenta uma nova. Sob que circunstâncias esse algoritmo pode produzir o tráfego de rede desnecessário?

6.4 O ARP deve modificar o cache mesmo quando recebe informações sem requisitá-las especificamente? Sim ou não? Por quê?

6.5 Qualquer implementação do ARP que use um cache de tamanho fixo pode falhar quando usada em uma rede que possui muitos hosts e muito tráfego ARP. Explique como.

6.6 ARP normalmente é citado como um ponto fraco na segurança. Explique por quê.

6.7 Suponha que a máquina C receba uma requisição ARP enviada de A, procurando o destino B, e suponha que C tenha o vínculo de I_B para H_B em seu cache. C deve responder à requisição? Explique.

6.8 ARP pode pré-construir um cache para todos os hosts possíveis em uma rede Ethernet pela iteração do conjunto de possíveis endereços IP e envio de uma solicitação ARP para cada um deles. Fazer isso é uma boa ideia? Sim ou não? Por quê?

6.9 A revalidação antecipada deve enviar uma requisição para todos os endereços IP possíveis na rede local, todas as entradas no cache ARP ou somente para destinos que experimentaram tráfego recentemente? Explique.

6.10 Como uma estação de trabalho pode usar ARP quando inicializa para descobrir se qualquer outra máquina na rede a está personificando? Quais são as desvantagens desse esquema?

6.11 Explique como o envio de pacotes IPv4 para endereços não existentes em uma Ethernet remota pode gerar o tráfego de broadcast nessa rede.

6.12 Suponha que um determinado comutador Ethernet conecta 4095 hosts e um roteador. Se 99% de todo o tráfego são enviados entre os hosts individuais e do roteador, o ARP ou NDP faz incorrer em mais despesas gerais?

6.13 Responda à pergunta anterior para o caso em que o tráfego é distribuído uniformemente entre pares aleatórios de hosts.

---

* Uma seção mais para frente descreve RARP, um protocolo que usa o mesmo formato de mensagem que o ARP.

** O formato de pacote ARP pode ser encontrado na página 73.

* Ver Capítulo 22 para uma discussão sobre NDP.

** Ver Capítulo 9 para uma discussão sobre ICMPv6.

CAPÍTULO 7

Protocolo de Internet: entrega de datagrama sem conexão (IPv4, IPv6)

CONTEÚDOS DO CAPÍTULO

7.1 Introdução

Os capítulos anteriores revisaram partes do hardware e software de rede que possibilitam a comunicação na internet, explicando as tecnologias de rede subjacentes e a resolução de endereço. Este capítulo explica o princípio fundamental da entrega sem conexão e discute como ele é fornecido pelo *Internet Protocol* (*IP*), que é um dos dois principais protocolos usados nas internets (TCP sendo o outro). Estudaremos o formato dos pacotes IP e veremos como eles formam a base para toda a comunicação na internet. Os próximos dois capítulos continuarão nosso exame do Internet Protocol, discutindo o encaminhamento de pacotes e tratamento de erros.

7.2 Uma rede virtual

O Capítulo 3 discute a arquitetura da internet em que os roteadores conectam várias redes físicas. A verificação da arquitetura pode ser enganosa, pois o foco da tecnologia de internet é a abstração que uma internet fornece para aplicações e usuários, e não a tecnologia de interconexão subjacente.

A tecnologia da internet apresenta a abstração de uma única rede virtual que interliga todos os hosts, e através da qual a comunicação é possível. A arquitetura subjacente é tanto escondida como irrelevante.

De certa forma, uma internet é uma abstração de uma grande rede física porque, no nível mais inferior, tem a mesma funcionalidade: aceitar e entregar pacotes. Níveis mais altos de software de internet acrescentam a maior parte da funcionalidade rica que os usuários percebem.

7.3 Arquitetura e filosofia da Internet

Conceitualmente, uma internet TCP/IP provê três conjuntos de serviços. A Figura 7.1 lista as três categorias e ilustra a dependência entre elas.

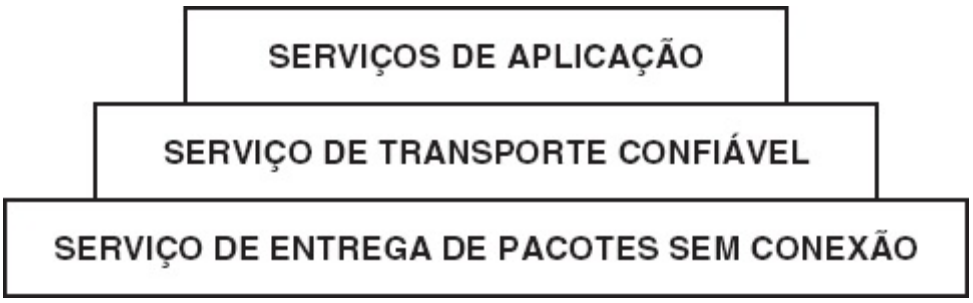


Figura 7.1 Os três níveis conceituais dos serviços de internet.

No nível mais baixo, um serviço de entrega sem conexão provê um alicerce no qual tudo se apoia. No nível seguinte, um serviço de transporte confiável oferece uma plataforma de nível superior da qual as aplicações dependem. Logo, exploraremos cada um desses serviços, entenderemos o que eles oferecem e veremos os protocolos associados a eles.

7.4 Princípios por trás da estrutura

Embora possamos associar software de protocolo a cada um dos níveis na Figura 7.1, o motivo para identificá-los como partes conceituais da tecnologia TCP/IP da Internet é que eles claramente indicam dois fundamentos filosóficos do projeto. Primeiro, a figura mostra que o projeto constrói serviços confiáveis no topo de uma não confiável base sem conexão. Segundo, ela mostra como o projeto foi tão amplamente aceito: o serviço em nível mais baixo ajusta perfeitamente as facilidades fornecidas pelo hardware subjacente de rede e o segundo nível fornece o serviço que o aplicativo espera.

Os três níveis de conceito explicam muito do sucesso da Internet; como consequência do projeto básico, a tecnologia da Internet tem sido surpreendentemente robusta e adaptável. O serviço sem conexão ocorre em

hardware de rede arbitrário, e o serviço de transporte confiável tem sido suficiente para uma ampla variedade de aplicações. Podemos fazer o resumo a seguir.

Protocolos de Internet são projetados em torno de três níveis conceituais de serviço. Um serviço de conexão no nível mais baixo corresponde bem a hardware subjacente, um serviço de transporte confiável fornece o serviço a aplicativos, e uma variedade de aplicações fornece os serviços que os usuários esperam.

O projeto na Figura 7.1 é significativo porque representa uma mudança dramática do pensamento anterior sobre comunicação de dados. As primeiras redes seguiram a abordagem de construir confiabilidade em cada nível. Os protocolos de Internet são organizados para começar com um serviço básico de entrega de pacotes e, em seguida, adicionar a confiabilidade. Quando o projeto foi inicialmente proposto, muitos profissionais duvidaram de que ele pudesse funcionar.

Uma vantagem da separação conceitual é que ela permite um serviço a ser melhorado ou substituído sem perturbar os outros. No início de Internet, pesquisa e desenvolvimento evoluíram simultaneamente em todos os três níveis. A separação será especialmente importante durante a transição do IPv4 para o IPv6, pois permite que os protocolos da camada superior e aplicações se mantenham inalterados.

7.5 Características do sistema de entrega sem conexão

O serviço de Internet mais fundamental consiste em um sistema de entrega de pacotes. Tecnicamente, o serviço é definido como "esforçado", mas não confiável sistema de entrega de pacotes sem conexão. O serviço é análogo ao serviço prestado pela maioria dos hardwares de rede, porque as tecnologias de comutação de pacotes, tais como Ethernet, operam em um paradigma de entrega de melhor esforço. Nós usamos o termo técnico *não confiável* para dizer que a entrega não é garantida. Um pacote pode ser perdido, duplicado, atrasado, ou entregue fora de ordem. O serviço sem conexão não detectará tais condições, nem informará ao remetente ou ao destinatário. O serviço básico é classificado como *sem conexão* porque cada pacote é tratado de forma independente de todos os outros. Uma

sequência de pacotes enviados de um computador para outro pode percorrer caminhos diferentes, ou alguns podem ser perdidos enquanto outros são entregues. Finalmente, diz-se que o serviço usa a *entrega de melhor esforço* porque o software Internet faz uma tentativa séria para entregar os pacotes. Ou seja, a Internet não descarta pacotes caprichosamente; insegurança surge apenas quando os recursos se esgotam ou as redes subjacentes falham.

7.6 Propósito e importância do protocolo de Internet

O protocolo que define o mecanismo de entrega sem conexão não confiável é chamado de *Protocolo de Internet* (*IP*). Vamos seguir a convenção usada em documentos de padrões usando os termos *Protocolo de Internet* e *IP* quando as declarações se aplicam amplamente, e usar *IPv4* ou *IPv6* apenas quando um detalhe especial é aplicado a uma versão, mas não a outra.

O Protocolo de Internet fornece três especificações importantes. Primeiro, o IP define a unidade básica de transferência de dados usada em toda a internet TCP/IP. Assim, ele especifica o formato exato do pacote usado para todos os dados, enquanto os dados passam através de uma internet. Em segundo lugar, o software de IP executa a função de *encaminhamento*, escolhendo um caminho ao longo do qual um pacote será enviado. As normas especificam como o encaminhamento é realizado. Em terceiro lugar, além da especificação precisa e formal de formatos de dados e encaminhamento, o IP inclui um conjunto de regras que incorporam a base de entrega não confiável. As regras definem como hosts e roteadores devem processar os pacotes, como e quando as mensagens de erro devem ser geradas e as condições em que os pacotes podem ser descartados. O Protocolo de Internet é uma parte tão fundamental do projeto que a Internet é muitas vezes chamada uma *tecnologia baseada em IP*.

Começamos nossas considerações sobre IP olhando para o formato do pacote que ele especifica. O Capítulo 1 analisa o formato de pacotes IPv4 e, em seguida, considera o formato utilizado com IPv6. Deixaremos para capítulos posteriores os temas de encaminhamento de pacotes e tratamento de erros.

7.7 O datagrama IP

Em uma rede física, a unidade de transferência é um frame que contém um cabeçalho e dados, e esse cabeçalho fornece informações, tais como a origem (física) e endereços de destino. A Internet chama sua unidade de transferência básica de *datagrama Internet*, geralmente abreviado *datagrama IP*.* Na verdade, a tecnologia TCP/IP tornou-se tão bem-sucedida que, quando alguém usa o termo *datagrama* sem qualquer qualificação, é geralmente aceito que significa *datagrama IP*.

A analogia entre um datagrama e um pacote de rede é forte. Como a Figura 7.2 ilustra, um datagrama é dividido em um cabeçalho e carga útil, como um frame típico de rede. Também como um frame, o cabeçalho do datagrama contém metadados, tais como endereços de origem e destino e um campo de tipo que identifica o conteúdo do datagrama. A diferença, claro, é que o cabeçalho do datagrama contém endereços IP, enquanto o cabeçalho do frame contém os endereços de hardware.

Figura 7.2 Forma geral de um datagrama IP, a analogia Internet de um frame de rede.

7.7.1 Formato do datagrama IPv4

Agora que já descrevemos o layout geral de um datagrama IP, podemos olhar para o conteúdo com mais detalhes. A Figura 7.3 mostra a disposição de campos em um datagrama IPv4. Os próximos parágrafos discutem alguns dos campos de cabeçalho; seções posteriores à fragmentação e as opções de cobrir os campos restantes.

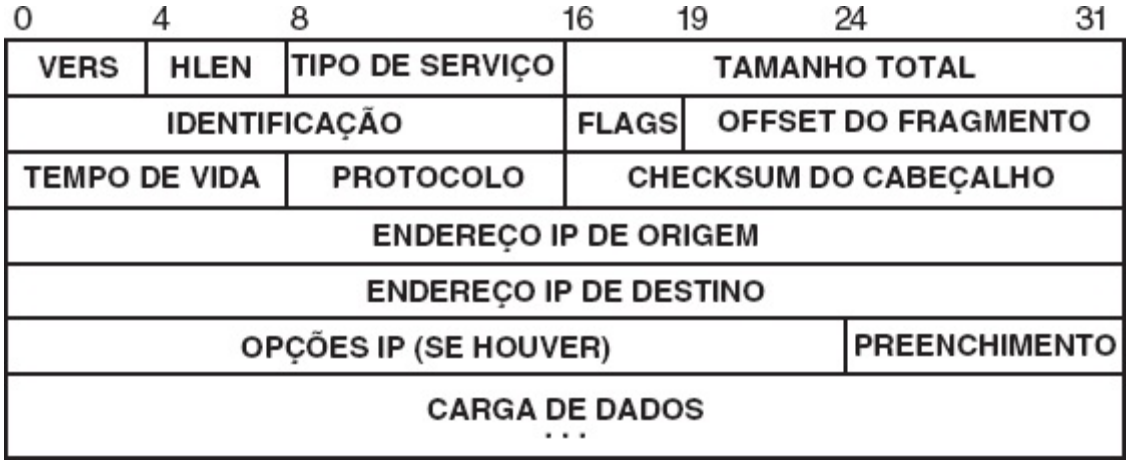


Figura 7.3 Formato de um datagrama IPv4, a unidade básica de transferência em uma internet TCP/IP.

Porque uma internet é virtual, o conteúdo e o formato não são limitados pelo hardware de rede. Por exemplo, o primeiro campo de 4 bits num datagrama (VERS) contém a versão do protocolo IP que foi utilizado para criar o datagrama. Assim, para o IPv4, o campo versão contém o valor *4*. O campo é utilizado para verificar se o remetente, o receptor e quaisquer roteadores entre eles concordam com o formato do datagrama. Todo software IP é necessário para verificar o campo de versão antes de processar um datagrama para garantir que corresponde ao formato que o software espera. Veremos que, embora o cabeçalho do datagrama IPv6 seja diferente do cabeçalho IPv4, o IPv6 usa também os primeiros quatro bits para um número de versão, tornando possível para um roteador ou computador host distinguir entre as duas versões. Em geral, um computador irá rejeitar qualquer datagrama se não tiver o software para lidar com a versão especificada no datagrama. Isso impede que computadores façam má interpretação do conteúdo dos datagramas ou apliquem um formato desatualizado.

O campo de comprimento do cabeçalho (HLEN), também de 4 bits, dá o comprimento do cabeçalho do datagrama medido em palavras de 32 bits. Como veremos, todos os campos do cabeçalho têm comprimento fixo, exceto para as *OPÇÕES DE IP* e campos de *preenchimento (PADDING)* correspondentes. O cabeçalho do datagrama mais comum, que não contém opções e nenhum preenchimento, mede 20 octetos e tem um tamanho de campo de cabeçalho igual a 5.

O campo TAMANHO TOTAL dá o comprimento do datagrama IP medido em octetos, incluindo octetos de cabeçalho e de carga útil. O tamanho da área de carga útil pode ser calculado subtraindo o comprimento de cabeçalho (trinta e duas vezes HLEN) do COMPRIMENTO TOTAL. Como o campo de COMPRIMENTO TOTAL é de 16 bits, o tamanho máximo possível de um datagrama IP é 2^{16} ou 65.535 octetos. Para a maioria das aplicações, o limite não representa um problema. Na verdade, a maioria das tecnologias de rede subjacentes usam tamanhos de frame muito menores; vamos discutir a relação entre o tamanho do datagrama e o tamanho do frame mais tarde.

O campo PROTOCOLO é análogo ao campo de tipo em um frame de rede; o valor especifica qual protocolo de alto nível foi utilizado para criar a mensagem transmitida na área de CARGA ÚTIL do datagrama. Em

essência, o valor do PROTOCOLO especifica o formato da área de CARGA ÚTIL. O mapeamento entre um protocolo de alto nível e o valor inteiro utilizado no campo PROTOCOLO deve ser administrado por uma autoridade central para garantir acordo em toda a Internet.

O campo HEADER CHECKSUM garante a integridade dos valores do cabeçalho. O header checksum IP é formado tratando o cabeçalho como uma sequência de inteiros de 16 bits (em ordem de byte de rede), juntando-os, usando um complemento aritmético e, em seguida, obtendo do resultado o complemento. Para fins de cálculo do checksum, o campo HEADER CHECKSUM é assumido como contendo zeros.

É importante notar que o checksum só se aplica para valores no cabeçalho IP e para a carga. Separar os checksums para cabeçalhos e para cargas apresenta vantagens e desvantagens. Como o cabeçalho geralmente ocupa menos octetos do que a capacidade de carga, ter um checksum separado reduz o tempo de processamento em roteadores que só precisam calcular checksum de cabeçalho. A separação também permite que os protocolos de nível superior escolham seu próprioesquema de checksum para as mensagens que enviam. A principal desvantagem é que protocolos de nível superior são obrigados a adicionar seus próprios despercebida.

Campos de ENDEREÇO IP DE ORIGEM e ENDEREÇO IP DE DESTINO contêm os endereços IP de 32 bits do remetente do datagrama e do destinatário desejado. Embora o datagrama possa ser transmitido através de vários roteadores intermediários, os campos de origem e de destino nunca mudam; eles especificam os endereços IP da fonte original e de destino final.* Note que os endereços de roteadores intermediários não aparecem no datagrama. A ideia é fundamental para a concepção global é descrita a seguir.

> *O campo de endereço de origem em um datagrama sempre se refere à fonte original, e o campo de endereço de destino refere-se ao último destino.*

O campo denominado CARGA DE DADOS, na Figura 7.3, mostra apenas o início da área do datagrama que transporta os dados. O tamanho da carga depende, é claro, do que está sendo enviado no datagrama. O campo OPÇÕES IP, discutido a seguir, é de comprimento variável. O campo PREENCHIMENTO depende das opções selecionadas. Ela representa os bits contendo zero, que podem ser necessários para garantir ao cabeçalho

do datagrama se estender a um múltiplo exato de 32 bits. (Lembre-se de que o campo de tamanho de cabeçalho é especificado em unidades de palavras de 32 bits.)

7.7.2 Formato do datagrama IPv6

O IPv6 revê completamente o formato do datagrama substituindo seu cabeçalho IPv4. Em vez de tentar especificar todos os detalhes em um único cabeçalho, o IPv6 usa uma capacidade de extensão que permite que o IETF adapte o protocolo. A Figura 7.4 ilustra o conceito: um datagrama IPv6 começa com um *cabeçalho-base* de tamanho fixo, seguido por zero, ou mais *cabeçalhos de extensão*, seguidos por uma carga de dados.

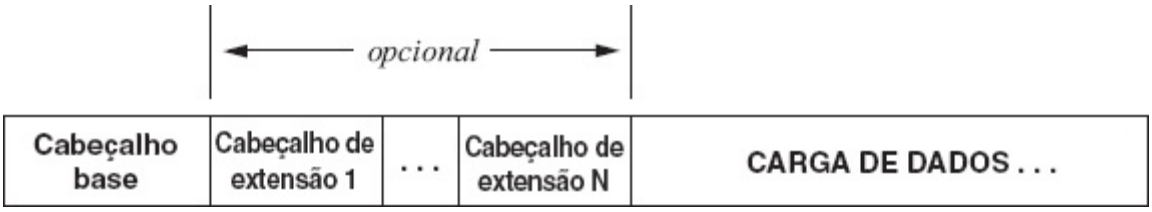


Figura 7.4 A forma geral de um datagrama IPv6 com um cabeçalho-base seguido por cabeçalhos de extensão opcionais.

Como pode um receptor saber quais cabeçalhos de extensão foram incluídos em um determinado datagrama? Cada cabeçalho IPv6 contém um campo PRÓXIMO CABEÇALHO (NEXT HEADER) que especifica o tipo de cabeçalho que vem a seguir. O último cabeçalho utiliza o campo PRÓXIMO CABEÇALHO para especificar o tipo de carga útil. A Figura 7.5 ilustra o uso de campos PRÓXIMO CABEÇALHO.

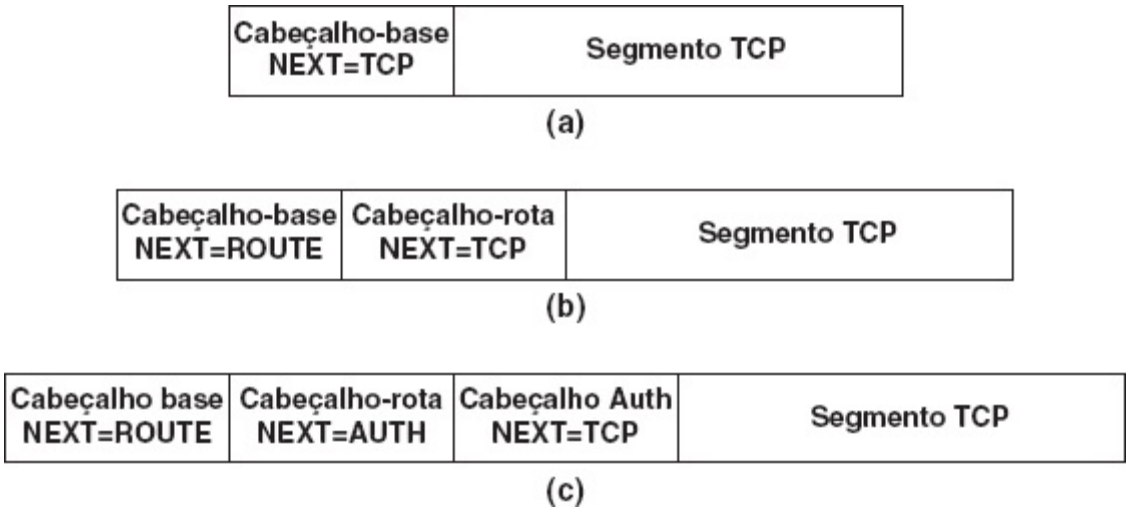


Figura 7.5 Ilustração dos campos NEXT HEADER nos datagramas IPv6 com (a) somente um cabeçalho-base, (b) um cabeçalho-base e uma extensão e (c) um cabeçalho-base e dois cabeçalhos de extensão.

O paradigma de cabeçalho-base fixo seguido por um conjunto de cabeçalhos de extensão foi escolhido como um compromisso entre generalidade e eficiência. Para ser totalmente geral, o IPv6 precisa incluir mecanismos para apoiar funções como fragmentação, roteamento de origem e autenticação. No entanto, escolher alocar campos fixos no cabeçalho do datagrama para todos os mecanismos é ineficiente, já que a maioria dos datagramas não usa todos os mecanismos; o grande tamanho do endereço IPv6 agrava a ineficiência. Por exemplo, ao enviar um datagrama através de uma única rede de área local, um cabeçalho que contém campos de endereço não utilizados pode ocupar uma fração substancial de cada frame. Mais importante, os desenvolvedores perceberam que ninguém pode prever que instalações serão necessárias. Portanto, optaram por cabeçalhos de extensão como forma de proporcionar generalidade sem forçar todos os datagramas a terem grandes cabeçalhos.

Alguns dos cabeçalhos de extensão são destinados ao processamento pelo destino final e alguns dos cabeçalhos de extensão são usados por roteadores intermediários ao longo do caminho. Observe que o uso de campos de PRÓXIMOS CABEÇALHOS significa que as extensões são processadas sequencialmente. Para acelerar o processamento, o IPv6 exige cabeçalhos de extensão que são usados por roteadores intermediários precedendo cabeçalhos de extensão utilizados pelo destino final. Usamos o termo *cabeçalho hop-by-hop* (salto a salto) para nos referirmos a um cabeçalho de extensão que um roteador intermediário deve processar. Assim, os cabeçalhos hop-by-hop precedem cabeçalhos end-to-end (fim a fim).

7.7.3 Formato-base de cabeçalho IPv6

Cada datagrama IPv6 começa com um cabeçalho-base de 40-octetos, como ilustra a Figura 7.6. Apesar de ser duas vezes maior que um cabeçalho típico de datagrama IPv4, o cabeçalho base IPv6 contém menos informação, pois informações fragmentadas foram movidas para cabeçalhos de extensão. Além disso, o IPv6 muda o alinhamento de múltiplos de 32-bit para múltiplos de 64-bit.

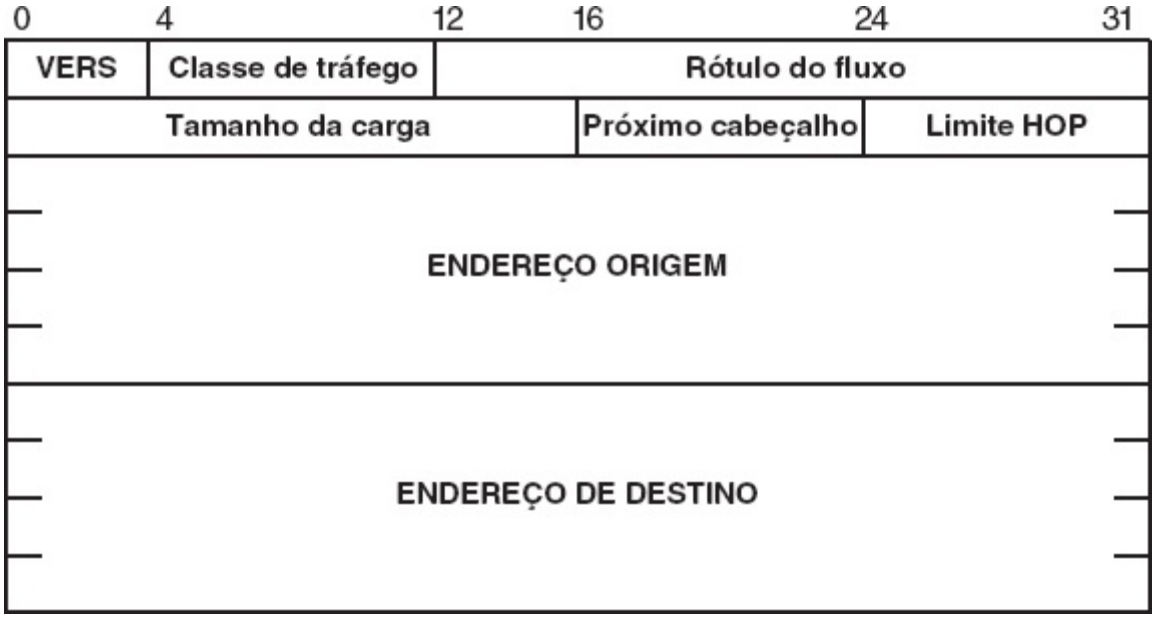


Figura 7.6 O formato-base de cabeçalho; o tamanho é fixado em 40 octetos.

Como no IPv4, o campo inicial VERS de 4 bits especifica a versão do protocolo; *6* especifica um datagrama IPv6. Conforme descrito a seguir, o campo TRAFFIC CLASS (CLASSE DE TRÁFEGO) é interpretado exatamente como o campo TIPO DE SERVIÇO IPv4. O campo FLOW LABEL (RÓTULO DO FLUXO) se destina a permitir que o IPv6 seja usado com tecnologias que suportam a reserva de recursos. A abstração subjacente, um *fluxo*, consiste em um caminho através de uma internet. Roteadores intermediários ao longo do trajeto garantem para os pacotes uma qualidade de serviço específico sobre o fluxo. O RÓTULO DE FLUXO tem uma identificação que permite que um roteador o identifique no fluxo, que é usado em vez do endereço de destino ao encaminhar um datagrama. O Capítulo 16 explica com mais detalhe as potenciais utilizações de um rótulo de fluxo. O IPv6 utiliza um campo de TAMANHO DE CARGA ÚTIL, em vez de um campo de comprimento do datagrama; a diferença é que TAMANHO DE CARGA ÚTIL refere-se apenas aos dados que estão sendo transportados e não inclui o tamanho do cabeçalho-base ou do(s) cabeçalho(s) de extensão. Para permitir a uma carga exceder 2^{16} octetos, o IPv6 define um cabeçalho de extensão que especifica que um datagrama é um *jumbogram*. Um campo de PRÓXIMO CABEÇALHO aparece em todos os cabeçalhos (o cabeçalho-base, como mostrado e cada cabeçalho de extensão); o campo especifica o tipo do próximo cabeçalho de extensão e, no cabeçalho final, dá o tipo de carga útil. O campo HOP LIMIT especifica

o número máximo de redes que o datagrama pode percorrer antes de ser descartado. Por fim, os campos de ENDEREÇO DE ORIGEM e de ENDEREÇO DE DESTINO especificam os endereços IPv6 do remetente original e de destino final.

7.8 Tipo de serviço de datagrama e serviços diferenciados

Informalmente chamado de *Tipo de Serviço – Type Of Service (TOS)* –, o campo TIPO SERVIÇO de 8 bits em um cabeçalho IPv4 e o campo TRÁFEGO DE CLASSE em um cabeçalho IPv6 especificam como o datagrama deve ser tratado. No IPv4, o campo foi originalmente dividido em subcampos que especificam a precedência do datagrama e as características do caminho desejado (baixo atraso ou alto rendimento). No final dos anos 1990, o IETF redefiniu o significado do campo para acomodar um conjunto de serviços diferenciados – *differentiated services (DiffServ)*. A Figura 7.7 ilustra a definição resultante que se aplica ao IPv6, bem como ao IPv4.

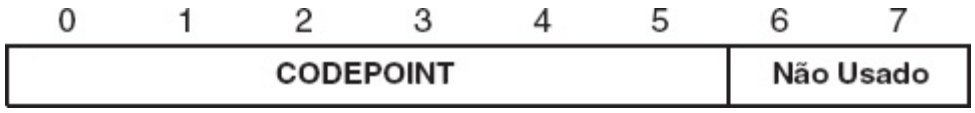


Figura 7.7 Os serviços diferenciados (DiffServ) de interpretação de bits no TIPO SERVIÇO IPv4 e campos de cabeçalho CLASSE DE TRÁFEGO IPv6.

Sobre o DiffServ, os primeiros seis bits do campo constituem um *codepoint*, que é por vezes abreviado *DSCP*, e os últimos dois bits não são utilizados. Um valor codepoint mapeia uma definição de serviço subjacente, normalmente por meio de uma série de indicadores. Embora seja possível definir 64 serviços distintos, os desenvolvedores sugerem que um determinado roteador só vai precisar de alguns serviços, e vários codepoints mapearão cada serviço. Por exemplo, um roteador pode ser configurado com um serviço *de voz*, um serviço *de vídeo*, um serviço de *gestão de rede* e um serviço *normal de dados.* Para manter compatibilidade com a definição original, as normas distinguem entre os primeiros três bits do codepoint (bits que eram usados anteriormente para precedência) e os três últimos bits. Quando os últimos três bits contêm

zero, os bits de precedência definem oito grandes classes de serviço que aderem às mesmas linhas mestras, como a definição original: datagramas com um número maior em seu campo precedência têm tratamento preferencial com relação a datagramas com um número menor. Ou seja, as classes de ordem oito são definidas por valores codepoint da seguinte forma:

xxx000

onde **x** pode ser tanto zero como um.

O projeto de serviços diferenciados também acomoda outra prática existente – o uso generalizado de precedência 6 ou 7 para dar maior prioridade ao roteamento de tráfego. O padrão inclui um caso especial para lidar com os dois valores de precedência. É necessário que um roteador implemente pelo menos dois esquemas de prioridade: um para o tráfego normal e um para o tráfego de alta prioridade. Quando os três últimos bits do campo CODEPOINT são zero, o roteador deve mapear um codepoint com precedência 6 ou 7 para a classe de maior prioridade e outros valores codepoint para a classe de menor prioridade. Assim, se um datagrama que chega foi enviado com o esquema de TOS original, um roteador, usando o esquema de serviços diferenciados, honrará precedência 6 e 7, como o remetente do datagrama espera.

A Figura 7.8 ilustra como o codepoint de valor 64 é dividido em três agrupamentos administrativos.

Grupo	Codepoint	Atribuído por
1	xxxxx0	Organização de normas
2	xxxx11	Local ou experimental
3	xxxx01	Local ou experimental

Figura 7.8 Três grupos administrativos de valores codepoint DiffServ.

Como a figura indica, a metade dos valores (ou seja, os 32 valores no grupo *1*) deve ter interpretações atribuídas pela IETF. Atualmente, todos os valores no grupo *2* e *3* estão disponíveis para uso experimental ou local. No entanto, o grupo *3* é provisório – se os organismos de normalização esgotarem todos os valores no grupo *1*, eles vão deixar o grupo *2* sozinho, mas podem também optar por atribuir valores no grupo *3*.

A divisão em grupos pode parecer incomum, pois depende dos bits de baixa ordem do valor para distinguir grupos. Assim, em vez de um conjunto contíguo de valores, o grupo *1* contém todos os outros valores codepoint (ou seja, os números pares entre 2 e 64). A divisão foi escolhida para manter os oito codepoints correspondentes aos valores xxx000 no mesmo grupo.

Se a interpretação TOS original ou a interpretação revisada de serviços diferenciados é usada, é importante perceber que o software de encaminhamento deve escolher entre as tecnologias subjacentes de rede física disponíveis e aderir a políticas locais. Desse modo, especificar um nível de serviço em um datagrama não garante que os roteadores ao longo do caminho vão concordar em honrar o pedido. Podemos então fazer o resumo a seguir.

Consideramos a especificação do tipo de serviço como uma dica para o algoritmo de encaminhamento que o ajuda a escolher entre vários caminhos para um destino com base nas políticas locais e em seu conhecimento das tecnologias de hardware disponíveis nesses caminhos. Uma internet não garante fornecer qualquer tipo particular de serviço.

7.9 Encapsulamento de datagrama

Antes que possamos entender os próximos campos em um datagrama, é importante considerar como os datagramas se relacionam com os frames da rede física. Começamos com uma pergunta: que tamanho um datagrama pode ter? Diferente dos frames de rede física, que precisam ser reconhecidos pelo hardware, os datagramas são tratados por software. Eles podem ter qualquer tamanho escolhido pelos projetistas de protocolo. Vimos que o formato do datagrama IPv4 aloca 16 bits ao campo de tamanho total, limitando o datagrama a, no máximo, 65.535 octetos.

Na prática, surgem limites mais fundamentais no tamanho do datagrama. Sabemos que, quando os datagramas passam de uma máquina para outra, eles sempre precisam ser transportados pelo hardware da rede física subjacente. Para tornar o transporte na internet eficiente, é preciso garantir que cada datagrama trafegue em um frame de rede distinto. Ou seja, queremos que nossa abstração do pacote da rede seja, se possível, mapeada diretamente para um pacote real.

A ideia de transportar um datagrama em um frame de rede é chamada de *encapsulamento* e é usada tanto com o IPv4 como com o IPv6. Para a rede subjacente, um datagrama é como qualquer outra mensagem enviada de uma máquina para outra. O hardware não reconhece o formato do datagrama nem entende o endereço de destino IP. Pelo contrário, a rede trata um datagrama como bits de dados a serem transferidos. A Figura 7.9 ilustra a ideia: quando uma máquina envia um datagrama IP para outra, o datagrama inteiro trafega na parte de carga de dados do frame da rede.

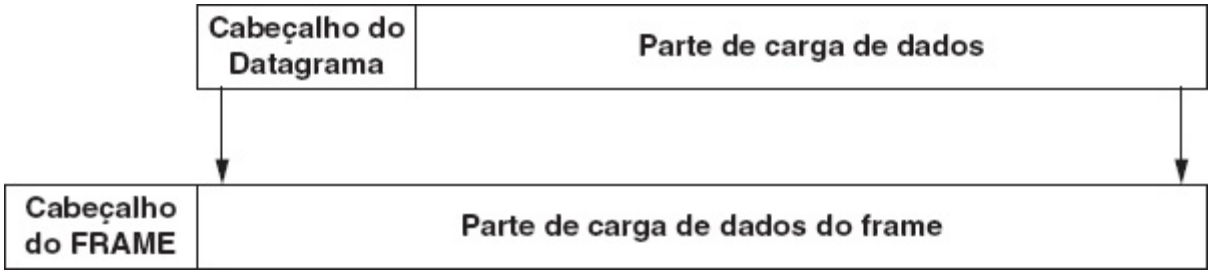


Figura 7.9 O encapsulamento de um datagrama IP em um frame. A rede subjacente trata todo o datagrama, inclusive o cabeçalho, como dados.

Como um receptor sabe que a parte de carga em um frame contém um datagrama IP? O campo de tipo no cabeçalho do frame identifica os dados que estão sendo transportados. Por exemplo, a Ethernet usa o valor tipo 0x0800 para especificar que a carga de dados contém um datagrama IPv4 encapsulado e 0x86DD para especificar que a carga de dados contém um datagrama IPv6.

7.10 Tamanho de datagrama, rede MTU e fragmentação

No caso ideal, o datagrama IP inteiro se encaixa em um frame físico, tornando eficiente a transmissão pela rede subjacente. Para alcançar essa eficiência, os desenvolvedores do IP poderiam ter selecionado um tamanho máximo de datagrama, de modo que um datagrama sempre encaixasse em um frame. Mas que tamanho de frame deveria ser escolhido? Afinal, um datagrama pode trafegar por muitos tipos de redes enquanto passa por uma internet desde sua origem até o seu destino final.

Para entender o problema, precisamos de um fato sobre o hardware da rede: cada tecnologia de comutação de pacotes impõe um limite superior fixo sobre a quantidade de dados que podem ser transferidos em um frame. Por exemplo, a Ethernet limita as transferências a 1.500 octetos de dados.*

Vamos nos referir ao tamanho limite como a *unidade de transferência máxima, unidade de transmissão máxima* (*MTU*) da rede. Os tamanhos de MTU podem ser maiores ou menores do que 1500: tecnologias como IEEE 802.15.4 limitam as transferências a 128 octetos. Limitar os datagramas para caberem na menor MTU possível na internet torna as transferências ineficientes. A ineficiência é especialmente severa, pois a maior parte dos caminhos na Internet pode transportar datagramas maiores. Porém escolher um tamanho maior causa outro problema: como o hardware não permitirá pacotes maiores que a MTU, não poderemos enviar datagramas grandes em um único frame de rede.

Dois princípios de projeto de internet fundamentais para nos ajudar a entender o dilema são descritos a seguir.

> *A tecnologia de internet deve acomodar a maior variedade possível de hardwares de rede.*
>
> *A tecnologia de internet deve acomodar a maior variedade possível de aplicações de rede.*

O primeiro princípio implica que não devemos descartar uma tecnologia de rede simplesmente porque ela tem uma MTU pequena. O segundo princípio sugere que os programadores de aplicativos devem ser autorizados a escolherem o tamanho de datagrama que achem apropriado.

Para satisfazer ambos os princípios, os protocolos TCP/IP usam um compromisso. Em vez de restringir o tamanho do datagrama a priori, as normas permitem a cada aplicativo escolher um tamanho de datagrama que seja mais adequado para a aplicação. Em seguida, quando da transferência de um datagrama, verificar seu tamanho para ver se é menor do que a MTU. Se o datagrama não couber em um frame, dividi-lo em pedaços menores chamados *fragmentos* e escolher o tamanho de fragmento de modo que cada um possa ser enviado em um frame de rede. O processo de dividir um datagrama é conhecido como fragmentação.

Para entender a fragmentação, considere três redes interconectadas por dois roteadores, como mostra a Figura 7.10.

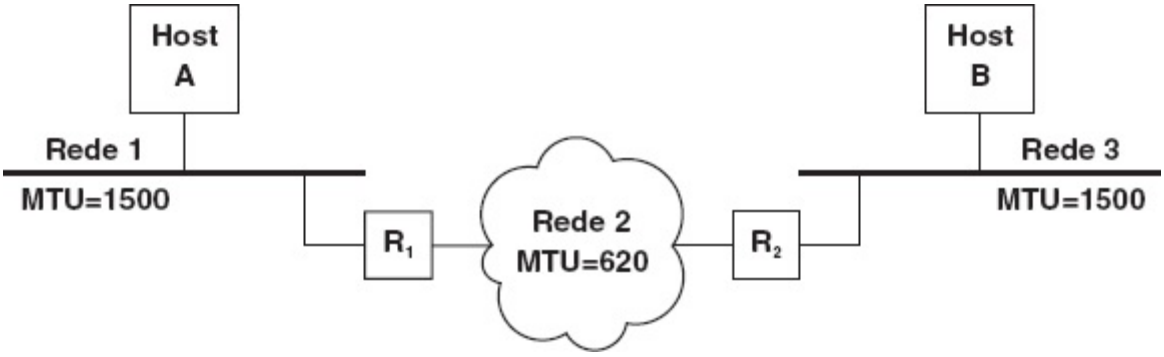


Figura 7.10 Uma ilustração de fragmentação IPv4. Cada roteador pode precisar fragmentar os datagramas antes de enviá-los pela rede 2.

Na figura, cada host se conecta diretamente a uma Ethernet, que tem uma MTU de 1500 octetos. A norma exige que os roteadores aceitem datagramas até o máximo de MTU das redes às quais estão conectados. Assim, o host pode criar e enviar um datagrama de até 1500 octetos para a MTU da rede diretamente conectada. Se um aplicativo em execução no host *A* envia um datagrama de 1500-octetos para *B*, o datagrama pode viajar através de rede 1 em um único frame. No entanto, porque a rede 2 tem uma MTU de 620, é necessária a fragmentação para o datagrama viajar através da rede 2.

Além de definir a MTU de cada rede individual, será importante considerá-la ao longo de um caminho através de uma internet. O *caminho MTU* é definido como sendo o mínimo das MTU em redes ao longo do caminho. Na figura, o percurso de *A* para *B* tem um *caminho MTU* de 620.

Embora cada um deles forneça fragmentação de datagrama, o IPv4 e o IPv6 assumem abordagens completamente diferentes. O IPv4 permite que qualquer roteador ao longo de um caminho fragmente um datagrama. De fato, se um roteador, mais tarde, ao longo do caminho considerar que um fragmento é muito grande, ele pode dividir esse fragmento em fragmentos ainda menores. O IPv6 exige que a fonte original conheça o caminho MTU e realize a fragmentação; roteadores são proibidos de realizar fragmentação. As próximas seções vão considerar as duas abordagens e dar os detalhes para o IPv4 e o IPv6

7.10.1 Fragmentação de datagrama IPv4

No IPv4, a fragmentação é adiada e realizada somente quando necessário. Se um datagrama vai ser fragmentado, depende do caminho que ele segue

através de uma internet. Ou seja, a origem só garante que um datagrama pode caber em um frame da primeira rede que ele deve percorrer. Cada roteador ao longo do caminho olha para o MTU da próxima rede na qual o datagrama deve passar e, se necessário, o fragmenta. Na Figura 7.10, por exemplo, o roteador R_1 vai fragmentar um datagrama de 1500 octetos antes de enviá-lo pela rede 2.

Dissemos que um host deve garantir que um datagrama pode caber em um frame da primeira rede. Os aplicativos geralmente tentam escolher um tamanho de mensagem que é compatível com a rede subjacente. No entanto, se um aplicativo optar por enviar um datagrama grande, o software do IP no host pode executar a fragmentação antes de enviá-lo. Na Figura 7.10, por exemplo, se um aplicativo no host *A* cria um datagrama maior do que 1500 octetos, o software IP no host fragmenta o datagrama antes de enviá-lo. O ponto importante é descrito a seguir.

A fragmentação IPv4 ocorre automaticamente em qualquer ponto ao longo do caminho, quando um datagrama é muito grande para a rede através da qual ele deve passar; a origem só precisa garantir que os datagramas podem viajar ao longo do primeiro hop.

Que tamanho deve ter cada fragmento? Dissemos que cada fragmento deve ser pequeno o suficiente para caber em um único frame. No exemplo, um fragmento deve ser de 620 octetos, ou menor. Um roteador poderia dividir o datagrama em fragmentos de tamanhos aproximadamente iguais. A maioria dos softwares IPv4 simplesmente extrai uma série de fragmentos que preenchem a MTU (cada um) e, em seguida, envia um último fragmento remanescente de qualquer tamanho que seja.

Você pode se surpreender ao saber que um fragmento de IPv4 usa o mesmo formato de um datagrama IPv4 completo. O campo FLAGS no cabeçalho do datagrama contém um bit que especifica se o datagrama é completo ou um fragmento. Outro bit no campo FLAGS especifica se mais fragmentos ocorrem (ou seja, se um fragmento específico ocupa a parte final do datagrama original). Finalmente, o campo OFFSET no cabeçalho especifica o lugar no datagrama original a que os dados do fragmento pertencem. Um detalhe interessante da fragmentação surge porque o campo OFFSET armazena uma posição em múltiplos de oito octetos. Ou seja, um deslocamento de octeto é calculado multiplicando-se o campo OFFSET por

oito. Como consequência, o tamanho de cada fragmento deve ser escolhido para ser um múltiplo de oito. Portanto, quando se realiza a fragmentação, o IP escolhe o tamanho do fragmento para ser o maior múltiplo de oito que seja menor ou igual ao tamanho da MTU. A Figura 7.11 ilustra a fragmentação IPv4.

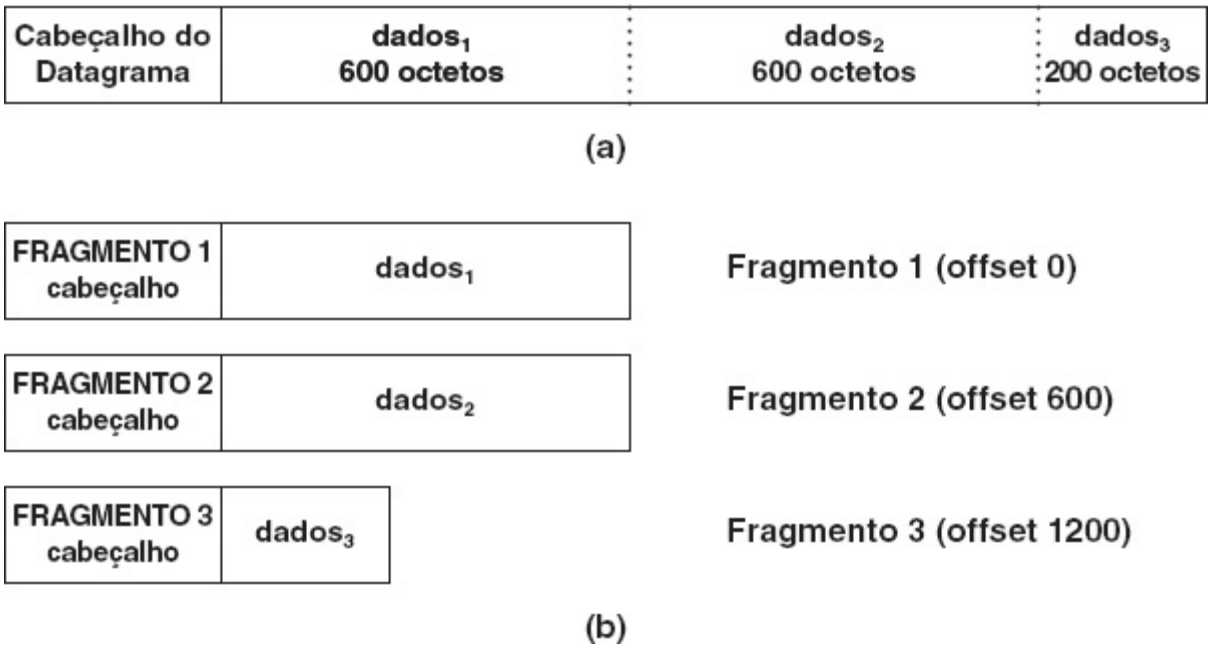


Figura 7.11 (a) Um datagrama original IPv4 transportando 1400 octetos de dados e (b) três fragmentos para uma MTU de 620.

A fragmentação começa por replicar o cabeçalho do datagrama original e, em seguida, modificar os campos FLAGS e OFFSET. Os cabeçalhos nos fragmentos 1 e 2 têm a *maior parte dos fragmentos* de conjunto de bits no campo de FLAGS; o cabeçalho no fragmento 3 tem zero na *maior parte* dos bits *dos fragmentos*. Nota: na figura, os offsets de dados são mostrados como offsets de octeto em decimal; eles devem ser divididos por oito para obter o valor armazenado nos cabeçalhos dos fragmentos.

Cada fragmento contém um cabeçalho de datagrama que duplica a maior parte do cabeçalho do datagrama original (exceto para bits no campo FLAGS que especificam a fragmentação), seguido por tantos dados quantos podem ser transportados no fragmento, mantendo o tamanho total menor do que a MTU da rede na qual eles devem viajar e o tamanho dos dados como múltiplos de oito octetos.

7.10.2 Fragmentação IPv6 e descoberta de caminho MTU (PMTUD)

Em vez de fragmentação atrasada, o IPv6 usa uma forma de ligação inicial: ao host fonte original é requerido encontrar a MTU mínima ao longo do caminho para o destino e fragmentar cada datagrama de acordo com o caminho que vai tomar. Não é permitido aos roteadores IP ao longo do caminho fragmentar datagramas IPv6; se um datagrama não cabe na MTU de uma rede, o roteador envia uma mensagem de erro para a fonte original e o descarta.

Em muitos aspectos, a abordagem IPv6 para a fragmentação é oposta à abordagem IPv4, o que é intrigante. Por que mudar? Quando o IPv6 estava sendo definido, empresas de telefonia estavam empurrando tecnologias *Asynchronous Transfer Mode* (*ATM*), e os desenvolvedores IPv6 assumiram que a ATM se tornaria amplamente utilizada. A ATM é uma tecnologia de conexão orientada, o que significa que um remetente deve preestabelecer um caminho para o destino e, em seguida, enviar por esse caminho. Assim, os desenvolvedores assumiram que um computador de origem iria aprender as características do caminho (incluindo o caminho MTU) quando ele fosse estabelecido e não o mudaria.

Como as tecnologias de rede utilizadas na Internet não informam um host sobre o caminho MTU, um host deve se engajar em um mecanismo de tentativa e erro para determinar esse caminho. Conhecido como *Path MTU Discovery* (*PMTUD*), o mecanismo consiste no envio de um datagrama IPv6 que caiba na MTU da rede diretamente conectada. Se uma rede ao longo do caminho tiver um MTU menor, um roteador irá enviar uma mensagem de erro ICMP para a fonte original que especifica o MTU menor. O host fragmentará o datagrama de acordo com o novo caminho MTU e tentará novamente. Se uma rede posterior ao longo do caminho tiver uma MTU que é ainda menor, outro roteador enviará uma mensagem de erro. Por tentativas repetitivas, um host eventualmente encontrará a menor MTU ao longo do caminho.

O que acontece se o caminho mudar e o novo caminho MTU for maior? A fonte não vai saber do aumento porque os roteadores não armazenam estado. Portanto, PMTUD especificará que um host deve sondar periodicamente através do envio de um datagrama maior. Porque não esperamos que rotas mudem com frequência e porque o caminho MTU muda com menos frequência do que as rotas, a maioria das implementações de IPv6 escolhe um período longo de tempo antes de voltar a sondar (por exemplo, 10 minutos).

Lembre-se de que o cabeçalho-base do IPv6 não inclui campos para especificar a fragmentação. Portanto, quando se fragmenta um datagrama IPv6, uma fonte insere um *Fragmento de Cabeçalho de Extensão* em cada fragmento. A Figura 7.12 ilustra o formato.

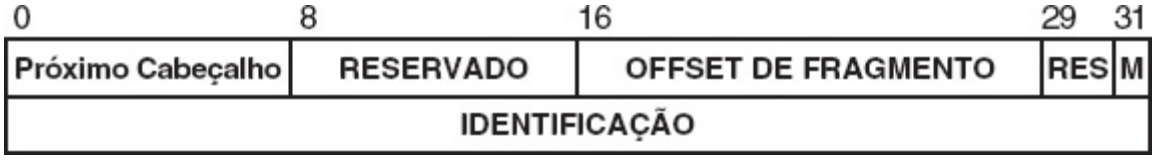


Figura 7.12 O formato de um *cabeçalho de extensão de fragmentação* IPv6.

Como a figura mostra, o cabeçalho de extensão inclui o necessário campo PRÓXIMO CABEÇALHO. Ele também inclui dois campos que são reservados para uso futuro. Os restantes três campos têm o mesmo significado que os campos de controle de fragmentação IPv4. Um campo FRAGMENTO OFFSET de 13 bits especifica a que lugar no datagrama original este fragmento pertence, o *M* bits é um bit *mais fragmentos* que especifica se um fragmento é o fragmento final (mais à direita) do datagrama original, e se o campo de IDENTIFICAÇÃO contém um datagrama ID único que é compartilhado por todos os fragmentos de um datagrama.

7.11 Reconstituição de datagrama

Eventualmente, os fragmentos devem ser *reagrupados* para produzirem uma cópia completa do datagrama original. Surge a pergunta: onde devem ser reagrupados os fragmentos? Ou seja, um datagrama deve ser remontado quando atinge uma rede com uma maior MTU ou deve permanecer fragmentado e os fragmentos devem ser transportados para o destino final? Veremos que a resposta revela outra decisão de projeto.

Em uma internet TCP/IP, uma vez que um datagrama foi fragmentado, os fragmentos devem viajar como datagramas separados por todo o caminho até o destino final, onde são remontados. Preservar fragmentos todo o caminho até o destino final pode parecer estranho, porque a abordagem tem duas desvantagens. Primeiro, se apenas uma rede ao longo do caminho tem um pequeno MTU, o envio de pequenos fragmentos sobre as outras redes é ineficiente, pois o transporte de pequenos pacotes significa mais sobrecarga do que o transporte de grandes pacotes. Assim, mesmo se as redes encontradas após o ponto de fragmentação tiverem MTU

muito grande, o IP irá enviar pequenos fragmentos através deles. Em segundo lugar, se os seus fragmentos são perdidos, o datagrama não pode ser reconstituído. O mecanismo usado para lidar com perda de fragmentos consiste em um *reassembly timer* (timer de reconstituição). Quando um fragmento de um determinado datagrama chega ao destino final, um temporizador se inicia. Se o timer expirar antes de todos os fragmentos chegarem, a máquina que os recebeu descartará os fragmentos sobreviventes. A fonte deve retransmitir todo o datagrama; não há nenhuma maneira de o receptor solicitar fragmentos individuais. Assim, a probabilidade de perda de datagrama aumenta quando ocorre a fragmentação, já que a perda de um único fragmento resulta na perda de todo o datagrama.

Apesar de desvantagens menores, realizar a reconstituição do datagrama no destino final funciona bem. Isso permite que cada fragmento seja transmitido de forma independente. Mais importante, não são necessários roteadores intermediários para armazenar ou remontar fragmentos. A decisão de reconstituir no destino final é derivada de um princípio importante no projeto Internet: o estado em roteadores deve ser minimo.

Na Internet, o destino final reconstitui os fragmentos. O projeto significa que os roteadores não precisam armazenar fragmentos ou manter outras informações sobre pacotes.

7.12 Campos de cabeçalho usados para reconstrução de datagrama

Três campos controlam a reconstituição de datagramas tanto em um cabeçalho de datagrama IPv4 como em um cabeçalho de extensão de fragmento IPv6: IDENTIFICAÇÃO, FLAGS (*M* em IPv6) e OFFSET DE FRAGMENTO. O campo IDENTIFICAÇÃO contém um único inteiro que identifica o datagrama. Ou seja, cada datagrama enviado por uma determinada fonte tem um único ID. Uma implementação típica usa um número sequencial – um computador envia um datagrama com identificação *S*, o próximo datagrama terá identificação *S + 1*. Atribuir uma identificação única para cada datagrama é importante porque a fragmentação começa copiando o número de identificação em cada fragmento. Assim, cada um tem exatamente o mesmo número de

IDENTIFICAÇÃO que o datagrama original. Um destino usa o campo IDENTIFICAÇÃO em fragmentos, juntamente com o endereço de origem do datagrama para agrupar todos os fragmentos de um determinado datagrama. O valor no campo OFFSET FRAGMENTO especifica o deslocamento no datagrama original da carga a ser transportada no fragmento, medida em unidades de oito octetos,* começando no deslocamento zero. Para reconstituir o datagrama, o destino deve obter todos os fragmentos começando com o fragmento offset *0* até o fragmento com o maior offset. Fragmentos não chegam necessariamente em ordem, e não há comunicação entre o sistema que fragmentou o datagrama (um roteador de IPv4 ou o remetente de IPv6) e o destino que está tentando reconstituí-lo.

No IPv4, os dois bits de baixa ordem do campo FLAGS de 3 bits controlam a fragmentação. Normalmente, as aplicações que utilizam TCP/IP não se preocupam com a fragmentação, pois tanto ela como a reconstituição ocorrem em procedimentos automáticos em níveis inferiores da pilha de protocolos, invisíveis para os aplicativos. No entanto, para testar software de rede ou depurar problemas operacionais, pode ser importante determinar o tamanho dos datagramas para os quais ocorre a fragmentação. O primeiro bit de controle auxilia nesse teste, especificando se o datagrama pode ser fragmentado. Ele é chamado bit *não fragmente* porque configurando o bit para *1* se especifica que o datagrama não deve ser fragmentado. Sempre que um roteador precisar fragmentar um datagrama que tem alocado o bit *não fragmente*, o roteador descartará o datagrama e enviará uma mensagem de erro de volta para a fonte.

O bit de baixa ordem no campo FLAGS no IPv4 ou o bit *M* no IPv6 especifica se a carga útil no fragmento pertence a algum lugar no meio do datagrama original ou à sua parte final. Ele é conhecido como bit *mais fragmentos*, porque o valor de *1* significa que a carga em que o fragmento está não é a cauda do datagrama. Como podem chegar fora de ordem, o destino precisa saber quando todos os fragmentos de um datagrama chegaram. Um determinado fragmento não especifica o tamanho do datagrama original, assim, um receptor deve calculá-lo. O bit *mais fragmentos* resolve o problema: uma vez que um fragmento chega com o bit *mais fragmentos* desligado, o destino sabe que o fragmento transporta os dados da parte da cauda do datagrama original. Do campo OFFSET DE

FRAGMENTO e do tamanho do fragmento, o destino pode calcular o comprimento do datagrama original. Assim, uma vez que a cauda do datagrama original chega, o destino pode dizer quando todos os outros fragmentos chegaram.

7.13 Tempo de vida (IPv4) e limite de salto (IPv6)

Originalmente, o campo de cabeçalho TEMPO DE VIDA (TTL) IPv4 especificava por quanto tempo, em segundos, um datagrama estava autorizado a permanecer em uma internet – um remetente definia o tempo máximo que cada datagrama deveria sobreviver, e os roteadores que o processavam diminuíam o TTL conforme o tempo passava. Quando um TTL chegava a zero, o datagrama era descartado.

Infelizmente, calcular um tempo exato é impossível porque os roteadores não sabem o tempo de trânsito para redes subjacentes. Além disso, a noção de datagramas que passam muitos segundos em trânsito tornou-se desatualizada (roteadores e redes atuais são projetados para encaminhar cada datagrama em alguns milissegundos). No entanto, um mecanismo ainda era necessário para lidar com casos em que uma internet tivesse um problema de encaminhamento quando roteadores encaminhassem datagramas em círculos. Para evitar que um datagrama viaje em círculo para sempre, foi adicionada uma regra como um mecanismo à prova de falhas. A regra exige que cada roteador ao longo do caminho da origem ao destino diminua o TTL em *1*. Em essência, cada rede que um datagrama atravessa conta como um *salto de rede* (*hop*). Assim, na prática, o campo TTL é agora usado para especificar o número de saltos que um datagrama pode fazer antes de ser descartado. O IPv6 inclui exatamente o mesmo conceito. Para esclarecer o significado, o IPv6 usa o nome LIMITE DE SALTO, HOP LIMIT,* no lugar de TEMPO DE VIDA, TIME-TO-LIVE.

O Software IP em cada máquina ao longo do caminho, da origem ao destino, diminui o campo TIME-TO-LIVE (IPv4) ou HOP LIMIT (IPv6). Quando o campo atinge zero o datagrama é descartado.

Um roteador faz mais do que simplesmente descartar um datagrama quando o TTL chega a zero – ele envia uma mensagem de erro de volta para a origem. O Capítulo 9 descreve como se lida com o erro.

7.14 Itens IP opcionais

O IPv4 e o IPv6 definem itens opcionais que podem ser incluídos em um datagrama. No IPv4, o campo OPÇÕES IP que segue o endereço de destino é usado para enviar itens opcionais. No IPv6, cada um dos cabeçalhos de extensão é opcional, e um determinado datagrama pode incluir várias extensões.

Na prática, alguns datagramas na Internet global incluem itens opcionais. Muitas das opções nas normas destinam-se a controle especial ou a testes de rede e depuração. Opções de processamento são parte integrante do protocolo IP; todas as implementações padrão devem incluí-las.

As próximas seções discutem opções em IPv4 e IPv6. Como o nosso objetivo é fornecer uma visão conceitual geral, em vez de um catálogo com todos os detalhes, o texto destaca exemplos e discute como cada exemplo pode ser usado.

7.14.1 Opções IPv4

Se um datagrama IPv4 contém opções, elas seguem o campo ENDEREÇO IP DE DESTINO no cabeçalho do datagrama. O tamanho do campo de opções depende de quais opções foram incluídas. Algumas têm um octeto de comprimento e outras têm comprimento variável. Cada opção começa com um *código de opção* de um único octeto que a identifica. Um código de opção pode ser seguido por um tamanho de um único octeto e um conjunto de octetos de dados para essa opção. Quando várias opções estão presentes, eles aparecem de forma contígua, sem separadores especiais entre cada um. Ou seja, a área de opções do cabeçalho é tratada como uma matriz de octetos, e as opções são colocadas na matriz uma após a outra. O bit de alta ordem de um octeto código de opção especifica se a opção deve ser copiada em todos os fragmentos ou somente no primeiro; uma seção posterior que discute opção de processamento explica a cópia.

A Figura 7.13 lista exemplos de opções que podem acompanhar um datagrama IPv4. Como se vê na lista, a maioria das opções é usada para fins de controle. As opções de rota e de timestamp (registro de data e hora) são as mais interessantes, pois fornecem uma maneira de monitorar ou controlar a forma como os roteadores encaminham datagramas.

Record Route Option – (*Opção de Gravação de Rota*). A opção de *rota de registro* permite que a fonte crie uma lista vazia de endereços IPv4 e solicite que cada roteador ao longo do caminho adicione seu endereço

IPv4 nela. A lista começa com um cabeçalho que especifica o tipo da opção, um campo de tamanho e um indicador. O campo de tamanho especifica o número de octetos na lista e o indicador, o deslocamento do próximo item livre. Cada roteador que encaminha o datagrama compara o indicador para o tamanho. Se o indicador for igual ou superior ao tamanho, a lista está cheia. Caso contrário, o roteador coloca o seu endereço IP nos próximos quatro octetos de opção, incrementa o indicador em quatro e encaminha o datagrama.

Número	Tamanho	Descrição
0	1	Fim da lista de opções. Usado se as opções não terminam no final do cabeçalho (ver campo de preenchimento de cabeçalho)
1	1	Nenhuma operação. Usado para alinhar octetos em uma lista
2	11	Restrições de Segurança e manuseio para aplicações militares
3	var	Loose Source Route (Fonte de Rota Livre). Usado para solicitor encaminhamento através de um conjunto de roteadores especificados
4	var	Registro de data e hora Internet. Utilizado para gravar a data e hora em cada salto ao longo do caminho através de uma internet
7	var	Gravação de rota. Faz com que cada roteador ao longo do caminho registre seu endereço IP nas opções do datagrama
9	var	Strict Source Route (Fonte de rota restrita). Usado para especificar um caminho exato através de um conjunto de roteadores
11	4	Tentativa MTU. Usado por um host durante IPv4 Path MTU Discovery
12	4	Resposta MTU. Devolvido pelo roteador durante IPv4 Path MTU Discovery
18	var	Traçador de rota. Usado pelo programa traceroute para encontrar os roteadores ao longo de um caminho
20	var	Alerta de roteador. Faz com que cada roteador ao longo de um caminho examine o datagrama, mesmo que não seja o destino Final

Figura 7.13 Exemplos de opções IPv4 com seus tamanhos e breve descrição de cada uma.

Source Route Options (*Fonte Opções de Rota*). Duas opções, *Strict Source Route* (*Fonte de Rota Restrita*) e *Loose Source Route* (*Fonte de Rota Livre*), fornecem uma maneira para um remetente controlar o encaminhamento ao longo de um caminho por uma internet. Por exemplo, para testar uma determinada rede, o administrador do sistema pode usar as opções de rota de origem para forçar datagramas IP a atravessarem a rede, mesmo se o encaminhamento normal usar outro caminho.

A capacidade de fornecer rotas a pacotes é especialmente importante como uma ferramenta para testes em um ambiente de produção. Ela dá liberdade de gerenciamento da rede para testar uma nova rede experimental, permitindo simultaneamente o tráfego dos usuários ao longo de um caminho que inclui apenas as redes de produção. Claro, fonte de roteamento de origem só é útil para alguém que entende a topologia da rede; um usuário médio não tem nem motivação para considerar fonte de roteamento nem o conhecimento necessário para usá-la.

Strict Source Route (*Fonte de Rota Restrita*). Fonte de rota restrita especifica um caminho completo através de uma internet (ou seja, o caminho que o datagrama deve seguir para chegar ao seu destino). O caminho consiste de endereços IPv4 que correspondem cada um a um roteador (ou para o destino final). A palavra *restrita* significa que cada par de roteadores ao longo do caminho deve ser diretamente conectado por uma rede; se um roteador não puder alcançar o próximo roteador especificado na lista, resultará um erro.

Loose Source Route (*Fonte de Rota Livre*). Fonte de rota livre especifica um caminho através de uma internet, e a opção inclui uma sequência de endereços IP. Ao contrário da fonte de rota restrita, uma fonte de rota livre especifica que o datagrama deve visitar a sequência de endereços IP, mas a rede permite múltiplos saltos entre endereços sucessivos na lista.

Ambas as opções de fonte exigem roteadores ao longo do caminho para substituir itens da lista de endereços com os respectivos endereços de rede local. Assim, quando o datagrama chega ao seu destino, ele contém uma lista de todos os endereços visitados, exatamente como a lista produzida pela opção de registro de rota.

Internet Timestamp Option. A opção de registro de data e hora Internet funciona como o registro de opção de rota: o campo de opção começa com uma lista inicialmente vazia, e cada roteador ao longo do caminho da origem ao destino preenche uma entrada. Ao contrário da opção de registro de rota, cada entrada em uma lista timestamp contém dois valores de 32 bits que estão definidos para o endereço IPv4 do roteador que preencheu a entrada e um inteiro timestamp de 32 bits. Timestamps fornecem a hora e a data em que um roteador lida com o datagrama, sendo a hora expressa em milissegundos desde a meia-noite, hora universal.*

7.14.2 Extensões opcionais IPv6

O IPv6 usa o mecanismo de cabeçalhos de extensão em vez das opções IPv4. A Figura 7.14 lista exemplos de cabeçalhos opcionais IPv6 e explica seus propósitos.

Próximo Cabeçalho	Tamanho	Descrição
0	var	Hop-by-Hop Options (Opções de salto a salto). Um conjunto de opções que deve ser examinado em cada salto (hop)
60	var	Opções de destino. Um conjunto de opções passadas para o primeiro roteador de salto e cada roteador intermediário
43	var	Route Header. Um cabeçalho que permite vários tipos de informações de roteamento a serem anexadas
44	8	Fragment Header (Cabeçalho de fragmento). Presente em um fragmento para especificar os campos usados para a remontagem
51	var	Authentication Header (Cabeçalho de Autenticação). Especifica o tipo de autenticação usado e dados para o receptor
50	var	Encapsulation Security Payload Header (Cabeçalho de encapsulamento de segurança de dados). Especifica a criptografia usada
60	var	Opções de destino. Um conjunto de opções passadas para o destino final
135	var	Mobility Header (Cabeçalho de mobilidade). Usado para especificar as informações de encaminhamento para um host móvel

Figura 7.14 Exemplo de opções de cabeçalhos usados com IPv6 e valor do PRÓXIMO CABEÇALHO atribuído a cada um.

Algumas opções IPv6 usam um cabeçalho de extensão de tamanho fixo. Por exemplo, um cabeçalho de fragmento contém exatamente oito octetos.** No entanto, muitos dos cabeçalhos de extensão IPv6, tais como os exemplos listados na Figura 7.14, são de tamanho variável; o tamanho depende do conteúdo. Por exemplo, o *Cabeçalho de Autenticação* especifica a forma de autenticação que está sendo usada e contém uma mensagem de autenticação, na forma especificada.

O formato dos cabeçalhos de extensão de comprimento variável não é fixo. Um cabeçalho pode conter um único item (por exemplo, o *Cabeçalho de Autenticação* contém uma mensagem de autenticação) ou um conjunto de itens. Como exemplo, considere o cabeçalho de extensão salto a salto (*Hop-By-Hop*). A norma especifica um formato geral que permite várias opções a serem anexadas. A Figura 7.15 ilustra o formato.

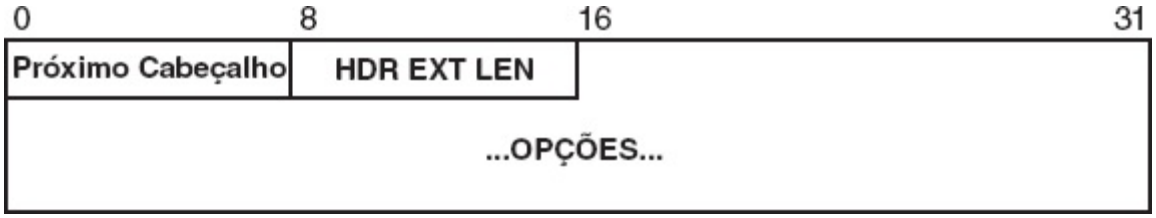


Figura 7.15 O cabeçalho de extensão IPv6 Hop-by-Hop (salto a salto) que inclui várias opções.

Como a figura indica, apenas os dois primeiros octetos são especificados: um campo de PRÓXIMO CABEÇALHO e um campo de *Comprimento de Cabeçalho de Extensão* (*HDR EXT LEN*). O comprimento do campo especifica o comprimento da extensão de cabeçalho, em octetos. O corpo do cabeçalho de extensão segue uma abordagem *Type-Length-Value* (*TLV*). O corpo consiste de opções que começam cada uma com um cabeçalho de 2 octetos. O primeiro octeto especifica o tipo de opção, o segundo especifica o comprimento e os próximos octetos contêm o valor. Como no IPv4, as opções no cabeçalho de extensão são contíguas.

O IPv6 exige cabeçalhos de datagramas em consonância a um múltiplo de oito octetos. Opções de tamanho variável significa que o *cabeçalho*

Hop-By-Hop não pode se alinhar corretamente. Em tais casos, o IPv6 define duas opções de preenchimento que um remetente pode utilizar para alinhar os cabeçalhos. Uma consiste em um único octeto de enchimento; outra usa dois octetos para especificar um comprimento de enchimento.

Inicialmente, o IPv6 incluiu muitas opções iguais às do IPv4. Por exemplo, um dos cabeçalhos de extensão do IPv6 é designado para ser um *Cabeçalho-Rota*, e a definição inicial prevê variações strict source route e loose source route. O formato geral do cabeçalho-rota também foi feito a partir do IPv4 – uma lista de endereços, um campo que especifica o comprimento da lista de octetos e um campo que apontava para o endereço seguinte. No entanto, uma avaliação de segurança concluiu que, dando aos usuários a habilidade de especificar uma rota de origem por meio de uma lista arbitrária de endereços permitiria que um invasor enviasse um datagrama em torno de um conjunto de roteadores, muitas vezes, consumindo largura de banda. Portanto, as opções de fonte de rota são agora obsoletas (ou seja, o IETF desencoraja seu uso). Elas foram substituídas por uma rota-fonte que inclui um site intermediário, porque um único site intermediário é necessário para Mobile IPv6.*

7.15 Opções de processamento durante fragmentação

O IPv4 e o IPv6 usam a mesma abordagem conceitual para lidar com opções durante a fragmentação. Ao criar fragmentos, o código IP examina cada uma das opções no datagrama original. Se uma opção deve ser processada por roteadores intermediários, a opção é copiada para cada fragmento. No entanto, se a opção é usada somente no destino final, a opção é copiada no cabeçalho do primeiro fragmento, mas não no resto. Omitir opções desnecessárias para os demais fragmentos reduz o número total de bits transmitidos. Curiosamente, omitir opções também pode reduzir o número de fragmentos necessários (isto é, um cabeçalho menor significa que um fragmento pode conter mais dados na da carga útil).

Apesar de usar o mesmo conceito, IPv4 e IPv6 são diferentes na maioria dos detalhes. As próximas seções descrevem como cada um lida com opções.

7.15.1 Processamentos opcionais IPv4 durante a fragmentação

Lembre-se de que, no IPv4, cada opção começa com um octeto de código. Cada octeto de código contém um *bit de cópia* que especifica se a opção

deve ser replicada em todos os fragmentos ou em apenas um fragmento. Como exemplo, considere a opção de registro de rota. Como cada fragmento é tratado como um datagrama independente, não há nenhuma garantia de que todos os fragmentos seguirão o mesmo caminho para o destino. Pode ser interessante conhecer um conjunto de caminhos que cada um dos fragmentos tomou, mas os desenvolvedores decidiram que um destino não terá nenhuma maneira de arbitrar entre os vários caminhos. Portanto, o padrão IP especifica que a opção de registro de rota só deve ser copiada em um dos fragmentos.

Fontes de opções de rota fornecem um exemplo de opções que devem ser copiadas para cada fragmento. Quando um remetente especifica uma rota-fonte, tem a intenção de que o datagrama siga o caminho especificado através de uma internet. Se o datagrama é fragmentado em algum ponto ao longo do caminho, todos os fragmentos devem seguir o resto do caminho que o remetente especificou, o que significa que a informação de encaminhamento de origem deve ser replicada em todos os cabeçalhos do fragmento. Portanto, a norma especifica que uma fonte de opção de rota deve ser copiada para todos os fragmentos.

7.15.2 Processamentos opcionais IPv6 durante a fragmentação

O IPv6 divide um datagrama em duas partes conceituais: um pedaço inicial que é classificado como *infragmentável* e o restante, que é classificado como *fragmentável*. O cabeçalho-base encontra-se na peça infragmentável e a carga encontra-se na peça fragmentável. Portanto, a única pergunta é sobre os cabeçalhos de extensão: como cada um deve ser classificado? Tal como acontece com o IPv4, cabeçalhos de extensão que só são processadas pelo destino final não precisam estar presentes em cada fragmento. Os padrões IPv6 especificam se um cabeçalho é fragmentável. Em particular, o *Cabeçalho Salto a Salto* (*hop-by-hop*) e o *Cabeçalho Rota* não são fragmentáveis; outros cabeçalhos de extensão são fragmentáveis. Portanto, a parte fragmentável do datagrama começa após os cabeçalhos de extensão infragmentáveis. Para esclarecer a ideia, considere o exemplo na Figura 7.16, que ilustra a fragmentação de um datagrama IPv6 que tem um cabeçalho-base, quatro cabeçalhos de extensão e 1400 octetos de carga útil.

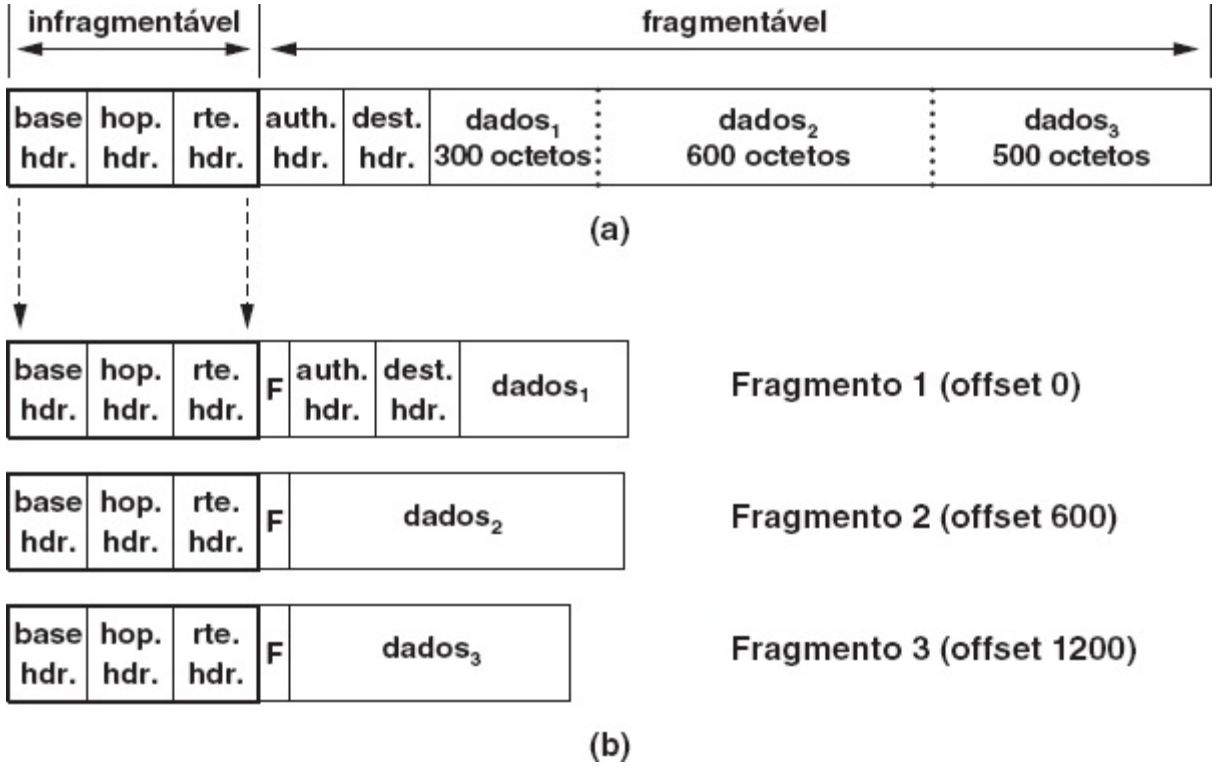


Figura 7.16 Fragmentação IPv6 com (a) um datagrama IPv6 com cabeçalhos de extensão, divididos em pedaços fragmentáveis e infragmentáveis, e (b) um conjunto de fragmentos.

Como a figura indica, cada fragmento começa com uma cópia do pedaço infragmentável do datagrama original. Na figura, o pedaço infragmentável inclui um *Cabeçalho-base*, um *Cabeçalho Salto a Salto* e um *Cabeçalho Rota*. Após o pedaço infragmentável, há um cabeçalho de *Cabeçalho de Fragmento* denominado *F* na Figura.

O IPv6 trata a parte fragmentável do datagrama como uma matriz de octetos para ser dividida em fragmentos. No exemplo, o primeiro fragmento carrega um *Cabeçalho de Autenticação*, um *Cabeçalho de Destino* e 300 octetos de dados da carga útil inicial. O segundo fragmento carrega os próximos 600 octetos da carga útil a partir do datagrama original, e o terceiro fragmento carrega o restante da carga útil. Podemos concluir a partir da figura que, neste caso particular, o *Cabeçalho de Autenticação* e o *Cabeçalho de Destino* ocupam exatamente 300 octetos.

7.16 Ordem de byte da rede

Nossa discussão de campos de cabeçalho omite uma ideia fundamental: protocolos devem especificar detalhes suficientes para garantir que ambos os lados interpretem os dados da mesma forma. Em particular, para manter interconexão de rede independente da arquitetura de máquina de qualquer fornecedor particular ou hardware de rede, temos de especificar um padrão para a representação de dados. Considere o que acontece, por exemplo, quando o software em um computador envia um inteiro binário de 32 bits para outro computador. Podemos assumir que o hardware de rede subjacente vai mover a sequência de bits da primeira para a segunda máquina, sem alterar a ordem. No entanto, nem todos os computadores armazenam inteiros de 32 bits da mesma forma. Em alguns (chamados *little endian*), o menor endereço de memória contém o byte de menor ordem do inteiro. Em outros (chamados *big endian*), o menor endereço de memória contém o byte de alta ordem do inteiro. Outros ainda armazenam números inteiros em grupos de palavras de 16 bits, com os menores endereços pegando a palavra de menor ordem, mas com bytes trocados. Assim, a cópia direta de bytes a partir de uma máquina para outra pode mudar o valor do inteiro.

A padronização de ordem de byte para inteiros é especialmente importante para cabeçalhos de protocolo porque um cabeçalho normalmente contém valores binários que especificam informações como o tamanho do pacote ou um campo de tipo que especifica o tipo de dados na área de carga útil. Tais quantidades devem ser entendidas tanto pelo emissor como pelo receptor.

Os protocolos TCP/IP resolvem o problema de ordem de byte, definindo uma *ordem de byte padrão de rede* que todas as máquinas devem usar para campos binários em cabeçalhos. Cada host ou roteador converte itens binários da representação local para a ordem de byte padrão de rede antes de enviar um pacote, e converte de ordem de byte de rede para ordem específica do host quando um pacote chega. Naturalmente, o campo carga útil em um pacote é isento do padrão de ordem de byte, pois os protocolos TCP/IP não sabem o que os dados estão transportando – os programadores de aplicativos são livres para formatar sua própria representação de dados e tradução. Ao enviar valores inteiros, muitos programadores de aplicativos optam por seguir os padrões de ordem de byte TCP/IP, mas a escolha é muitas vezes feita por mera conveniência (ou seja, usa um padrão existente, em vez de escolher um só para uma aplicação). Em qualquer caso, a questão da ordem de bytes é relevante apenas para programadores de aplicativos; usuários raramente lidam diretamente com problemas de ordem de bytes.

O padrão Internet para a ordem de byte especifica que os inteiros são enviados com o byte mais significativo primeiro (ou seja, estilo *big endian*). Se considerarmos os bytes sucessivos de um pacote que se desloca a partir de uma máquina para outra, um número inteiro binário nesse pacote tem o seu bit mais significativo próximo do início do pacote e o byte menos significativo mais próximo do fim do pacote.

> *Os protocolos de Internet definem* ordem de bytes de rede *para serem big endian. Um remetente deve converter todos os campos inteiros nos cabeçalhos dos pacotes para ordem de byte de rede antes de enviar um pacote, e um receptor deve converter todos os campos inteiros em cabeçalhos de pacote para ordem de byte local antes de processar um pacote.*

Tem havido muita discussão sobre qual representação dos dados deve ser usada, e o padrão Internet ainda fica sob ataque de vez em quando. Em particular, os defensores da mudança argumentam que, embora a maioria dos computadores eram big endian quando a norma foi definida, a maioria agora é little endian. No entanto, todos concordam que ter um padrão é fundamental, e que a forma exata do padrão é muito menos importante.

7.17 Resumo

O serviço fundamental fornecido pelo software de Internet TCP/IP consiste em um sistema sem conexão de entrega não confiável de pacotes por melhor esforço. O Protocolo de Internet (IP) especifica formalmente o formato dos pacotes de internet, chamado de *datagramas*, e informalmente encarna as ideias de entrega sem conexão. Este capítulo se concentrou em formatos de datagramas; capítulos posteriores discutirão o encaminhamento IP e tratamento de erros.

Analogamente a um frame físico, o datagrama IP é dividido em áreas de cabeçalho e de dados. Entre outras informações, o cabeçalho do datagrama contém os endereços de origem e destino Internet e o tipo do item que segue o cabeçalho. A versão 6 do Protocolo Internet altera o formato de um único cabeçalho com vários campos para um cabeçalho-base mais uma série de cabeçalhos de extensão.

Um datagrama grande pode ser dividido em fragmentos para transmissão através de uma rede que tem um MTU pequeno. Cada fragmento viaja como um datagrama independente; o destino final reconstitui os fragmentos. No IPv4, um roteador executa a fragmentação quando um datagrama deve ser enviado através de uma rede e não se encaixa no frame da rede. No IPv6, a fonte original realiza toda a fragmentação; um host deve investigar para encontrar o caminho MTU. As opções que devem ser processadas pelos roteadores intermediários são copiadas em cada fragmento; opções que são manipuladas pelo destino final são enviadas no primeiro fragmento.

EXERCÍCIOS

7.1 Qual é a única maior vantagem de ter uma cobertura checksum apenas do cabeçalho do datagrama e não da carga útil? Qual é a desvantagem?

7.2 É necessário o uso de um checksum IP ao enviar pacotes através de uma rede Ethernet? Sim ou não? Por quê?

7.3 Qual é a MTU de uma rede 802.11? Fibre Channel? 802.15.4?

7.4 Você espera que uma rede local de alta velocidade tenha MTU maior ou menor do que uma rede de longa distância? Por quê?

7.5 Justifique a afirmação de que um fragmento *não* deve se parecer com um datagrama.

7.6 A Ethernet atribui um novo valor tipo para IPv6, o que significa que o tipo do frame pode ser usado para distinguir entre datagramas de chegada IPv6 e IPv4. Por que é necessário ter um número de versão nos quatro primeiros bits de cada datagrama?

7.7 No exercício anterior, estimar o total de bits que são transmitidos em todo o mundo a cada ano só para levar o número da versão de 4 bits.

7.8 Qual é a vantagem de usar um checksum complementar para IP em vez de um Cyclic Redundancy Check?

7.9 Suponha que o projeto Internet foi alterado para permitir aos roteadores ao longo de um caminho remontarem datagramas. Como a mudança afeta a segurança?

7.10 Qual é a MTU mínima de rede necessária para enviar um datagrama IPv4 que contém, pelo menos, um octeto de dados? E um datagrama IPv6?

7.11 Suponha que você seja contratado para implementar o processamento de datagramas IP em hardware. Existe algum rearranjo de campos no

cabeçalho que faria o seu hardware mais eficiente, mais fácil de construir?

7.12 Quando um datagrama IP de tamanho mínimo viaja através de uma Ethernet, qual o tamanho do frame? Explique.

7.13 A interpretação dos serviços diferenciados do campo TIPO DE SERVIÇO permite até 64 níveis de serviço diferentes. Justifique por que são necessários menos níveis (ou seja, fazer uma lista de todos os serviços possíveis que um usuário pode acessar).

---

* Profissionais da rede às vezes chamam de "pacotes de Internet" um datagrama enquanto viaja através de uma rede; a distinção ficará clara quando falarmos de encapsulamento.

* Uma exceção é feita quando o datagrama inclui as opções de rota de origem listadas a seguir.

* O limite de 1500 octetos tornou-se importante devido ao fato de muitas redes na Internet global usarem tecnologia Ethernet.

* Offsets são especificados em múltiplos de 8 octetos para economizar espaço no cabeçalho.

* A maior parte das redes profissionais usam o termo hop count (*contador de salto*) em vez de hop limit (*limite de salto*).

* A hora universal era antes chamada Hora Média de Greenwich; é a hora do dia, no meridiano de referência.

** A Figura 7.12 na página 89 ilustra campos no cabeçalho de fragmento.

* O Capítulo 18 descreve o IP móvel.

CAPÍTULO 8

Protocolo de Internet: encaminhando datagramas IP

CONTEÚDOS DO CAPÍTULO

8.1 Introdução

Vimos que todos os serviços de internet utilizam um sistema de entrega de pacotes subjacente, sem conexão, e que a unidade básica de transferência em uma rede TCP/IP é o datagrama IP. Este capítulo acrescenta a descrição do serviço sem conexão, mostrando como os roteadores encaminham datagramas IP e os remetem aos seus destinos finais. Pensamos no formato de datagrama do Capítulo 7 como caracterizando os aspectos estáticos do Internet Protocol. A descrição de encaminhamento neste capítulo caracteriza os aspectos operacionais. O próximo capítulo completa nossa apresentação básica do IP, descrevendo como os erros são tratados. Capítulos posteriores mostram como outros protocolos utilizam o IP para oferecer serviços de nível superior.

8.2 Encaminhamento em uma Internet

Tradicionalmente, o termo *roteamento* (*routing*) era usado com sistemas de comutação de pacotes, como a Internet, para se referir ao processo de escolher um caminho pelo qual os pacotes são enviados, e o termo *roteador* para descrever o dispositivo de comutação de pacote que faz essa escolha. Aproximadamente vinte anos depois do início da Internet, profissionais de rede adotaram o termo *encaminhamento* (*forwarding*)

para se referir ao processo de escolha do caminho para um pacote, mas, curiosamente, mantiveram o termo *roteador* para se referir ao sistema que faz o encaminhamento. Seguiremos o uso popular, e usaremos o termo *encaminhamento*.

Encaminhamento ocorre em vários níveis. Por exemplo, dentro de uma Ethernet comutada que abrange múltiplos chassis físicos, os comutadores são responsáveis pelo encaminhamento de frames Ethernet entre os computadores. O frame entra no comutador através de uma porta que se conecta ao computador de envio, e o comutador o transmite para fora da porta que leva ao host de destino. Esta transmissão interna é completamente autossuficiente dentro de uma única rede Ethernet. Máquinas do lado de fora não participam do encaminhamento Ethernet; elas apenas veem a rede como uma entidade que aceita e entrega pacotes.

Lembre-se de que o objetivo do IP é fornecer uma rede virtual, que abrange várias redes físicas e oferece um serviço de entrega de datagrama sem conexão que é uma versão abstrata do serviço fornecido por um comutador Ethernet. Ou seja, queremos que a Internet aceite um pacote Internet e o entregue ao destinatário desejado (isto é, opere como se a Internet funcionasse como um comutador Ethernet gigante). As diferenças principais são que, em vez de frames, a Internet aceita e entrega datagramas IP, e, em lugar de endereços Ethernet, a Internet usa endereços IP. Portanto, ao longo do capítulo, vamos restringir a discussão no *encaminhamento IP*.

O uso do software IP de informação para tomar decisões de encaminhamento é conhecido como *Base de Informação de Encaminhamento*, *Forwarding Information Base* (*FIB*). Cada módulo IP tem seu próprio FIB, e cada um tem de tomar decisões de encaminhamento. A ideia básica é simples: dado um datagrama, o IP escolhe como enviá-lo ao destino. Ao contrário de encaminhamento dentro de uma única rede, no entanto, o algoritmo de encaminhamento IP não escolhe simplesmente entre um conjunto de computadores de destino. Em vez disso, o IP deve ser configurado para enviar um datagrama através de múltiplas redes físicas.

O encaminhamento em uma internet pode ser difícil, especialmente entre computadores que possuem várias conexões de rede física. Você deve imaginar que software de encaminhamento escolherá um caminho de acordo com a capacidade de carga por toda a rede, o tamanho de datagrama, o tipo de dado que está sendo transportado, o tipo de serviço requerido no cabeçalho do datagrama, e (talvez) o custo econômico dos vários caminhos. Veremos que a maior parte dos softwares de encaminhamento internet é

muito menos sofisticada, e seleciona rotas baseado em premissas fixas sobre os menores caminhos.

Para entender completamente o encaminhamento IP, devemos pensar na arquitetura de uma internet TCP/IP. Primeiro, lembre-se de que uma internet é composta de várias redes físicas interconectadas por roteadores. Cada roteador tem conexões diretas com duas ou mais redes. Por outro lado, um computador host geralmente se conecta diretamente a uma rede física. Sabemos que é possível ter um multi-homed host diretamente ligado a várias redes, mas vamos adiar a reflexão sobre multi-homed hosts por enquanto.

Ambos, hosts e roteadores, participam no encaminhamento de um datagrama IP para o seu destino. Quando um programa de aplicação em um host se comunica com um aplicativo remoto, o software de protocolo no host começa a gerar datagramas IP. Quando ele recebe um datagrama de saída, o software IP no host toma uma decisão de encaminhamento: ele escolhe para onde enviar o datagrama. Mesmo que um host se conecte a uma única rede, pode precisar tomar decisões de roteamento. A Figura 8.1 mostra uma arquitetura de exemplo em que um host com uma conexão de rede deve tomar decisões de encaminhamento.

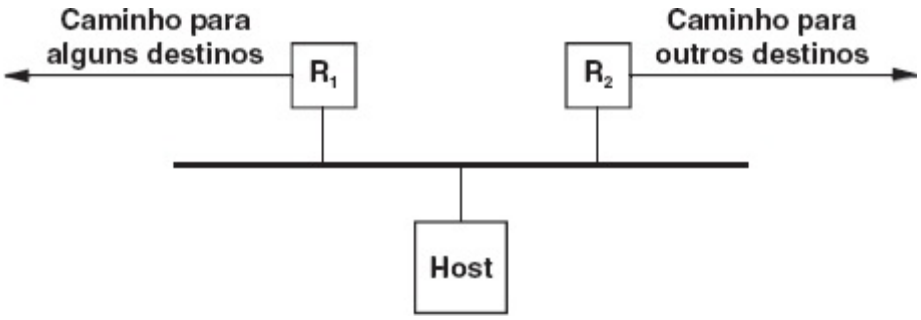


Figura 8.1 Um exemplo de um host singly-homed que precisa escolher entre o roteador R_1 e o roteador R_2 para enviar um datagrama.

Na figura, dois roteadores se conectam à mesma rede que o host. Alguns destinos de internet estão além do roteador R_1 e outros, além do roteador R_2. O host deve decidir qual roteador usar para um determinado datagrama.

Um roteador realiza *encaminhamento de trânsito* (*transit forwarding*), o que significa que irá aceitar datagramas de entrada a partir de qualquer uma das redes a que estiver conectado, e encaminhará cada datagrama ao seu destino. Surge uma questão para os hosts multi-homed. Suponha, por exemplo, que um computador tenha conexões de rede celular

Wi-Fi e 4G. O computador pode agir como um roteador e fornecer encaminhamento de trânsito entre as redes? Veremos que qualquer computador com TCP/IP tem todo o software necessário para encaminhar datagramas. Assim, em teoria, qualquer computador com várias conexões de rede pode funcionar como um roteador. No entanto, os padrões TCP/IP estabelecem uma distinção nítida entre as funções de um host e as de um roteador. Qualquer um que tente misturar as funções de host e de roteador em uma única máquina, configurando um host para fornecer o encaminhamento de trânsito, descobre que a máquina pode não funcionar como esperado. Por enquanto, vamos distinguir hosts de roteadores e assumir que os hosts não executam a função de roteador na transferência de pacotes de uma rede para outra.

Apesar de um host com múltiplas conexões de rede poder ser configurado para funcionar como um roteador, o sistema resultante pode não funcionar como esperado.

8.3 Entrega direta e indireta

Grosseiramente falando, podemos dividir encaminhamento de duas formas: *entrega direta* e *indireta*. Na entrega direta, a transmissão de um datagrama diretamente de uma máquina para outra através de uma única rede física é a base sobre a qual toda a comunicação internet repousa. Duas máquinas podem se envolver em entrega direta somente se ambas se conectam diretamente no mesmo sistema físico subjacente de transmissão (por exemplo, uma única Ethernet). Entrega indireta ocorre quando o destino de um datagrama não está em uma rede diretamente conectada. Como o destino final não pode ser acessado diretamente, o remetente deve escolher um roteador, transferir o datagrama através de uma rede conectada diretamente ao roteador e permitir que ele encaminhe o datagrama para o destino final.

8.4 Transmissão através de rede única

Sabemos que uma máquina em uma determinada rede física pode enviar um frame diretamente para outra máquina na mesma rede. Vimos também como o software IP usa o hardware. Para transferir um datagrama IP, o remetente o encapsula em um frame físico, como descrito no Capítulo 7, mapeia o endereço IP do próximo salto para um endereço de hardware,

coloca o endereço de hardware no frame e usa o hardware de rede para transferir o frame. Como o Capítulo 6 descreve, o IPv4 geralmente usa ARP para mapear um endereço IP em um endereço de hardware, e o IPv6 usa Neighbor Discovery para aprender os endereços de hardware de nós dos vizinhos. Portanto, os capítulos anteriores examinaram todas as peças necessárias para entender a entrega direta. Podemos fazer o resumo a seguir.

A transmissão de um datagrama IP entre duas máquinas em uma única rede física não envolve roteadores. O remetente encapsula o datagrama em um frame físico, vincula o endereço do próximo salto para um endereço de hardware físico e envia o frame resultante diretamente para o destino.

A ideia de enviar datagramas diretamente através de uma única rede pode parecer óbvia, mas originalmente não era. Antes de o TCP/IP ser inventado, várias tecnologias de rede requeriam o equivalente a um roteador anexado a cada rede. Quando dois computadores da rede precisavam se comunicar, o faziam através do roteador local. Seus defensores argumentavam que ter toda a comunicação passando por um roteador significava que a comunicação utilizava o mesmo paradigma e permitia uma implementação fácil de segurança. O projeto TCP/IP mostrou que a comunicação direta reduziu o tráfego de rede por um fator de dois.

Suponha que o software IP em uma máquina receba um datagrama IP. Como o software pode saber se o destino se encontra em uma rede conectada diretamente? O teste é simples e ajuda a explicar o esquema de endereçamento IP. Recorde-se de que cada endereço IP é dividido em um prefixo que identifica a rede e um sufixo, que identifica um host. Para determinar se um destino se encontra em uma das redes diretamente conectadas, o software IP extrai a parte de rede do endereço IP de destino e compara o ID da rede com o ID de rede de seu(s) próprio(s) endereço(s) IP. Uma coincidência significa que o destino encontra-se em uma rede conectada diretamente e o datagrama pode ser entregue diretamente ao destino. O teste é computacionalmente eficiente, o que destaca por que o esquema de endereços Internet funciona bem como podemos verificar a seguir.

Como os endereços de internet de todas as máquinas em uma única rede incluem um prefixo de rede comum e extrair esse prefixo requer apenas algumas instruções de máquina, é eficiente testar se um destino pode ser alcançado diretamente.

Do ponto de vista de internet, é mais fácil pensar em entrega direta como a etapa final em qualquer transmissão de datagramas. Um datagrama pode atravessar muitas redes e roteadores intermediários enquanto trafega da origem para o destino. O roteador final ao longo do caminho irá se conectar diretamente à mesma rede física de destino. Assim, o roteador final vai entregar o datagrama usando entrega direta. Em essência, um caminho através de uma internet envolve zero ou mais roteadores intermediários mais uma etapa de entrega direta. O caso especial surge quando não há roteadores no caminho – o host de envio deve executar a etapa de entrega diretamente.

8.5 Entrega indireta

A entrega indireta é mais difícil do que a direta porque a máquina de envio deve identificar um roteador inicial para lidar com o datagrama. O roteador deve, então, encaminhar o datagrama para a rede de destino

Para visualizar como o encaminhamento indireto funciona, imagine uma internet grande com muitas redes interconectadas por roteadores, mas com somente dois hosts no final. Quando um host tem um datagrama para enviar, ele encapsula o datagrama em um frame e envia para o roteador mais próximo. Sabemos que o host pode alcançar um roteador porque todas as redes físicas estão interconectadas, então deve haver um roteador anexado a cada rede. Então o host de origem pode alcançar um roteador usando uma única rede física. Uma vez que o frame alcança o roteador, o software extrai o datagrama encapsulado e o software IP escolhe o próximo roteador ao longo do caminho em direção ao destino. O datagrama é novamente colocado em um frame e enviado pela próxima rede física para um segundo roteador, e assim por diante, até que ele possa ser entregue diretamente. O conceito pode ser resumido como verificamos a seguir.

Roteadores em uma internet TCP/IP formam uma estrutura cooperativa, interconectada. Datagramas passam de roteador para

roteador até chegarem a um roteador que possa entregar o datagrama diretamente.

O design de internet concentra conhecimento de encaminhamento em roteadores e garante que um roteador possa encaminhar um datagrama arbitrário. Hosts dependem de roteadores para toda entrega indireta. Podemos então fazer o resumo a seguir.

- Um host só sabe sobre redes conectadas diretamente; um host baseia-se em roteadores para transferir datagramas para destinos remotos.
- Cada roteador sabe como chegar a todos os destinos possíveis na internet; dado um datagrama, um roteador pode encaminhá-lo corretamente.
- Um roteador só conhece redes conectadas diretamente; um host baseia-se em roteadores para transferir datagra-mas para destinos remotos.

Como um roteador pode saber como chegar a um destino remoto? Como um host pode saber qual roteador usar para um determinado destino? As duas perguntas estão relacionadas, pois ambas envolvem encaminhamento IP. Vamos responder às questões em dois estágios: considerando um algoritmo básico de encaminhamento controlado por tabela, neste capítulo, e postergando uma discussão de como roteadores aprendem sobre destinos remotos até os Capítulos 12 a14.

8.6 Encaminhamento IP controlado por tabela

O IP encaminha datagramas. O algoritmo de encaminhamento IP emprega uma estrutura de dados que armazena informações sobre possíveis destinos e sobre como alcançá-los. A estrutura de dados é conhecida formalmente como uma *tabela de roteamento do Internet Protocol*, ou *tabela de roteamento IP*, e informal e simplesmente como uma *tabela de roteamento*.*

Como cada um deles deve encaminhar datagramas, tanto hosts quanto roteadores possuem tabelas de roteamento. Veremos que a tabela de roteamento em um host típico é muito menor que a de um roteador, mas a vantagem de usar uma tabela é que um único mecanismo de roteamento atende a ambos os casos. Sempre que o software de encaminhamento IP em um host ou roteador precisa transmitir um datagrama, ele consulta a tabela

de roteamento a fim de decidir para onde enviar o datagrama.

Que informações devem ser mantidas em uma tabela de roteamento? Se cada tabela de roteamento tivesse informações sobre cada endereço de destino possível em uma internet, seria impossível mantê-las atualizadas. Além do mais, como o número de destinos possíveis é grande, pequenos sistemas de uso especial não poderiam se conectar à Internet, pois não teriam espaço suficiente para armazenar as informações.

Conceitualmente, é desejável usar o princípio de ocultar informações e permitir que as máquinas tomem decisões de encaminhamento com um mínimo de informações. Por exemplo, gostaríamos de isolar informações sobre hosts específicos para o ambiente local em que eles existem e organizar as máquinas que estão distantes de modo a encaminhar pacotes para elas sem conhecer tais detalhes. Felizmente, o esquema de endereçamento IP ajuda a conseguir esse objetivo. Lembre-se de que os endereços IP são atribuídos para fazer com que todas as máquinas conectadas a determinada rede física compartilhem um prefixo comum (a parte de rede do endereço). Já vimos que essa atribuição torna eficiente o teste de entrega direta. Isso também significa que as tabelas de roteamento só precisam conter prefixos de rede, e não endereços IP completos. A distinção é fundamental: a Internet global tem mais de 800.000.000 de computadores individuais, mas somente 400.000 prefixos IPv4 únicos. Então, a necessidade de prefixos para o encaminhamento de informações é três ordens de grandeza menor do que o encaminhamento de informações para computadores individuais. O ponto importante é colocado a seguir.

> *Por permitir que o encaminhamento esteja baseado nos prefixos de rede, o esquema de endereçamento IP controla o tamanho das tabelas de roteamento.*

Quando discutirmos propagação de roteamento, veremos que o esquema de encaminhamento IP tem outra vantagem: só nos é solicitado propagar informação sobre redes, e não sobre hosts individuais. Na verdade, um host pode se conectar a uma rede (por exemplo, Wi-Fi hot spot) e começar a usar a rede sem quaisquer alterações nas tabelas de roteamento.

8.7 Encaminhamento do próximo salto

Dissemos que o uso da parte de rede de um endereço IP em vez do lugar do endereço de host completo torna o encaminhamento eficiente e mantém as

tabelas de roteamento pequenas. Mais importante, isso ajuda a ocultar informações, mantendo os detalhes dos hosts específicos confinados ao ambiente local em que os hosts operam. Conceitualmente, uma tabela de encaminhamento contém um conjunto de pares (N, R), em que N é o *prefixo de rede* para uma rede na internet e R é o endereço IP do "próximo" roteador ao longo do caminho até a rede N. O roteador R é chamado de *próximo salto* (*next hop*), e a ideia de usar uma tabela de encaminhamento para armazenar o próximo salto para *cada* destino é chamada de *encaminhamento de próximo salto*. Assim, a tabela de encaminhamento em um roteador R só especifica uma etapa ao longo do caminho de R até uma rede de destino – o roteador não conhece o caminho completo até um destino.

É importante entender que cada entrada em uma tabela de encaminhamento aponta para um roteador que pode ser alcançado por uma única rede. Ou seja, todos os roteadores listados na tabela de encaminhamento da máquina M precisam estar em redes às quais M se conecta diretamente. Quando um datagrama estiver pronto para sair de M, o software IP localiza o endereço IP de destino e extrai a parte da rede. Depois, M procura a parte de rede na tabela de encaminhamento, selecionando uma das entradas. A entrada selecionada na tabela vai especificar um roteador próximo salto que pode ser alcançado diretamente.

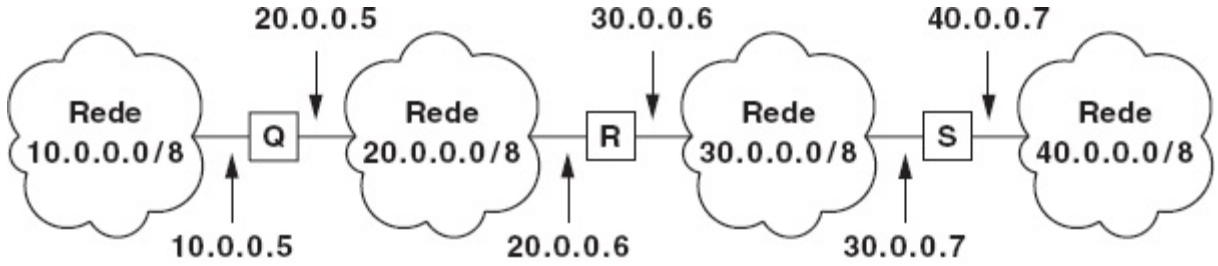
Na prática, aplicamos o princípio de ocultar informações também para os hosts. Insistimos que, embora os hosts tenham tabelas de encaminhamento IP, eles precisam manter informações mínimas em suas tabelas. A ideia é forçar os hosts a contarem com roteadores para a maior parte do encaminhamento.

A Figura 8.2 mostra um exemplo concreto que ajuda a explicar as tabelas de encaminhamento. A internet do exemplo consiste em quatro redes conectadas por três roteadores. A tabela na figura corresponde à tabela de encaminhamento para o roteador R. Embora o exemplo use endereços IPv4, o conceito se aplica igualmente para o IPv6.

Na figura, a cada rede foi atribuído um prefixo slash-8, e a cada interface de rede foi atribuído um endereço IPv4 de 32 bites. O administrador de rede que atribui endereços IP escolheu o mesmo sufixo host para ambas as interfaces de um roteador. Por exemplo, a interface no roteador Q tem

endereços 10.0.0.5 e 20.0.0.5. Apesar de o IP permitir sufixos arbitrários, escolher o mesmo valor para ambas as interfaces torna o endereçamento mais fácil para humanos lembrarem.

Como o roteador *R* se conecta diretamente às redes 20.0.0.0 e 30.0.0.0, ele pode usar a entrega direta para enviar a um host em qualquer uma dessas redes. O roteador usa o ARP (IPv4) ou mapeamento direto (IPv6) para encontrar endereços físicos de um computador nessas redes. Dado um datagrama destinado a um host na rede 40.0.0.0, *R* não pode entregar diretamente. Ao contrário, *R* encaminha o datagrama para um roteador *S* (com endereço 30.0.0.7) que vai então entregá-lo diretamente. *R* pode alcançar o endereço 30.0.0.7, pois ambos *R e* S se conectam diretamente à rede 30.0.0.0.



(a)

PARA ALCANÇAR HOSTS NA REDE	ROTA PARA O ENDEREÇO
20.0.0.0/8	ENTREGA DIRETA
30.0.0.0/8	ENTREGA DIRETA
10.0.0.0/8	20.0.0.5
40.0.0.0/8	30.0.0.7

(b)

Figura 8.2 (a) Um exemplo de internet com 4 redes e 3 roteadores, e (b) a tabela de encaminhamento em *R*.

Como demonstra a Figura 8.2, o tamanho da tabela de encaminhamento depende do número de redes na internet; a tabela só cresce quando novas redes são acrescentadas. Ou seja, o tamanho e o conteúdo da tabela são

independentes do número de hosts individuais conectados às redes. Podemos resumir o princípio básico a seguir.

> *Para ocultar informação, manter as tabelas de encaminhamento pequenas e tornar as decisões de encaminhamento eficientes, o software de encaminhamento IP só mantém informação sobre endereços de rede de destino, e não sobre endereços de host individuais.*

A escolha de rotas com base apenas no prefixo da rede de destino apresenta diversas consequências. Primeiro, na maior parte das implementações, isso significa que todo o tráfego destinado a determinada rede segue o mesmo caminho. Como resultado, mesmo quando existem vários caminhos, eles podem não ser usados simultaneamente. Além disso, no caso mais simples, todo o tráfego segue o mesmo caminho, sem considerar o atraso ou a vazão das redes físicas. Segundo, como apenas o roteador final do caminho tenta se comunicar com o host de destino, somente ele pode determinar se o host existe ou está operante. Assim, precisamos arrumar um meio para que o roteador final envie relatórios de problemas de entrega de volta à origem. Terceiro, como cada roteador encaminha o tráfego independentemente, os datagramas trafegando do host *A* para o host *B* podem seguir um caminho inteiramente diferente dos datagramas trafegando do host *B* de volta para o host *A*. Além disso, o caminho em uma direção pode estar desativado (por exemplo, se uma rede ou roteador falharem) mesmo se o caminho na direção oposta permanecer disponível. Precisamos nos assegurar de que os roteadores cooperem para garantir que a comunicação bidirecional seja sempre possível.

8.8 Rotas default e um exemplo host

O projeto IP inclui uma otimização interessante que esconde mais informações e reduz o tamanho das tabelas de encaminhamento: a consolidação de várias entradas em um único caso *default*. Conceitualmente, um caso-padrão apresenta um algoritmo de duas etapas. Na primeira etapa, o software de encaminhamento IP olha na tabela de encaminhamento para encontrar um próximo salto. Se nenhuma entrada na tabela corresponde ao endereço de destino, o software de encaminhamento

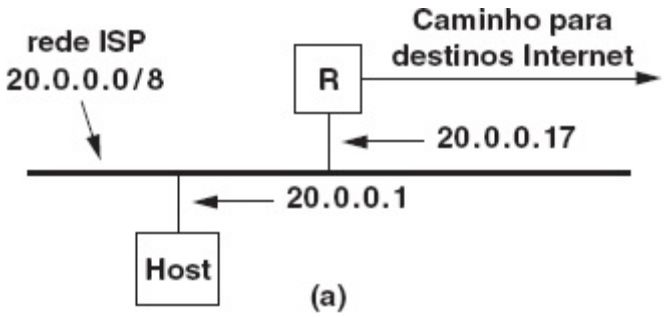
faz uma segunda etapa de verificação para uma rota *default*. Dizemos que o próximo salto especificado em uma rota-padrão é um *roteador default*.

Na prática, veremos que o roteamento default não requer duas etapas separadas. Em vez disso, uma rota default pode ser incorporada a uma tabela de encaminhamento. Isto é, uma entrada extra pode ser adicionada a uma tabela de encaminhamento que especifica um roteador default como próximo salto. O algoritmo de pesquisa pode ser organizado para coincidir com outras entradas da tabela em primeiro lugar e só examinar a entrada default se nenhuma das outras entradas corresponder. Uma seção posterior explica o algoritmo de encaminhamento que acomoda rotas default.

Uma rota default é especialmente útil quando muitos destinos estão além de um único roteador. Por exemplo, considere uma empresa que usa um roteador para conectar duas pequenas redes de departamento com a sua intranet. O roteador tem conexão com cada rede de departamento e uma conexão com o resto da intranet da empresa. O encaminhamento é simples, pois o roteador só precisa de três entradas na sua tabela de encaminhamento: uma para cada uma das duas redes departamentais e uma rota default para todos os outros destinos.

Rotas default funcionam especialmente bem para computadores hosts típicos que obtêm serviço a partir de um ISP. Por exemplo, quando um usuário adquire serviço de uma linha DSL ou modem com cabo, o hardware conecta o computador a uma rede no ISP. O host usa um roteador na rede ISP para alcançar um destino arbitrário na Internet global. Nesses casos, a tabela de encaminhamento na tabela do host precisa somente de duas entradas: uma para a rede local no ISP e uma entrada default que aponta para o roteador ISP. A Figura 8.3 ilustra a ideia.

Embora o exemplo na figura use endereços IPv4, o mesmo princípio é válido para IPv6: um host só precisa conhecer a rede local e ter uma rota default que é usada para alcançar o resto da Internet. Claro, um endereço IPv6 é quatro vezes maior que um endereço IPv4, o que significa que cada entrada em uma tabela de encaminhamento é quatro vezes maior. A consequência para os tempos de busca é mais significativa: em computadores modernos, um endereço IPv4 cabe em um único inteiro, o que significa que um computador pode usar uma comparação de um único inteiro para comparar dois endereços IPv4. Quando se compara dois endereços IP6, são necessárias várias comparações.



PARA ALCANÇAR HOSTS NESTA REDE	ENCAMINHAR PARA ESTE ENDEREÇO
20.0.0.0/8	ENTREGA DIRETA
DEFAULT	20.0.0.17

(b)

Figura 8.3 (a) Um exemplo de conexão Internet usando endereço IPv4 e (b) a tabela de encaminhamento usada no host.

8.9 Rotas específicas do host

Embora tenhamos dito que todo o encaminhamento é baseado em redes, e não em hosts individuais, a maioria dos softwares de encaminhamento IP permite uma *host-specific* route (*rota específica do host*) especificada como um caso especial. Ter rotas específicas de host dá mais controle ao administrador da rede. A habilidade para especificar uma rota para máquinas individuais muda para que haja várias possibilidade de uso como os descritos a seguir.

- *Controle sobre o uso da rede.* Um administrador pode enviar tráfego por certos hosts por outro caminho. Por exemplo, um administrador pode separar o tráfego destinado ao servidor de rede da empresa de outros tráfegos.
- *Testar uma nova rede.* Uma nova rede, paralela, pode ser instalada e testada enviando-se o tráfego por hosts específicos através dela enquanto se deixa qualquer outro tráfego na rede antiga.
- *Segurança.* Um administrador pode usar rota específica de host para direcionar tráfego de sistemas de segurança. Por exemplo, tráfego destinado ao departamento financeiro da empresa pode precisar atravessar uma rede segura que tem filtros especiais instalados.

Embora conceitos como rotas default e rotas específicas de host parecem ser casos especiais que requerem tratamento especial, a próxima seção explica como toda a informação encaminhada pode ser combinada em uma única tabela e tratada por um único e uniforme algoritmo de busca.

8.10 O algoritmo de encaminhamento IP

Levando em consideração todos os casos especiais descritos anteriormente, pode parecer que o software IP deve considerar as etapas a seguir para decidir como encaminhar um datagrama.

1. Extrair o endereço IP de destino, *D*, do datagrama.
2. Se a tabela de encaminhamento contiver uma entrada host específico para destino *D*, encaminhar o datagrama para o próximo salto especificado na entrada.
3. Se o prefixo da rede coincidir com o prefixo de alguma rede conectada diretamente, enviar o datagrama diretamente pela rede para *D*.
4. Se a tabela de encaminhamento contiver uma entrada que coincida com o prefixo da rede de *D*, encaminhar o datagrama para o próximo salto especificado na entrada.
5. Se a tabela de encaminhamento contiver uma rota default, enviar o datagrama para o próximo pulo especificado na rota default.
6. Se nenhum dos casos acima tiver encaminhado o datagrama, declare um erro de encaminhamento.

Pensar nas seis etapas individualmente nos ajuda a entender todos os casos a serem considerados. Em termos de implementação, entretanto, programar seis etapas separadas torna o código grosseiro e cheio de casos especiais (ou seja, verificar se uma rota default foi especificada). Inicialmente na história da Internet, desenvolvedores encontraram uma forma de unificar todos os casos em um único mecanismo de busca que é atualmente usado na maior parte dos softwares de IP comerciais. Vamos explicar o algoritmo conceitual, verificar uma implementação simples usando uma tabela e, então, considerar uma versão que pode ser dimensionada para lidar com roteadores de encaminhamento próximo a um centro de Internet que tem grandes tabelas de encaminhamento.

O esquema unificado de busca requer que quatro itens sejam especificados para cada entrada na tabela de encaminhamento como

descrito a seguir.

- O endereço IP, *A*, que fornece o destino para a entrada.
- Uma máscara de endereço, *M*, que especifica quantos bits de *A* examinar.
- O endereço IP de um roteador próximo salto, *R*, ou "entrega direta".
- Uma interface de rede, *I*, para usar durante o envio.

Os quatro itens definem uma rota sem ambiguidade. Deve estar claro que cada entrada em uma tabela de encaminhamento precisa do terceiro item, um endereço de roteador próximo salto. O quarto item é necessário, pois um roteador que se conecta a múltiplas redes tem múltiplas interfaces de rede internas. Quando se envia um datagrama, o IP deve especificar que interface interna usar quando enviá-lo. Os dois primeiros itens definem um prefixo de rede – a máscara especifica quais bits do endereço de destino devem ser usados durante a comparação e o endereço IP, *A*, fornece um valor com que comparar. Ou seja, o algoritmo computa o bit *logical and* da máscara, *M*, com o endereço de destino e então compara o resultado com *A*, o primeiro item na entrada.

Definimos o *tamanho* de uma máscara de endereço para ser o número de 1 bit na máscara. Em notação de barra, o comprimento de uma máscara é dado explicitamente (por exemplo, / 28 indica uma máscara com tamanho de 28). O tamanho de uma máscara de endereço é importante porque o algoritmo unificado de encaminhamento inclui mais do que prefixos de rede tradicionais. A máscara, que determina quantos bits examinar durante as comparações, também nos permite lidar com host específico e casos default. Por exemplo, considere um destino IPv6. Uma máscara / 64 significa que a comparação vai considerar os primeiros 64 bits do endereço (isto é, o prefixo de rede). Uma máscara / 128 significa que todos os 128 bits de endereço *A* na entrada vão ser comparados com o de destino (ou seja, a entrada especifica uma rota de host específico).

Como outro exemplo de como os quatro itens são suficientes para o encaminhamento arbitrário, considere uma rota default. Para criar uma entrada para uma rota default, a máscara, *M*, é definida como zero (todos os bits zero), e o campo de endereço, *A*, é definido como zero. Não importa qual é o endereço de destino num datagrama, usar uma máscara com tudo

zero resulta num valor de zero, o que é igual ao valor de *A* na entrada. Em outras palavras, a entrada sempre corresponde (isto é, proporciona uma rota *default*). O algoritmo 8.1 resume as etapas seguidas para transmitir um datagrama.

Em essência, o algoritmo interage através das entradas na tabela de encaminhamento até encontrar uma correspondência e assume que as entradas são organizadas em ordem de prefixo mais longo (ou seja, as entradas com a máscara mais longa ocorrem primeiro). Portanto, logo que o destino corresponde a uma entrada, o algoritmo pode enviar o datagrama para o próximo salto especificado. Há dois casos: entrega direta ou indireta. Para a entrega direta, o destino do datagrama é utilizado como o próximo hop. Para a entrega indireta, a tabela de envio contém o endereço de um roteador, *R*, para usar como o próximo salto. Uma vez que o próximo salto foi determinado, o algoritmo mapeia o endereço do próximo salto para um endereço de hardware, cria um frame, preenche o endereço de hardware no frame, e o envia levando o datagrama para o próximo salto.

O algoritmo assume que a tabela de envio contém uma rota default. Assim, mesmo se nenhuma outra entrada corresponder a um determinado destino, a entrada default irá corresponder. Claro, um gerente poderia cometer um erro e, inadvertidamente, remover a rota default. Nesses casos, o algoritmo irá percorrer a tabela inteira sem encontrar uma correspondência, e, então, chegará a um ponto em que ia declarar que um erro de encaminhamento ocorreu.

8.11 Paradigma Longest-Prefix Match

Para fazer com que o algoritmo funcione corretamente, as entradas na tabela devem ser examinadas em uma ordem que garanta que as entradas com máscaras maiores são verificadas antes das entradas com máscaras menores. Por exemplo, suponha que a tabela contenha uma rota de host específico para um host X e também uma rota específica da rede para a parte de rede de X. Ambas as entradas irão coincidir com o X, mas o encaminhamento deve escolher a correspondência mais específica (ou seja, a rota de host específico).

```
Encaminhar Datagrama IP (Datagrama, Tabela de Encaminhamento){
   Garantir que a tabela de encaminhamento esteja ordenada com os
   prefixos maiores primeiro
   Extrair o destino, D, do datagrama
   para cada entrada da tabela {
      Computar o logical and de D com máscara para obter um prefixo, P
      se o prefixo P corresponder a A, o endereço na entrada {
         /* Encontrar uma entrada coincidente - encaminhar como especificado*/
         se (próximo salto na entrada é "entrega direta") {
            Colocar Próximo Salto no endereço de destino, D
         } caso contrário {
              Colocar Próximo Salto para o endereço do roteador na entrada, R
         }
         Resolver endereço Próximo
         Salto para um endereço de hardware, H
         Encapsular o datagrama em um frame usando endereço H
         Iniciar o datagrama pela rede usando interface I
         Parar porque o datagrama foi enviado com sucesso
      }
   }
   Parar e declarar que ocorreu um erro de encaminhamento
}
```

Algoritmo 8.1 Algoritmo IP Unificado de Encaminhamento no qual cada entrada de tabela contém um endereço, A, uma máscara, M, um roteador próximo salto, R (ou "entrega direta"), e uma interface de rede, I.

Usamos o termo *longest-prefix match (correspondência ao prefixo mais longo)* para descrever a ideia de examinar as rotas mais específicas primeiro. Se imaginarmos a tabela de encaminhamento como sendo uma matriz, a regra da correspondência do prefixo mais longo significa que as entradas na matriz devem ser classificadas em ordem decrescente, de acordo com o tamanho da sua máscara.

8.12 Encaminhamento de tabelas e endereços IP

É importante entender que, exceto por decrementar o limite de salto (TTL no IPv4) e recalcular o checksum, o encaminhamento IP não altera o datagrama original. Em particular, os endereços de origem e destino do datagrama permanecem inalterados; eles especificam o endereço IP da origem e o endereço IP do destino final.* Quando executa o algoritmo de encaminhamento, o IP seleciona um novo endereço, o endereço IP da máquina à qual o datagrama deve ser enviado em seguida. O novo endereço provavelmente é o endereço de um roteador. Porém, se o datagrama puder

ser entregue diretamente, o novo endereço é o mesmo que o do destino final.

No algoritmo, o endereço IP selecionado pelo algoritmo de encaminhamento IP é chamado de *endereço do próximo salto*, porque ele diz para onde o datagrama deve ser enviado em seguida. Onde o IP armazena o endereço do próximo salto? Não no datagrama; nenhum lugar é reservado para ele. Na verdade, o IP não armazena o endereço do próximo salto. Depois que executa o algoritmo de encaminhamento, o módulo IP passa o datagrama e o endereço do próximo salto para a interface de rede responsável pela rede através da qual o datagrama deve ser enviado. Em essência, o IP solicita que o datagrama seja enviado para o endereço do próximo salto especificado.

Quando recebe um datagrama e um endereço do próximo salto de IP, a interface de rede deve mapear o endereço do próximo salto para um endereço de hardware, criar um frame, colocar o endereço de hardware no campo de endereço de destino do frame, encapsular o datagrama na área de carga útil do frame e transmitir o resultado. Uma vez que tenha obtido um endereço de hardware, o software de interface de rede descarta o endereço do próximo salto.

Pode parecer estranho que as tabelas de encaminhamento armazenem o endereço IP de cada próximo salto em vez do seu endereço de hardware. Um endereço IP precisa ser traduzido para um endereço de hardware correspondente antes que o datagrama possa ser enviado. Se imaginarmos um host enviando uma sequência de datagramas ao mesmo endereço de destino, o uso de endereços IP para encaminhamento poderá parecer incrivelmente ineficiente. Cada vez que um aplicativo gera um datagrama, o IP extrai o endereço de destino e procura na tabela de encaminhamento para produzir um endereço próximo salto. O IP então passa o datagrama e o endereço do próximo salto para a interface de rede, que recalcula o vínculo para um endereço de hardware. Se a tabela de roteamento armazenasse endereços de hardware, o vínculo entre o endereço IP do próximo salto e o endereço de hardware poderia ser realizado uma vez, economizando computação desnecessária.

Por que o software IP evita o uso de endereços de hardware em uma tabela de encaminhamento? A Figura 8.4 ajuda a ilustrar as duas importantes razões. Primeiro, uma tabela de encaminhamento provê uma interface clara entre o software IP que encaminha datagramas e as ferramentas de gerenciamento e software de alto nível que manipulam as rotas. Segundo, o objetivo da interligação de redes é esconder detalhes de redes subjacentes. Usar apenas endereços IP em tabelas de

encaminhamento permite que gerentes de rede trabalhem em um nível mais alto de abstração. Um gerente pode examinar ou mudar regras de encaminhamento e depurar problemas de encaminhamento enquanto usa apenas endereços de IP. Assim, um gerente não precisa se preocupar com os endereços de hardware subjacentes ou entendê-los.

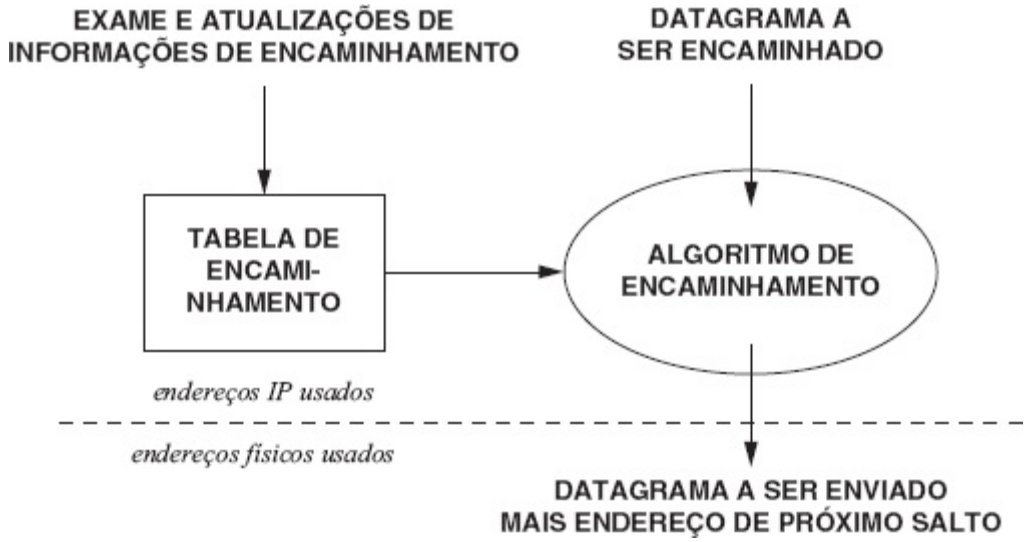


Figura 8.4 Ilustração do algoritmo de encaminhamento usando uma tabela de encaminhamento durante verificação ou alteração de entradas.

A figura ilustra uma ideia interessante: o acesso concomitante para encaminhamento de informação. O encaminhamento de datagrama, que acontece milissegundo a milissegundo, pode continuar a usar a tabela de encaminhamento enquanto um gerente examina entradas ou faz mudanças. Uma mudança toma efeito imediatamente porque o encaminhamento continua concomitantemente (a menos que o gerente desative manualmente a interface de rede).

A Figura 8.4 ilustra o *limite de endereço*, uma divisão conceitual importante entre software de baixo nível, que entende endereços de hardware, e software de internet, que usa somente endereços de alto nível. Acima do limite, todo software pode ser escrito para usar endereços internet; o conhecimento de endereços de hardware é relegado a poucas e pequenas funções de baixo nível que transmitem um datagrama. Veremos que observar o limite de endereço também ajuda a manter a implementação de protocolos de maior nível, tais como TCP, que são fáceis de entender, testar, e modificar.

8.13 Tratando de datagramas de entrada

Até aqui, discutimos o encaminhamento IP descrevendo como as decisões de encaminhamento são tomadas sobre datagramas de saída. Deve ficar claro, entretanto, que o software IP deve processar também datagramas de entrada.

Primeiro considere software host. Quando um datagrama chega a um host, o software de interface de rede o entrega para o módulo IP processar. Se o endereço de destino do datagrama coincide com o endereço IP do host (ou um dos endereços de host), o software IP no host aceita o datagrama e o passa ao software de protocolo de alto nível apropriado para continuar o processamento. Se o endereço de destino IP não coincide com um dos endereços host, o host é requerido a descartar o datagrama (ou seja, hosts são proibidos de tentar encaminhar datagramas que sejam acidentalmente encaminhados para a máquina errada).

Agora considere o software de roteador. Ao contrário dos hosts, os roteadores realizam o encaminhamento. Entretanto, um roteador também pode executar apps (por exemplo, apps de gerenciamento de rede). Quando um datagrama IP chega a um roteador, ele é entregue ao software IP, e surgem duas possibilidades: o datagrama atingiu seu destino final (ou seja, ele é direcionado a uma app no roteador), ou ele pode ter de viajar ainda mais. Assim como nos hosts, se o endereço IP de destino do datagrama combinar com o próprio endereço IP do roteador, o IP passará o datagrama para o protocolo de nível superior para processamento. Se o datagrama não tiver atingido seu destino final, o IP o encaminhará usando o algoritmo-padrão e a informação na tabela de roteamento local.

Dissemos que um host *não* deve encaminhar datagramas (ou seja, um host deve descartar datagramas entregues incorretamente). Existem quatro motivos para que um host abstenha-se de realizar qualquer encaminhamento. Primeiro, quando um host recebe um datagrama direcionado para alguma outra máquina, algo saiu errado com o endereçamento, o encaminhamento ou a entrega da internet. O problema pode não ser revelado se o host tomar ação corretiva encaminhando o datagrama. Segundo, o encaminhamento causará tráfego desnecessário na rede (e pode roubar tempo de CPU para usos legítimos do host). Terceiro, erros simples podem causar caos. Suponha que cada host encaminhe tráfego e imagine o que acontece se uma máquina acidentalmente transmitir por broadcast um datagrama destinado a um host, *H*. Por ter sido enviado por broadcast, cada host na rede receberá uma cópia do datagrama e encaminhará sua cópia para *H*, que será bombardeado com muitas cópias.

Quarto: como mostram os próximos capítulos, os roteadores realizam mais do que simplesmente encaminhar tráfego. Conforme explicaremos no capítulo seguinte, os roteadores usam um protocolo especial para relatar erros, enquanto os hosts não fazem isso (novamente, para evitar que vários relatórios de erro bombardeiem uma origem). Os roteadores também propagam informações para garantir que suas tabelas de roteamento sejam consistentes e corretas. Se os hosts encaminham datagramas sem participar plenamente de todas as funções do roteador, podem surgir anomalias inesperadas.

8.14 Encaminhamento em presença de broadcast e multicast

Determinar se um datagrama IP chegou ao seu destino final não é tão trivial como descrito anteriormente. Dissemos que, quando um datagrama chega, a máquina receptora deve comparar o endereço de destino no datagrama com o endereço IP de cada interface de rede. Claro que, se o destino corresponde, a máquina mantém o datagrama e o processa. No entanto, a máquina também deve lidar com datagramas que são transmitidos por broadcast (IPv4) ou multicast (IPv4 e IPv6) através de uma das redes a que a máquina está conectada. Se a máquina está participando no grupo multicast* ou um datagrama IPv4 foi transmitido através de uma rede local, uma cópia do datagrama deve ser passada para a pilha de protocolo local para processamento. Pode ser preciso que um roteador também envie uma cópia em uma ou mais redes. Para IPv4, broadcast direcionado apresenta várias possibilidades. Se uma transmissão broadcast direcionada é endereçada a uma rede *N* e o datagrama chega pela rede *N*, um roteador só precisa manter uma cópia para a pilha do protocolo local. No entanto, se uma transmissão direcionada para a rede *N* chega a outra rede, o roteador deve manter uma cópia e também transmitir por broadcast uma cópia através da rede *N*.

8.15 Roteadores software e lookup sequencial

Nossa descrição do algoritmo de encaminhamento implica que o IP procure uma tabela de encaminhamento sequencial-mente. Para roteadores low-end, é de fato utilizada a busca sequencial. Roteadores low-end são frequentemente chamados de *roteadores de software* para enfatizar que o roteador não tem hardware de propósito especial para auxiliar em

encaminhamento. Em vez disso, um roteador de software consiste em um computador de uso geral com um processador, memória e placas de interface de rede. Todo o encaminhamento de IP é feito por software.

Roteadores de software podem usar busca sequencial na tabela, pois elas são pequenas. Por exemplo, considere um host típico. Esperamos que a tabela de encaminhamento contenha duas entradas, como a Figura 8.3 ilustra. Em tal situação, a busca sequencial funciona bem. Na verdade, as técnicas de pesquisa mais sofisticadas apenas pagam por tabelas maiores – a sobrecarga de iniciar uma pesquisa significa que a pesquisa sequencial ganha para tabelas pequenas.

Apesar de ser útil em roteadores de software, a pesquisa sequencial não é suficiente para todos os casos. Um roteador high-end perto do centro da Internet IPv4 tem cerca de 400.000 entradas na sua tabela de encaminhamento. Nesses casos, a pesquisa sequencial na tabela leva muito tempo. A estrutura de dados mais comum usada para tabelas de encaminhamento de high-end consiste em uma *trie*.* Não é importante saber os detalhes de uma estrutura de dados trie e o algoritmo de pesquisa usado para procurá-la, mas devemos estar conscientes de que os roteadores high-end usam mecanismos mais sofisticados do que os descritos neste capítulo.

8.16 Estabelecendo tabelas de encaminhamento

Discutimos o algoritmo de encaminhamento IP e descrevemos como o encaminhamento utiliza uma tabela. No entanto, não especificamos como os hosts ou roteadores inicializam suas tabelas de encaminhamento, nem descrevemos como o conteúdo das tabelas de encaminhamento é atualizado conforme as alterações na rede. Os últimos capítulos lidam com essas questões e discutem protocolos que permitem aos roteadores manterem tabelas de encaminhamento consistentes. Por enquanto é importante entender que o software IP sempre usa uma tabela de encaminhamento, para decidir como encaminhar um datagrama. A consequência é que alterar os valores em uma tabela de encaminhamento vai mudar os caminhos que os datagramas seguem.

8.17 Resumo

O software IP encaminha datagramas; a computação consiste em usar o endereço IP de destino e encaminhar a informação. A entrega direta é possível se a máquina de destino estiver em uma rede à qual a máquina emissora se conecta. Se o emissor não puder alcançar o destino diretamente,

deverá encaminhar o datagrama a um roteador. O paradigma geral é que os hosts enviam datagramas encaminhados indiretamente ao roteador mais próximo; os datagramas viajam na internet de roteador para roteador até que o último roteador do caminho possa fazer sua entrega diretamente ao destino final.

O IP mantém as informações necessárias para o encaminhamento em uma tabela conhecida como tabela de encaminhamento. Quando o IP encaminha um datagrama, o algoritmo de encaminhamento produz o endereço IP da máquina seguinte (ou seja, o endereço do próximo salto) ao qual o datagrama deve ser enviado; o IP passa o datagrama e o endereço do próximo salto para o software da interface de rede. Este o encapsula em um frame de rede, mapeia o endereço internet do próximo salto para um endereço de hardware, usa o esse endereço de hardware como um frame de destino e o envia através da rede de hardware subjacente.

O encaminhamento de Internet só usa endereços IP; a ligação entre um endereço IP e um endereço de hardware não faz parte da função de encaminhamento do IP. Como cada entrada na tabela de encaminhamento inclui uma máscara de endereço, um único algoritmo de encaminhamento unificado pode lidar com rotas específicas de rede, rotas de hosts específicos e rota default.

EXERCÍCIOS

8.1 Crie tabelas de encaminhamento para todos os roteadores da Figura 8.2. Qual roteador ou quais roteadores se beneficiarão mais com o uso de uma rota default?

8.2 Examine o algoritmo de encaminhamento usado no seu sistema operacional local. Todos os casos de encaminhamento mencionados no capítulo são abordados? O algoritmo permite algo não descrito no capítulo?

8.3 Quando um roteador modifica o campo *limite de salto* (ou *tempo de vida*) no cabeçalho de um datagrama?

8.4 Considere uma máquina com duas conexões de rede físicas e dois endereços IP, I_1 e I_2. É possível que essa máquina receba um datagrama destinado a I_2 pela rede com endereço I_1? Explique.

8.5 No exercício anterior, qual é a resposta apropriada se essa situação surgir?

8.6 Considere dois hosts, *A* e *B*, que se conectam a uma rede física comum, *N*. O que acontece se outro host na rede enviar para *A* um datagrama que tem destino de IP de *B*?

8.7 Modifique o algoritmo 8.1 para acomodar as opções de fonte de rota do IPv4 discutidas no Capítulo 7.

8.8 Quando encaminha um datagrama, um roteador realiza cálculos que tomam tempo de forma proporcional ao tamanho do cabeçalho do datagrama. Explique o cálculo.

8.9 Na questão anterior, você pode achar uma otimização que executa os cálculos com poucas instruções de máquina?

8.10 Um administrador de rede quer monitorar o tráfego destinado para o host *H* e comprou um roteador *R* com software de monitoramento. O gerente só quer o tráfego destinado para *H* que passa por *R*. Explique como organizar o encaminhamento para atender ao gerente.

8.11 O algoritmo 8.1 permite a um gerente especificar encaminhamento para um endereço multicast? Explique.

8.12 O algoritmo 8.1 se aplica a fragmentos ou somente a datagramas completos? Explique.

---

* A tabela é conhecida como "tabela de roteamento" (opção adotada), mas alguns profissionais de rede usam a terminologia "tabela de encaminhamento".

* A única exceção ocorre quando um datagrama contém uma opção de fonte de rota.

* Pronunciado "try".

CAPÍTULO 9

Protocolo de Internet: mensagens de erro e controle (ICMP)

CONTEÚDOS DO CAPÍTULO

9.1 Introdução

O capítulo anterior descreveu o IP como um mecanismo de melhor esforço que faz uma tentativa de entregar datagramas, mas não garante a entrega e mostrou como o Protocolo Internet faz para que cada roteador encaminhe datagramas para seus destinos. Um datagrama viaja através da Internet, de um roteador para outro, até que alcance um que possa entregá-lo diretamente para o seu destino final. Se um roteador não puder encaminhar ou entregar um datagrama, ou se o roteador detectar uma condição incomum que afete sua capacidade de realizar o encaminhamento (por exemplo, falha de rede), o roteador precisará informar a origem sobre o problema. Este capítulo discute um mecanismo que os roteadores e hosts utilizam para comunicar tais informações de controle ou erro. Veremos que os roteadores usam o mecanismo para relatar problemas e os hosts o utilizam para achar vizinhos e testar se os destinos são alcançáveis.

9.2 Internet Control Message Protocol (ICPM)

No sistema sem conexão que descrevemos até aqui, cada roteador opera de forma autônoma. Quando um datagrama chega, um roteador encaminha ou entrega o datagrama e então segue para o próximo; o roteador não coordena com o remetente original de um datagrama. Esse sistema funciona bem se todos os hosts e roteadores estiverem configurados corretamente, pois concordam com as rotas para cada destino.

Infelizmente, nenhum grande sistema de comunicação funciona corretamente o tempo todo. Além de falhas de rede e processadores, o IP pode não conseguir entregar datagramas quando a máquina de destino estiver temporária ou permanentemente desconectada da rede, quando o contador de tempo de vida expirar, ou quando roteadores intermediários ficarem tão congestionados que tenham que descartar o datagrama. A diferença importante entre ter uma única rede implementada com hardware homogêneo e dedicado e uma internet implementada com vários sistemas independentes é que, no primeiro caso, o desenvolvedor pode programar o hardware subjacente para informar aos hosts conectados quando surgirem problemas. Em uma internet que não possui tal mecanismo de hardware, um emissor não pode dizer se uma falha de entrega resultou de um defeito local ou de falha do sistema em algum ponto do caminho para o destino. A depuração nesse panorama se torna extremamente difícil. O próprio protocolo IP não contém nada para ajudar o emissor a testar a conectividade ou descobrir tais falhas. Embora tenhamos dito que o IP não é confiável, queremos, sempre que possível, que nossa internet detecte e se recupere de erros.

Para permitir que os roteadores em uma internet informem erros ou ofereçam informações sobre circunstâncias inesperadas, os desenvolvedores incluíram um mecanismo de mensagem de uso especial aos protocolos TCP/IP. O mecanismo, conhecido como *Internet Control Message Protocol*(*ICMP*), é considerado uma parte obrigatória do IP e precisa ser incluído em toda implementação IP.*

O ICMP é em princípio destinado a informar uma fonte quando um datagrama enviado por ela enfrenta problemas. Entretanto, o destino final de uma mensagem ICMP não é um programa aplicativo em execução no computador de origem ou o usuário que lançou a aplicação. Em vez disso, mensagens ICMP são enviadas para o software Protocolo Internet no computador de origem. Ou seja, quando uma mensagem de erro ICMP chega a um computador, o módulo de software ICMP lida com ela. Claro que o ICMP pode tomar outras ações em resposta à mensagem que chegou. Por exemplo, ICMP pode informar um aplicativo ou um protocolo de maior nível sobre essa mensagem. Podemos então fazer o resumo a seguir.

O Internet Control Message Protocol permite que os roteadores enviem mensagens de erro ou controle de volta para a origem de um datagrama que causou problema. Mensagens ICMP não são

normalmente entregues para aplicativos. Pensamos no ICMP como fornecedor de comunicação entre módulos ICMP de duas em uma máquina e um módulo ICMP em outra.

Inicialmente projetado para permitir que os roteadores informem a causa dos erros de entrega aos hosts, o ICMP não é restrito a roteadores. Embora as diretrizes especifiquem que algumas mensagens ICMP só possam ser enviadas por roteadores, qualquer máquina pode enviar uma mensagem ICMP a qualquer outra máquina. Assim, um host pode usar ICMP para se corresponder com um roteador ou outro host. A principal vantagem de permitir que os hosts utilizem ICMP é este que oferece um único mecanismo, usado para todas as mensagens de controle e informação.

9.3 Relato de erro *versus* correção de erro

Tecnicamente, o ICMP é um *mecanismo de relato de erro*. Ele fornece um meio para que os roteadores que encontram um erro o relatem à origem do datagrama, mas não interage com o host nem faz tentativa para corrigir o erro. A ideia de relatar problemas em vez de trabalhar para corrigir problemas surge do princípio fundamental de projeto discutido anteriormente: roteadores devem ser tão *stateless* quanto possível. Notamos que a ideia de relatar erros em vez de corrigi-los ajuda a melhorar a segurança. Se um roteador tentar manter *state* quando um erro ocorrer, um invasor pode simplesmente inundá-lo com pacotes incorretos e tanto não responder como responder muito devagar quando esse roteador tentar corrigir o problema. Tal ataque poderia exaurir os recursos do roteador. Então, a ideia de somente relatar erros pode prevenir certos riscos de ataques.

Embora a especificação do protocolo esboce os usos intencionados do ICMP e sugira possíveis ações a serem tomadas em resposta aos relatórios de erro, o ICMP não especifica a ação a ser tomada para cada erro possível. Dessa forma, hosts têm a flexibilidade de decidir como relacionar relatórios de erro a aplicações. Podemos fazer o resumo a seguir.

Quando um datagrama causa um erro, o ICMP só pode relatar a condição de erro de volta à origem do datagrama; a origem precisa

relacionar o erro a um aplicativo individual ou tomar outra ação para corrigir o problema.

A maioria dos erros vem da origem, mas outros não. Como o ICMP relata problemas à origem, ele não pode ser usado para informar aos roteadores intermediários sobre os problemas. Por exemplo, suponha que um datagrama percorra um caminho por uma sequência de roteadores *R1*, *R2*, …, *Rk*. Se *Rk* tiver informações de roteamento incorretas e, por engano, encaminhar o datagrama ao roteador *RE*, *RE* não poderá usar o ICMP para informar o erro ao roteador *Rk*. O ICMP só pode enviar um relatório de volta à origem. Infelizmente, a origem não tem responsabilidade pelo problema ou controle sobre o roteador com erro de comportamento. Na verdade, a origem pode não ser capaz de determinar qual roteador causou o problema.

Por que restringir a comunicação do ICMP com a origem? A resposta deve estar clara pela nossa discussão sobre formatos e encaminhamento de datagrama, nos capítulos anteriores. Um datagrama só contém campos que especificam a origem e o destino final; ele não contém um registro completo de sua viagem pela rede (exceto para casos incomuns, em que a opção de registro de rota é utilizada). Além do mais, como os roteadores podem estabelecer e alterar suas próprias tabelas de roteamento, não existe um conhecimento global das rotas. Assim, quando um datagrama alcança determinado roteador, é impossível conhecer o caminho que percorreu para chegar lá. Se o roteador detectar um problema, o IP não poderá saber o conjunto de máquinas intermediárias que processaram o datagrama, de modo que não poderá informar-lhes quanto ao problema. Em vez de descartar silenciosamente o datagrama, o roteador utiliza o ICMP para informar a origem de que ocorreu um problema, e confia que os administradores do host cooperarão com os administradores da rede para localizá-lo e repará-lo.

9.4 Entrega de mensagem ICMP

Os desenvolvedores do ICMP tomaram uma nova abordagem para relatar erro: em vez de usar um sistema de comunicação de baixo nível para lidar com erros, escolheram usar o IP para transportar mensagens ICMP. Isso significa que, como todos os outros tráfegos, mensagens ICMP viajam pela

internet na área de carga útil dos datagramas IP. A escolha reflete uma consideração importante: erros são raros. Em particular, assumimos que o encaminhamento de datagrama permanecerá intacto na maior parte das vezes (ou seja, mensagens de erro serão entregues). Na prática, a consideração se tornou válida – erros são realmente raros.

Como cada mensagem ICMP viaja em um datagrama IP, dois níveis de encapsulamento são necessários. A Figura 9.1 ilustra o conceito.

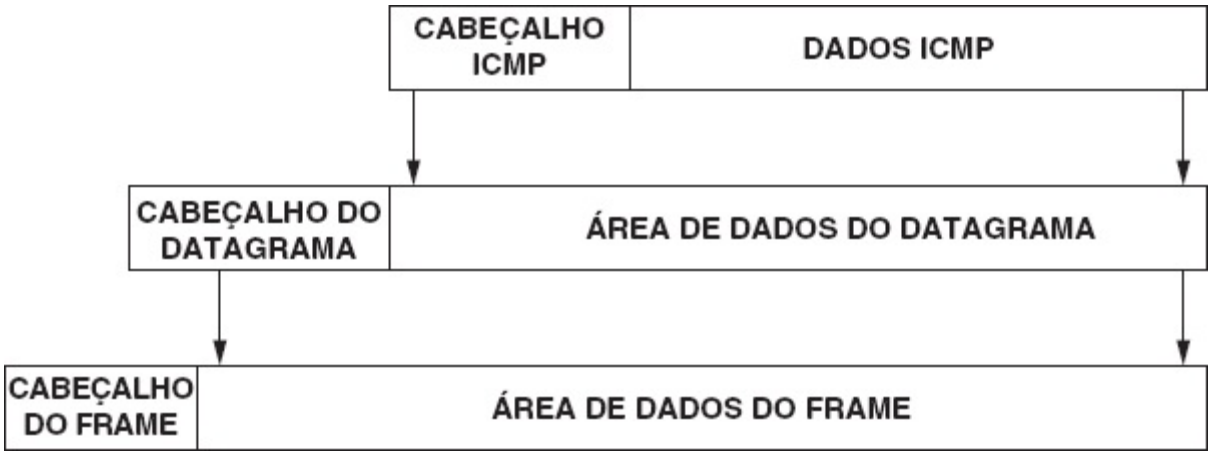


Figura 9.1 Os dois níveis de encapsulamento usados quando uma mensagem ICMP é enviada através da rede.

Como a figura mostra, cada mensagem ICMP viaja através da internet na parte de dados de um datagrama IP, que viaja, ele próprio, através de uma rede subjacente na parte de carga útil de um frame. Embora tanto o IPv4 quanto o IPv6 usem um datagrama para transportar uma mensagem ICMP, eles diferem em detalhes. O IPv4 usa o campo de PROTOCOLO no cabeçalho do datagrama como um campo de tipo. Quando uma mensagem ICMP é carregada na área de carga útil de um datagrama IPv4, o campo PROTOCOLO é determinado como 1. O IPv6 usa o campo PRÓXIMO ENDEREÇO para especificar o tipo de item que está sendo carregado. Quando uma mensagem ICMP é carregada na área de carga útil de um datagrama IPv6, o campo PRÓXIMO ENDEREÇO no cabeçalho que é anterior à mensagem ICMP contém 58.

Em termos de processamento, um datagrama que carrega uma mensagem ICMP é encaminhado exatamente como um datagrama que carrega informação para usuários; não há confiabilidade ou prioridade adicional. Então, as mensagens de erro propriamente ditas podem ser perdidas, duplicadas ou descartadas. Além disso, em uma rede já congestionada, a mensagem de erro pode aumentar o congestionamento. Uma exceção é feita

para os procedimentos de tratamento de erro se um datagrama IP carregando uma mensagem ICMP causar um erro. A exceção, estabelecida para evitar o problema de se ter mensagens de erro sobre mensagens de erro, especifica que mensagens ICMP não são geradas por erros que resultem de datagramas carregando mensagens ICMP de erro.

9.5 Layering conceitual

Geralmente, encapsulamento e layering andam lado a lado. Por exemplo, considere I a Ethernet IP. Quando ela viaja através de uma Ethernet, um datagrama IP é encapsulado em um frame Ethernet. O encapsulamento segue o esquema de layering apresentado no Capítulo 4, pois o IP é um protocolo da camada 3 (*layer 3*) e Ethernet é uma tecnologia da camada 2 (*layer 2*). O campo tipo Ethernet permite que uma variedade de pacotes de camada mais alta (*higher-layer packets*) seja encapsulada em frames Ethernet sem que haja ambiguidade.

O ICMP representa uma exceção importante. Embora cada mensagem ICMP seja encapsulada em um datagrama IP, o ICMP não é considerado um protocolo de maior nível. Em vez disso, ele é uma parte obrigatória do IP, o que significa que é classificado como um protocolo da camada 3. Podemos pensar no encapsulamento como o uso do esquema de roteamento com base no IP existente em vez da criação de um mecanismo paralelo de encaminhamento para mensagens ICMP. O ICMP deve enviar relatórios de erro para a fonte original, então uma mensagem ICMP deve viajar através de múltiplas redes subjacentes para atingir seu destino final. Assim, uma mensagem ICMP não pode ser entregue por um transporte de camada 2 sozinho.

9.6 Formato de mensagem ICMP

As normas definem dois conjuntos de mensagens ICMP: um conjunto para IPv4 e um conjunto maior para IPv6. Em ambas as versões de IP, cada mensagem ICMP tem seu próprio formato. Entretanto, todas as mensagens ICMP começam com os mesmos três campos. A Figura 9.2 ilustra o formato geral de uma mensagem ICMP.

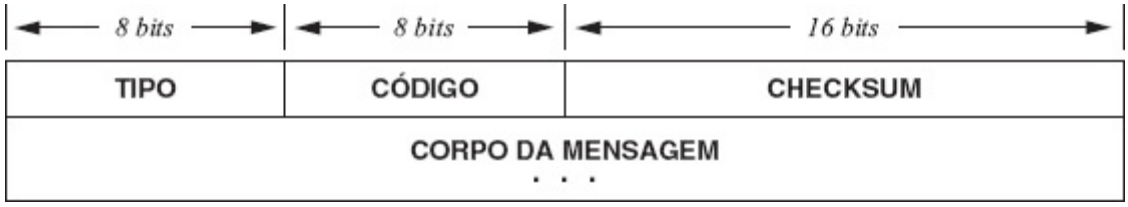


Figura 9.2 Os três primeiros campos em cada mensagem ICMP.

Como a figura mostra, uma mensagem ICMP começa com uma mensagem ICMP integral de 8 bits no campo TIPO . O campo TIPO identifica a mensagem ICMP específica que vem a seguir. Devido ao formato da mensagem ser definido pelo tipo da mensagem, um receptor usa o valor no campo TIPO para saber como analisar o restante da mensagem.

Um campo CÓDIGO de 8 bits em uma mensagem ICMP fornece mais informação sobre o tipo da mensagem. Por exemplo, uma mensagem ICMP TEMPO EXCEDIDO (TIME EXCEEDED) pode ter um valor de código para indicar que o contador de salto (hop count – TTL) do datagrama chegou a zero ou que o prazo para a reconstituição expirou antes de todos os fragmentos chegarem.

O terceiro campo em cada mensagem ICMP consiste de um CHECKSUM de 16 bits que é computado sobre a mensagem ICMP inteira. O ICMP usa o mesmo complemento checksum 16 bits que o IP.

O corpo da mensagem em uma mensagem ICMP depende inteiramente do tipo do ICMP. Entretanto, para mensagens ICMP que relatam um erro, o corpo da mensagem sempre inclui o cabeçalho mais octetos adicionais do datagrama que causou o problema.*

A razão para que o ICMP retorne mais do que somente o cabeçalho do datagrama é permitir que o receptor determine mais precisamente qual(is) protocolo(s) e quais programas aplicativos foram responsáveis pelo datagrama. Como veremos mais adiante, protocolos de maior nível no comutador TCP/IP são designados para que informações cruciais sejam codificadas em alguns dos primeiros octetos além do cabeçalho IP.**

9.7 Exemplo de tipos de mensagem ICMP usados com IPv4 e IPv6

A Figura 9.3 lista exemplos de tipos de mensagens ICMP utilizadas com IPv4. As seções seguintes descrevem os significados mais detalhadamente e dão exemplos de formatos de mensagem.

Tipo	Significado

0	Resposta de eco
3	Destino inalcançável
4	Extinção de origem
5	Redirecionamento (mudar uma rota)
6	Endereço de host alternativo
8	Requisição de eco
9	Anúncio de Roteador
10	Solicitação de Roteador
11	Tempo excedido
12	Problema de parâmetro
13	Requisição de estampa de tempo
14	Resposta de estampa de tempo
15	Requisição de informação
16	Resposta de informação
17	Requisição de máscara de endereço
18	Resposta de máscara de endereço
30	Traceroute
31	Erro de conversão de datagrama
32	Redirecionar host móvel
33	Onde está você (do IPv6)
34	Eu estou aqui (do IPv6)
35	Requisição de registro móvel
36	Resposta de registro móvel
37	Requisição de nome de domínio
38	Resposta de nome de domínio
39	SKIP (Simple Key Mgmt)
40	Photuris
41	Mobilidade experimental

Figura 9.3 Exemplo de tipo de mensagem ICMPv4 e significado de cada uma. Valores não listados são não atribuídos ou reservados.

Como a figura mostra, muitas das mensagens originais ICMP foram designadas para carregar informações em vez de mensagens (ou seja, um host usa o tipo 17 para requisitar a máscara de endereço que está sendo

usada em uma rede e um roteador responde com o tipo 18 para informar a máscara de endereço). O IPv6 faz distinção entre mensagens de erro e mensagens de informação dividindo o valor tipo em dois conjuntos: tipos menores que 128 são usados para mensagens de erro e tipos entre 128 e 255 são usados para mensagem de informação. A Figura 9.4 lista exemplos de tipos de mensagem ICMP usados com IPv6 e mostra que, embora somente quatro mensagens de erro tenham sido definidas, o IPv6 define muitas mensagens de informação.

Tipo	Significado
1	Destino inacessível
2	Pacote muito grande
3	Tempo excedido
4	Problema de parâmetro
128	Requisição de eco
129	Resposta de eco
130	Multicast Listener Query
131	Multicast Listener Report
132	Multicast Listener Done
133	Solicitação de roteador (NDP)
134	Anúncio de roteador (NDP)
135	Solicitação de vizinho (NDP)
136	Anúncio de vizinho (NDP)
137	Redirecionar mensagem
138	Renumeração de roteador
139	Pesquisa de Info. sobre nós ICMP
140	Resposta de Info. sobre nós ICMP
141	Solicitação de vizinho inverso
142	Anúncio de vizinho inverso
143	Multicast Listener Reports
144	Home Agent Request
145	Home Agent Reply
146	Solicitação de prefixo móvel
147	Anúncio de prefixo móvel
148	Solicitação de certificação de caminho

149 Anúncio de certificação de caminho

151 Anúncio de roteador Multicast

Figura 9.4 Exemplo de tipos de mensagem ICMPv6 e o significado de cada uma.

Como mostra a figura, o IPv6 incorpora três subsistemas principais no ICMP: solicitação de vizinho (*Neighbor Discovery Protocol – NDP*), mencionado no Capítulo 6, suporte multicast, descrito no Capítulo 15, e mobilidade de IP (IP mobility), descrito no Capítulo 18. Mensagens ICMP foram definidas para cada um dos subsistemas. Por exemplo, quando se usa solicitação de vizinho (Neighbor Discovery), um nó IPv6 pode transmitir por broadcast uma *mensagem de solicitação de vizinho* (*Neighbor Solicitation Message* – tipo 135) para descobrir vizinhos diretamente encontráveis (directly-reachable neighbors) ou uma *mensagem de solicitação de roteador* (*Router Solicitation Message* – type 133) para encontrar roteadores diretamente encontráveis (directly-reachable routers).

9.8 Teste de alcance e status do destino (ping)

O programa ping é, talvez, a ferramenta de diagnóstico mais amplamente usada. Originalmente criado para o IPv4, o ping foi estendido para se acomodar ao IPv6. Em cada caso, o ping envia uma mensagem ICMP *requisição de eco (echo request)* a um computador remoto. Qualquer computador que receba um ICMP Requisição de Eco cria um ICMP *resposta de eco (Echo Reply)* e retorna a resposta ao emitente original. Então, o programa ping recebe uma resposta de eco de uma máquina remota. A mensagem de requisição contém uma seção para dados, e a resposta contém uma cópia dos dados que foram enviados na requisição.

Como pode uma simples troca de mensagem ajudar no diagnóstico de problemas de internet? Quando enviar uma requisição de eco, um usuário deve especificar um destino. Uma resposta direta é que uma requisição e uma resposta associada podem ser usadas para testar se um destino é atingível e se está respondendo. Como ambas, a requisição e a resposta, trafegam em datagramas IP, receber a resposta de uma máquina remota confirma que as maiores parte do sistema de transporte IP estão funcionando corretamente. Primeiro, o software IP no computador de origem deve ter uma entrada em sua tabela de encaminhamento para o

destino. Segundo, o computador de origem deve ter criado um datagrama correto. Terceiro, a fonte deve estar apta a alcançar um roteador, o que significa que a ARP (IPv4) ou a solicitação de vizinho (IPv6) está funcionando. Quarto, roteadores intermediários entre a origem e o destino devem estar operando e devem encaminhar datagramas corretamente em ambas as direções entre a origem e o destino. Finalmente, a máquina de destino deve estar funcionando, o driver do dispositivo deve estar apto a receber e enviar pacotes, e tanto o ICMP quanto o módulo de software IP devem estar funcionando.

Existem várias versões de ping. A maioria inclui opções que permitem a um usuário especificar se envia uma requisição e espera por resposta, envia várias requisições e então espera por resposta, ou envia requisições periodicamente (por exemplo, a cada segundo) e mostra todas as respostas. Se o ping envia uma série de requisições, ele mostra estatísticas sobre mensagem perdida. A vantagem de enviar uma série contínua de requisitos surge de uma habilidade para encontrar problemas intermitentes. Por exemplo, considere uma rede wireless onde interferência elétrica causa perda, mas ocorre randomicamente (por exemplo, quando inicia uma impressão).

A maior parte das versões de ping também permite ao usuário especificar a quantidade de dados que são enviados em cada requisição. Enviar um grande pacote ping é útil para testar fragmentação e reconstituição. Pacotes grandes também forçam o IPv6 a se engajar em uma path MTU discovery. Então, uma aplicação aparentemente trivial tem diversos usos.

9.9 Requisição de eco e formato de mensagem de resposta

Tanto o IPv4 quanto o IPv6 usam um único formato para todas as mensagens de Requisições de Eco ICMP e Resposta de Eco. A Figura 9.5 ilustra o formato da mensagem.

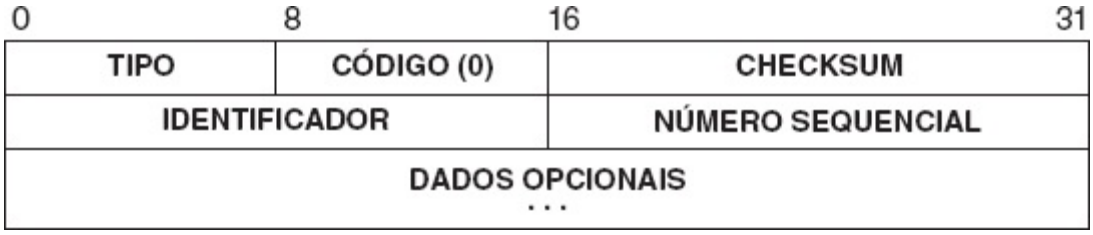


Figura 9.5 Formato de mensagem ICMP de requisição ou resposta de eco.

Embora o mesmo formato de mensagem seja usado para requisições e respostas de eco, o valor do TIPO é diferente. Para o IPv4, o TIPO é *8* em uma requisição e *0* em uma resposta. Para o IPv6, o TIPO é *128* em uma requisição e *129* em uma resposta. Para qualquer valor no campo TIPO, o CÓDIGO é zero (ou seja, as solicitações e respostas de eco não usam o campo código).

Os campos IDENTIFICADOR e NÚMERO SEQUENCIAL são usados pelo emissor para coincidir respostas e solicitações. Um recebimento ICMP não interpreta os dois campos, mas retorna na resposta o mesmo valor que foi encontrado na requisição. Dessa maneira, uma máquina que envia uma requisição pode colocar um valor no campo IDENTIFICADOR que identifique uma aplicação, e pode usar o campo NÚMERO SEQUENCIAL para numerar sucessivas requisições enviadas pela aplicação. Por exemplo, o IDENTIFICADOR deve ser o ID de processo da aplicação enviada, que permite ao software ICMP coincidir respostas de entrada com a aplicação que envia a requisição.

O campo denominado DADOS OPCIONAIS é um campo de tamanho variável que contém dados a serem retornados ao emitente. Uma resposta eco sempre retorna exatamente os mesmos dados como tinha recebido na requisição. Embora dados arbitrários possam ser enviados, programas ping típicos armazenam valores sequenciais em octetos da área de dados, tornando fácil verificar que os dados que retornaram são exatamente os mesmos que foram enviados sem a necessidade de armazenar cópias de pacotes. Como mencionado anteriormente, o tamanho variável permite que um gerente teste a fragmentação.

9.10 Computação checksum e o pseudocabeçalho IPv6

Tanto o IPv4 quanto o IPv6 usam o campo CHECKSUM em mensagem ICMP, e ambos requerem que um emissor compute um complemento checksum de 16 bits da mensagem completa. Além disso, ambas as versões requerem um receptor para validar o checksum e descartar mensagem ICMP que tenha checksum inválido. Entretanto, os detalhes da computação do checksum diferem, pois o IPv6 acrescenta um requisito adicional: o checksum usado com IPv6 também abrange campos do cabeçalho base IP. Conceitualmente, os campos de cabeçalho designados são organizados em um *pseudocabeçalho*, como ilustra a Figura 9.6.

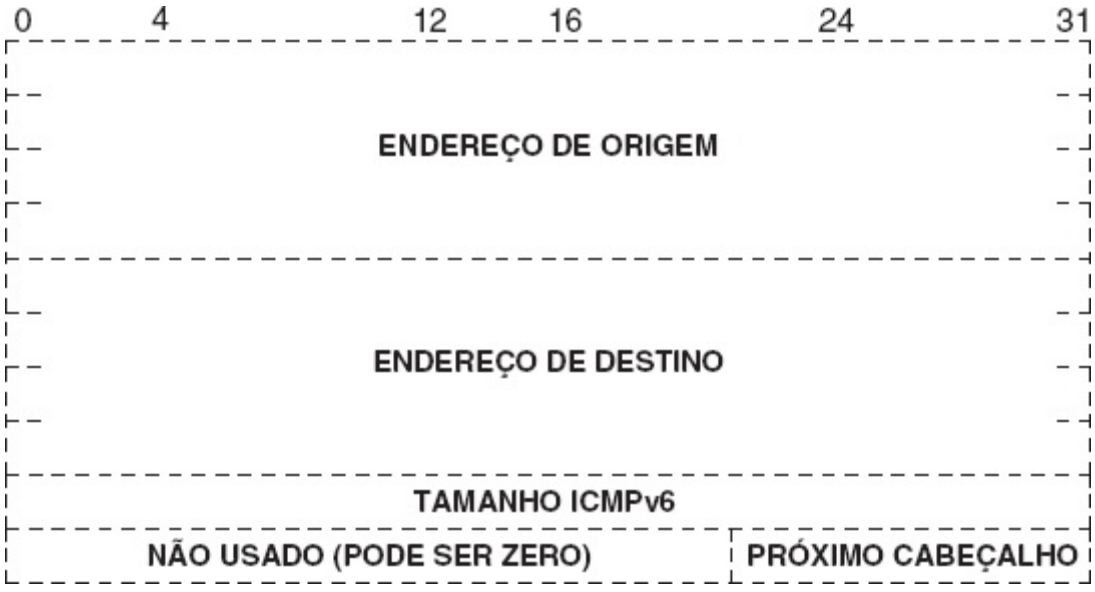


Figura 9.6 O pseudocabeçalho IPv6 usado para computação do checksum com ICMPv6.

O termo *pseudocabeçalho* e o uso de linhas tracejadas na figura indicam que o arranjo de campos extra é meramente usado para computação checksum, e nunca é colocado em um pacote. Podemos imaginar, por exemplo, que o código checksum cria um pseudocabeçalho na memória copiando campos do cabeçalho-base, anexa uma cópia da mensagem ICMP para o pseudocabeçalho e, em seguida, computa um checksum em ambos, pseudocabeçalho e mensagem.*

Por que o IPv6 incluiu um pseudocabeçalho no checksum? Os desenvolvedores do IPv6 estavam conscientes de possíveis falhas de segurança, e queriam assegurar que um computador não processaria erroneamente uma mensagem ICMP que não se destinasse a ele. A inclusão de um pseudocabeçalho no checksum acrescenta uma verificação adicional de que a mensagem foi entregue corretamente. O pseudocabeçalho não garante a precisão – se for necessária uma segurança melhor, o datagrama deverá ser criptografado.

9.11 Relatos de destino inalcançável

Embora o IP implemente um mecanismo de entrega de melhor esforço, o descarte de datagramas não deve ser tomado levianamente. Sempre que um erro impedir um roteador de encaminhar ou entregar um datagrama, o roteador envia de volta para a fonte uma mensagem ICMP de *destino*

inalcançável e depois *desiste* (ou seja, descarta) do datagrama. Erros de rede inalcançável implicam falhas de encaminhamento em pontos intermediários; erros de host inalcançável implicam falhas na entrega ao longo do último salto.**

Tanto o IPv4 quanto o IPv6 usam o mesmo formato para mensagens de destino inalcançável. A Figura 9.7 ilustra esse formato.

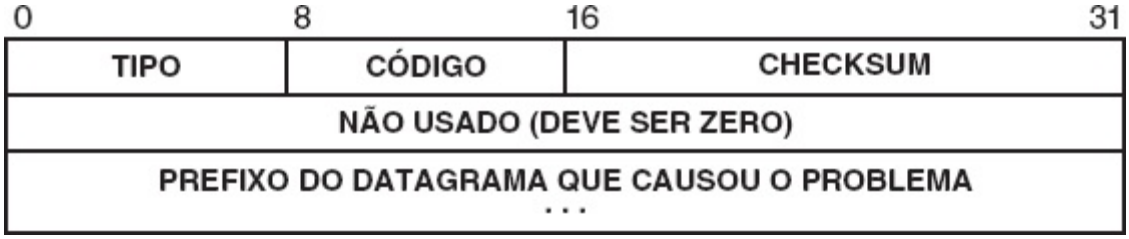


Figura 9.7 Formato de mensagem ICMP de destino inalcançável.

Apesar de utilizar o mesmo formato de mensagem, a forma como o IPv4 e o IPv6 interpretam campos na mensagem é um pouco diferente. O IPv4 define o TIPO como *3* e o IPv6 define o TIPO como *1*. Tal como acontece no caso das mensagens de requisição de eco e mensagens de resposta, o IPv4 calcula o CHECKSUM apenas para a mensagem ICMP, e o IPv6 inclui um pseudocabeçalho no checksum.

O campo de CÓDIGO contém um inteiro que descreve ainda mais os problemas; os códigos para o IPv4 e o IPv6 são diferentes. A Figura 9.8 lista o significado dos valores de CÓDIGO.

Interpretação IPv4

Código	Significado
0	Rede inalcançável
1	Host inalcançável
2	Protocolo inalcançável
3	Porta inalcançável
4	Fragmentação necessária
5	Rota de origem falhou
6	Rede de destino desconhecida
7	Host de destino desconhecido
8	Host de origem isolado

9	Comunicação com rede proibida
10	Comunicação com host proibida
11	Rede inalcançável para TDS
12	Host inalcançável para TDS
13	Comunicação proibida
14	Violação de precedência de
15	Host Corte de precedência

Interpretação IPv6

Código	Significado
0	Sem rota para o destino
1	Comunicação proibida
2	Além do escopo src.
3	Endereço inalcançável
4	Porta inalcançável
5	Endereço de origem falhou
6	Rota para destinorejeitada
7	Erro de rota de origem

Figura 9.8 Os valores de CÓDIGO para uma mensagem de destino ICMP inalcançável.

As duas versões do IP também diferem na maneira de selecionar um prefixo do datagrama que causou o problema. O IPv4 envia o cabeçalho do datagrama mais os primeiros 64 bits da carga útil do datagrama. O IPv6 permite que o datagrama que leva a mensagem ICMP seja de até 1280 octetos (o mínimo MTU IPv6), e escolhe um tamanho máximo de prefixo em conformidade. Como a mensagem de erro ICMP contém um prefixo curto do datagrama que causou o problema, a fonte vai saber exatamente qual endereço está inalcançável.

Destinos podem estar inalcançáveis devido ao hardware estar temporariamente fora de serviço, porque o remetente especificou um endereço de destino inexistente ou (em raras circunstâncias) porque o roteador não tem uma rota para a rede de destino. Note-se que, embora os roteadores reportem falhas que encontram, eles podem não detectar todas as falhas de entrega. Por exemplo, se a máquina de destino se conecta a uma

rede Ethernet, o hardware de rede não fornece confirmações. Portanto, um roteador pode continuar a enviar pacotes para um destino após este estar desconectado, sem receber qualquer indicação de que os pacotes não estão sendo entregues. Podemos então fazer o resumo a seguir.

Embora um roteador envie uma mensagem de destino inalcançável quando encontra um datagrama que não pode ser enviado ou entregue, um roteador não consegue detectar todos esses erros.

O significado de mensagens de porta inalcançável ficará claro quando estudarmos como os protocolos de maior nível utilizam pontos de destino abstratos chamados *portas*.

Muitos dos códigos restantes são autoexplicativos. Por exemplo, um site deve escolher restringir certos datagramas de entrada ou saída por razões administrativas. No IPv6, alguns endereços são classificados como *site-local*, significando que não podem ser usados na Internet global. A tentativa de enviar um datagrama que tem um endereço de origem site-local vai disparar um erro de que o datagrama foi enviado além do escopo válido de endereço de origem.

9.12 Relatos de erro com relação à fragmentação

O IPv4 e o IPv6 permitem a um roteador reportar um erro quando um datagrama é muito grande para uma rede através da qual deve viajar e não pode ser fragmentado. No entanto, diferem em pormenores. O IPv4 envia uma mensagem de *destino inalcançável* com o campo CÓDIGO definido como *4*. Já o IPv6 envia uma mensagem de *pacote muito grande*, que tem o campo TIPO de *2*. Pode parecer que relatos de fragmentação não são necessários para o IPv4 porque um roteador pode fragmentar um datagrama IPv4. Lembre-se de que um cabeçalho IPv4 inclui um bit "não fragmentar". Quando o bit é definido, um roteador está impedido de realizar a fragmentação, o que faz com que envie uma mensagem ICMPv4 de *destino inalcançável* com o CÓDIGO *4*.

A razão de o IPv6 definir uma mensagem ICMP separada para relatar problemas de fragmentação decorre do projeto. Roteadores são sempre

proibidos de fragmentar um datagrama IPv6, o que significa que uma fonte deve executar uma solicitação de caminho MTU. Uma parte fundamental da solicitação de caminho MTU envolve receber informações sobre o MTU de redes remotas. Portanto, a mensagem de *pacote muito grande* contém um campo que um roteador usa para informar a fonte sobre o MTU da rede que causou o problema. A Figura 9.9 ilustra o formato da mensagem.

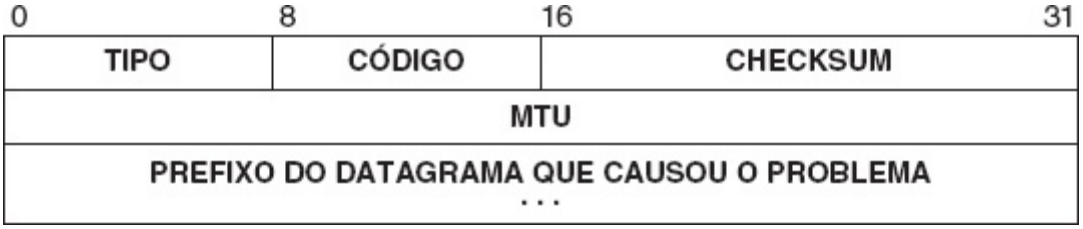


Figura 9.9 O formato de uma mensagem ICMPv6 *pacote muito grande*.

9.13 Requisições de mudança de rota dos roteadores

As tabelas de encaminhamento normalmente permanecem estáticas por longos períodos de tempo. Os hosts as inicializam por um arquivo de configuração na partida do sistema, e os administradores do sistema raramente mudam as tabelas durante as operações normais. Conforme veremos em capítulos posteriores, os roteadores são mais dinâmicos – eles trocam informações de roteamento periodicamente para acomodar mudanças de rede e manter suas tabelas de encaminhamento atualizadas. Assim, via de regra, podemos fazer a afirmação a seguir.

> *Os roteadores conhecem as rotas corretas; os hosts começam com informações mínimas e descobrem novas rotas a partir dos roteadores.*

Para seguir a regra e evitar o envio de informações por encaminhamento enquanto um host está sendo configurado, a configuração do host normalmente especifica o mínimo possível de informação de encaminhamento necessária para comunicação (ou seja, o endereço de um único roteador default). Então, um host pode começar com informação incompleta e contar com roteadores para atualizar suas tabelas de encaminhamento conforme necessário. Quando um roteador detecta um host usando um primeiro salto não ideal, envia a esse host uma mensagem ICMP *redirecionar*, que o instrui a mudar sua tabela de encaminhamento.

O roteador também encaminha o datagrama original para seu destino.

A vantagem do esquema de redirecionamento ICMP é a simplicidade: um host inicializa sem qualquer necessidade de baixar uma tabela de encaminhamento e pode se comunicar imediatamente com qualquer destino. Um roteador só envia uma mensagem de redirecionamento se o host enviar um datagrama ao longo de uma rota não ideal. Assim, a tabela de encaminhamento de host permanece pequena, mas usa as melhores rotas para todos os destinos.

Como um roteador e um host estão limitados a interações em uma rede conectada diretamente, redirecionar mensagens não resolve o problema da propagação de informações de roteamento de uma forma geral. Para entender o porquê, considere a Figura 9.10, que ilustra um conjunto de redes conectadas por roteadores.

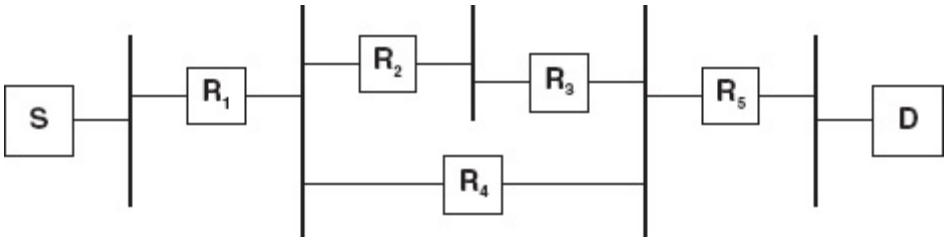


Figura 9.10 Exemplo de topologia mostrando por que mensagens ICMP *redirect* não resolvem todos os problemas de roteamento.

Na figura, o host S envia um datagrama para o destino D. Se roteador R_1 encaminhar incorretamente o datagrama através de roteador R_2 em vez de através de roteador R_4 (ou seja, R_1 escolhe incorretamente um caminho mais longo do que o necessário), o datagrama chegará ao roteador R_5. No entanto, R_5 não pode enviar uma mensagem de redirecionamento ICMP para R_1 porque R_5 não sabe o endereço de R_1. Os próximos capítulos exploram o problema de como propagar informações de roteamento através de múltiplas redes.

Tal como acontece com vários outros tipos de mensagens ICMP, o IPv4 e o IPv6 usam o mesmo formato geral para mensagens *redirect*. A mensagem começa com os requisitos dos campos TIPO, CÓDIGO, e CHECKSUM. A mensagem contém mais duas informações: o endereço IP de um roteador para usar como um primeiro salto e o endereço do destino que causou o

problema. Os formatos das mensagens têm diferenças. Uma mensagem IPv4 redirect contém o endereço IPv4 de 32 bits de um roteador, seguido pelo prefixo do datagrama que foi incorretamente encaminhado. Uma mensagem IPv6 redirect contém o endereço IPv6 de um roteador e o endereço IPv6 do destino que deveria ter sido encaminhado pelo roteador. A Figura 9.11 ilustra o formato usado com o IPv6.

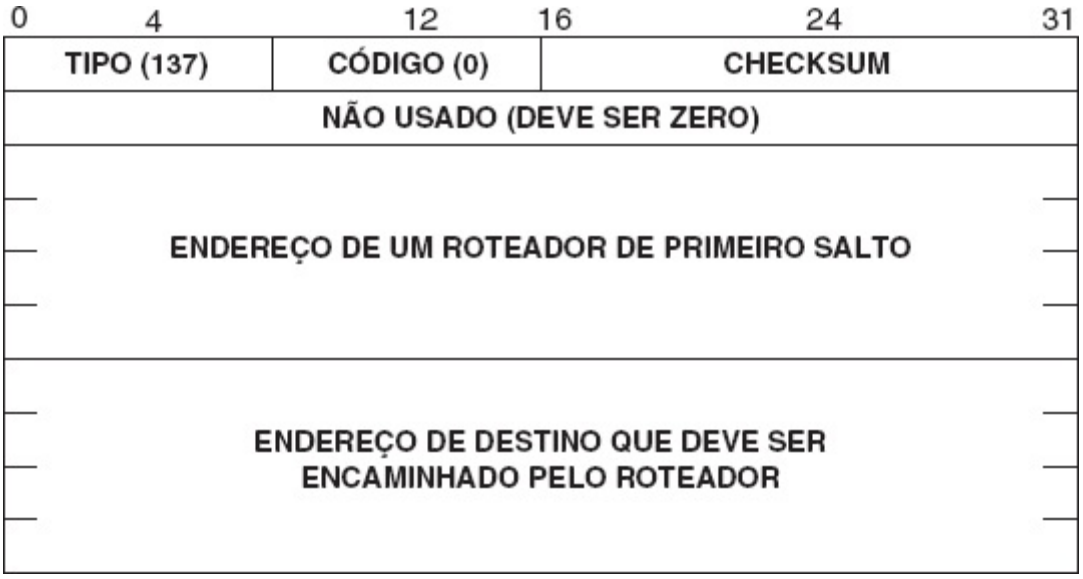


Figura 9.11 Formato de mensagem *redirect* ICMPv6.

Como regra geral, roteadores só enviam requisições ICMP de redirecionamento para hosts, e não para outros roteadores. Capítulos posteriores explicarão protocolos que roteadores usam para intercâmbio de informações de roteamento.

9.14 Detectando rotas circulares ou excessivamente longas

Como cada roteador da internet usa informações locais ao encaminhar datagramas, erros ou inconsistências no encaminhamento de informações podem produzir um ciclo conhecido como um *loop de roteamento* (*routing loop*) para um determinado destino, *D*. Um loop de roteamento pode consistir em dois roteadores que encaminham, cada um, um determinado datagrama para o outro, ou pode consistir de vários roteadores sendo que cada um transmite um datagrama para o roteador seguinte no ciclo. Se um datagrama entrar em um loop de roteamento, ele vai passar em torno do loop indefinidamente. Como mencionado anteriormente, para

evitar que os datagramas circulem para sempre em uma internet TCP/IP, cada datagrama IP contém um *limite de salto* (*hop limit*). Sempre que processa um datagrama, um roteador diminui o limite de salto e descarta o datagrama quando a contagem chegar a zero. Um roteador não se limita a descartar um datagrama que excedeu seu limite hop. Em vez disso, um roteador tem uma ação adicional de enviar para a fonte uma mensagem ICMP de *tempo excedido* (*time exceeded).*

Tanto o IPv4 como o IPv6 enviam uma mensagem de *tempo excedido*, e ambos usam o mesmo formato, como a Figura 9.12 ilustra. O IPv4 define o TIPO em *11* e o IPv6 define o TIPO em *3*.

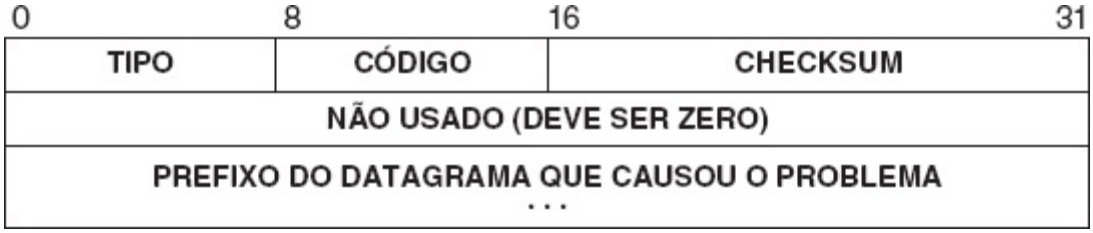


Figura 9.12 Formato da mensagem ICMPF tempo excedido.

Pode parecer estranho que a mensagem se refira ao momento em que o erro que está sendo relatado é um limite de salto. O nome é derivado de IPv4, que originalmente interpretava o limite de salto como um contador *tempo de vida* (*Time To Live – TTL*). Assim, fazia sentido interpretar expiração TTL como o fato de exceder a um limite de tempo. Embora a interpretação do campo tenha mudado, o nome persiste.

A mensagem *tempo excedido* também é usada para relatar outra expiração de tempo: tempo de espera (timeout) durante a reconstituição de um datagrama fragmentado. Lembre-se de que o software IP em um host de destino deve reunir os fragmentos e construir um datagrama completo. Quando um fragmento chega para um novo datagrama, o host aciona um timer. Se um ou mais fragmentos não chegarem antes de expirar o timer, os fragmentos serão descartados e o host receptor enviará uma mensagem *tempo excedido* de volta para a fonte. O ICMP usa o campo CÓDIGO em uma mensagem *tempo excedido* para explicar a natureza do tempo limite que está sendo relatado, como a Figura 9.13 mostra.

Significado

Valor do Código	
0	Limite de salto excedido
1	Tempo de reconstituição do fragmento excedido

Figura 9.13 Interpretação do campo *CÓDIGO* em uma mensagem ICMP *tempo excedido*. Tanto o IPv4 como o IPv6 usam a mesma interpretação.

9.15 Relatando outros problemas

Quando um roteador ou host encontra problemas com um datagrama não coberto por mensagens ICMP de erro anteriores (por exemplo, um cabeçalho de datagrama incorreto), ele envia uma mensagem de *problema de parâmetro* (*parameter problem*) para a fonte original. No IPv4, uma possível causa de tais problemas ocorre quando argumentos para uma opção estão incorretos. No IPv6, os problemas de parâmetros podem surgir se o valor em um campo de cabeçalho estiver fora da faixa, o tipo de CABEÇALHO SEGUINTE ou uma das opções não forem reconhecidos. Em tais casos, um roteador envia uma mensagem de *problema de parâmetro* e usa o campo CÓDIGO para distinguir entre subproblemas. A Figura 9.14 ilustra o formato de uma mensagem de *problema de parâmetro*. Essas mensagens só são enviadas quando um problema é tão grave que o datagrama deve ser descartado.

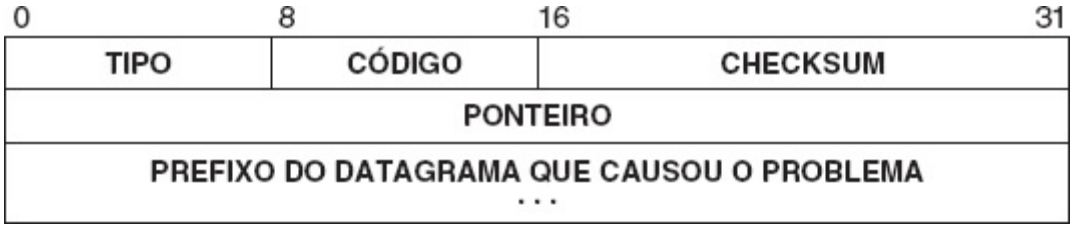


Figura 9.14 Formato da mensagem ICMP problema de parâmetro.

Para deixar a mensagem sem ambiguidade, o emissor usa o campo PONTEIRO no cabeçalho da mensagem para identificar o octeto que causou o problema no datagrama.

9.16 Mensagens ICMP mais antigas usadas na inicialização

Originalmente, o ICMP definiu um conjunto de mensagens que um host usava na inicialização para determinar o seu endereço IP, o endereço de um roteador e a máscara de endereço usada na rede. Finalmente, um protocolo conhecido como *DHCP** foi introduzido. Ele fornece um host IPv4 com todas as informações necessárias em uma única troca. Além disso, o ICMP definiu mensagens que um host ou roteador poderiam usar para obter a hora atual. Foram criados também protocolos para intercâmbio de informação de tempo, tornando a versão ICMP obsoleta. Como consequência, o IPv4 já não usa mensagens ICMP que foram projetadas para obter informações na inicialização.

Curiosamente, o ICMPv6 voltou a uma das ideias que era originalmente parte de ICMPv4: solicitação de roteador. Na inicialização, um host IPv6 transmite por multicast uma mensagem ICMPv6 *Solicitação de Roteador* (*Roteador Discovery*) para conhecer os roteadores na rede local. Há duas diferenças conceituais entre solicitação de roteador e DHCP que a tornam atraente para o IPv6. Primeiro, como a informação é obtida diretamente a partir do próprio roteador, nunca há um erro de terceiros. Com o DHCP, esses erros são possíveis porque um servidor DHCP deve ser configurado para distribuir informações. Se um gerente de rede falha ao atualizar a configuração DHCP depois de alterações na rede, os hosts podem receber informações desatualizadas. Em segundo lugar, a solicitação de roteador ICMP usa uma técnica *soft state* com timers para evitar que hosts mantenham uma entrada na tabela de roteamento após uma falha de um roteador – roteadores anunciam suas informações periodicamente, e um host descarta uma rota se o timer para esta expirar.

9.17 Resumo

O protocolo Internet de controle de mensagens (Message Control Protocol Internet) é uma parte necessária e integrante do IP que é usada para reportar erros e enviar informações de controle. Na maioria dos casos, mensagens ICMP de erro se originam em um roteador na Internet. Uma mensagem ICMP vai sempre voltar para a fonte original do datagrama que causou o erro.

O ICMP inclui mensagens de *destino inalcançável*, que informam quando um datagrama não pode ser encaminhado para o seu destino; mensagens de *pacote muito grande* (*packet too big*), que especificam que um datagrama não cabe na MTU de uma rede; mensagens de *redirecionar*

mensagens, que solicitam a um host que mude o primeiro salto em sua tabela de encaminhamento; mensagens de *tempo excedido*, que informam quando um limite de salto ou tempos de reconstituição expiram; mensagens de *problema de parâmetro*, para outros problemas de cabeçalho. Além disso, mensagens de *requisição/resposta de eco* (*echo request/reply menssages*) ICMP podem ser usadas para testar se um destino está acessível. Um conjunto de mensagens mais antigas ICMPv4 que foram destinadas a fornecer informações para inicialização de um host não é mais usado.

Uma mensagem ICMP viaja na área de dados de um datagrama IP e tem três campos de comprimento fixo no início da mensagem: um campo de mensagem ICMP *tipo*, um campo *código* e um campo ICMP *checksum*. O tipo de mensagem determina o formato do restante da mensagem, bem como o seu significado.

EXERCÍCIOS

9.1 Crie um experimento para registrar quantos de cada tipo de mensagem ICMP aparecem no seu host durante um dia.

9.2 Examinar a aplicação de *ping* no seu computador. Tente usar o *ping* com um endereço IPv4 de rede broadcast ou um endereço IPv6 *All Nodes*. Quantos computadores respondem? Leia os documentos de protocolo para determinar se é necessário responder a um pedido de transmissão broadcast, se é recomendado responder, se não é recomendado ou se é proibido.

9.3 Explique como um aplicativo *traceroute* pode usar ICMP.

9.4 Seria possível um roteador dar prioridade a mensagens ICMP sobre o tráfego normal? Sim ou não? Por quê?

9.5 Considere uma Ethernet que tem um host convencional, *H* e 12 roteadores conectados a ele. Encontre um único (ligeiramente ilegal) frame transportando um pacote IP que, quando enviado pelo host *H*, faz com que *H* receba exatamente 24 pacotes.

9,6 Não há nenhuma mensagem ICMP que permita que uma máquina informe a fonte cujos erros de transmissão estão fazendo datagramas

chegarem com checksum incorreto. Explique o porquê.

9.7 Na questão anterior, em que circunstâncias pode tal mensagem ser útil?

9.8 Mensagens ICMP de erro devem conter uma estampa de tempo que especifique quando foram enviadas? Sim ou não? Por quê?

9.9 Se os roteadores no seu site participam de solicitação de roteador ICMP, descubra quantos endereços cada roteador anuncia em cada interface.

9.10 Tente acessar um servidor em um host inexistente em sua rede local. Também tente se comunicar com um host inexistente em uma rede remota. Em que caso(s) você recebe uma mensagem ICMP de erro? Qual(is) mensagem(s) você recebe? Por quê?

---

* Quando nos referirmos à versão do ICMP que acompanha o IPv4, escreveremos ICMPv4, e quando nos referirmos à versão que acompanha o IPv6, escreveremos ICMPv6.

* ICMP somente retorna a parte do datagrama que causou o problema para evitar ter mensagem ICMP fragmentada.

** Ambos TCP e UDP armazenam o número das portas de protocolo nos primeiros 32 bits.

* Na prática, é possível calcular um checksum nos campos de pseudocabeçalho sem copiá-los.

** O IETF recomenda relatar somente mensagens de host inalcançável para a fonte original, e usar protocolos de roteamento para lidar com outros problemas de encaminhamento.

* O Capítulo 22 analisa o uso do DHCP assim como o dos protocolos IPv6 na inicialização.

CAPÍTULO 10

User Datagram Protocol (UDP)

CONTEÚDOS DO CAPÍTULO

10.1 Introdução

Os capítulos anteriores descrevem uma internet abstrata capaz de transferir datagramas IP entre computadores host, onde cada datagrama é encaminhado através da internet com base no endereço IP do destino. Na camada internet, um endereço de destino identifica um computador host; nenhuma outra distinção é feita em relação a qual usuário ou qual aplicativo no computador receberá o datagrama. Este capítulo estende a família de protocolo TCP/IP, adicionando um mecanismo que distingue entre os destinos dentro de um determinado host, permitindo que vários programas aplicativos em execução em um determinado computador enviem e recebam datagramas de forma independente.

10.2 Usando uma porta de protocolo como um destino final

Os sistemas operacionais na maioria dos computadores permitem executar simultaneamente múltiplas aplicações. Usando o jargão do sistema operacional, nos referimos a cada aplicativo em execução como um *processo*. Pode parecer natural dizer que uma aplicação é o destino final de uma mensagem. No entanto, especificar um determinado processo em uma máquina específica, como o destino final de um datagrama, é um pouco enganoso. Primeiro, porque um processo é criado quando se inicia um

aplicativo e destruído quando ele é encerrado, um remetente raramente tem conhecimento suficiente sobre qual processo em uma máquina remota está executando uma determinada aplicação. Segundo, gostaríamos de um esquema que permitisse que o TCP/IP fosse usado em um sistema operacional arbitrário, e que os mecanismos utilizados para identificar um processo variassem entre sistemas operacionais. Terceiro, reiniciar um computador pode mudar o processo associado a cada aplicação, mas não é necessário que os remetentes saibam sobre essas mudanças. Quarto, buscamos um mecanismo que possa identificar um serviço que o computador oferece, sem saber como o serviço é implementado (por exemplo, para permitir que um remetente entre em contato com um servidor web, sem saber qual processo na máquina de destino implementa a função de servidor).

Em vez de pensar em um aplicativo em execução como o destino final, vamos imaginar que cada máquina contém um conjunto de pontos de destino abstrato chamado *portas de protocolo*. Cada porta de protocolo é identificada por um número inteiro positivo. O sistema operacional local fornece um mecanismo de interface que analisa o uso para especificar uma porta ou acessá-la.

A maioria dos sistemas operacionais fornece acesso síncrono às portas. Do ponto de vista de um aplicativo, acesso síncrono significa que a computação para quando o aplicativo acessa a porta. Por exemplo, se um aplicativo tenta extrair dados de uma porta antes de qualquer dado chegar, o sistema operacional para temporariamente (bloqueia) a aplicação até que os dados cheguem. Uma vez que estes chegam, o sistema operacional transfere os dados para a aplicação e recomeça a execução. Em geral, as portas são armazenadoras (*buffered*) – se os dados chegam antes de um aplicativo estar pronto para aceitá-los, o software de protocolo irá armazená-los para que não sejam perdidos. Para alcançar o buffer, o protocolo de software localizado no interior do sistema operacional coloca os pacotes que chegam para uma porta de protocolo particular em uma fila (finita) até que a aplicação os extraia.

Para se comunicar com uma porta remota, um remetente precisa saber o endereço IP da máquina de destino e um número de porta de protocolo dentro dessa máquina. Cada mensagem possui dois números de porta de protocolo: um número de *porta de destino*, que especifica uma porta no computador de destino para a qual a mensagem foi enviada, e um número de *porta de origem*, que especifica uma porta na máquina emissora a partir

da qual a mensagem foi enviada. Como uma mensagem contém o número da porta do aplicativo de envio que usou, o aplicativo na máquina de destino tem informações suficientes para gerar uma resposta e encaminhá-la de volta para o remetente.

10.3 O User Datagram Protocol (UDP)

No conjunto de protocolos TCP/IP, o *User Datagram Protocol* (*UDP*) fornece o mecanismo primário que programas aplicativos usam para enviar datagramas para outros programas de aplicação. Mensagens UDP contêm números de porta de protocolo que são usados para distinguir vários aplicativos em execução em um único computador. Isto é, além dos dados enviados, em cada mensagem UDP há tanto um número de porta de destino como um número de porta de origem, o que torna possível para o software de UDP no destino entregar uma mensagem de entrada ao destinatário correto e para o receptor enviar uma resposta.

O UDP usa o protocolo Internet subjacente para transportar uma mensagem de uma máquina para outra. Surpreendentemente, ele fornece aplicativos com a mesma semântica de entrega de datagramas sem conexão por melhor esforço que o IP. Ou seja, o UDP não garante que as mensagens cheguem, nem que cheguem na mesma ordem em que foram enviadas, e também não fornece qualquer mecanismo para controlar a taxa com que a informação flui entre o par de hosts em comunicação. Então, as mensagens podem ser perdidas, duplicadas ou chegar fora de ordem. Além disso, pacotes podem chegar mais rapidamente do que o destinatário é capaz de processá-los. Podemos então fazer o resumo a seguir.

> *O User Datagram Protocol (UDP) fornece um serviço por melhor esforço não confiável de entrega sem conexão, usando IP para transportar mensagens entre máquinas. O UDP usa o IP para levar mensagens, mas adiciona a capacidade de distinguir entre os vários destinos dentro de um determinado computador host.*

Uma consequência importante surge da semântica UDP: um aplicativo que usa o UDP deve assumir total responsabilidade para lidar com os problemas de confiabilidade, incluindo perda de mensagens, duplicação, atraso, entrega fora de ordem e perda de conectividade. Infelizmente, desenvolvedores de aplicativos por vezes escolhem o UDP sem entender

essa responsabilidade. Além disso, como o software de rede é geralmente testado através de redes locais que têm alta confiabilidade, alta capacidade, baixo atraso e sem perda de pacotes, o teste pode não expor falhas potenciais. Assim, aplicativos que dependem de UDP, que funcionam bem em um ambiente local, podem falhar de forma dramática quando utilizados através da Internet global.

10.4 Formato de mensagem UDP

Usamos o termo *user datagram* para descrever uma mensagem UDP; a ênfase no *user* destina-se a distinguir datagramas UDP de datagramas IP. Conceitualmente, um user datagram consiste de duas partes: um cabeçalho que contém metainformação, tal como números de porta de protocolo de origem e de destino, e uma área de carga que contém os dados que estão sendo enviados. A Figura 10.1 ilustra a organização.

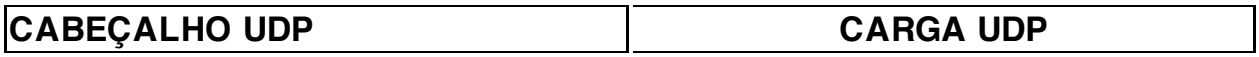


Figura 10.1 Organização conceitual de uma mensagem UDP.

O cabeçalho de um user datagram é extremamente pequeno: é composto por quatro campos que especificam a porta de protocolo a partir da qual a mensagem foi enviada, a porta de protocolo para a qual a mensagem é destinada, o tamanho da mensagem e um checksum UDP. Cada campo tem um tamanho de 16 bits, o que significa que todo o cabeçalho ocupa apenas um total de 8 octetos. A Figura 10.2 ilustra o formato do cabeçalho.

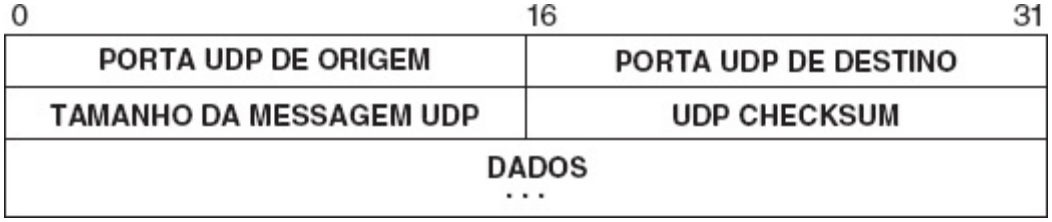


Figura 10.2 Formato dos campos em um datagrama UDP.

O campo PORTA DE ORIGEM UDP contém um número de porta de protocolo de 16 bits usado pelo aplicativo de envio, e o campo de PORTA DE DESTINO UDP contém o número da porta UDP de 16 bits do aplicativo de recebimento. Em essência, o software de protocolo usa os números de porta para demultiplexar datagramas entre os aplicativos que estão

esperando para recebê-los. Curiosamente, a PORTA DE ORIGEM UDP é opcional. Nós pensamos nela como identificadora da porta à qual a resposta deve ser enviada. Em uma transferência de mão única, onde o receptor não envia uma resposta, a porta de origem não é necessária e pode ser ajustada para zero.

O campo TAMANHO DA MENSAGEM UDP (UDP MESSAGE LENGTH) contém uma contagem de octetos no datagrama UDP, incluindo o cabeçalho UDP e os dados do usuário. Assim, o valor mínimo é oito, tamanho só do cabeçalho. O campo TAMANHO DA MENSAGEM UDP é composto por 16 bits, o que significa que o valor máximo que pode ser representado é 65.535. Como uma questão prática, no entanto, veremos que uma mensagem UDP deve caber na área de carga útil de um datagrama IP. Por conseguinte, o tamanho máximo permitido depende do tamanho do(s) cabeçalho(s) de IP, o qual é consideravelmente maior em um datagrama IPv6 do que num datagrama IPv4.

10.5 Interpretação do checksum UDP

IPv4 e IPv6 diferem na sua interpretação do campo UDP CHECKSUM. Para o IPv6, o checksum UDP é necessário. Para o IPv4, o checksum UDP é opcional e não precisa ser usado; um valor de zero no campo CHECKSUM significa que nenhum checksum foi calculado (isto é, um receptor não deve verificar o checksum). Os desenvolvedores IPv4 escolheram tornar o checksum opcional para permitir às implementações operarem com pouca sobrecarga computacional ao usar UDP através de uma rede local de alta confiabilidade. Lembre-se, no entanto, de que o IP não computa o checksum na parte de dados de um datagrama IP. Assim, o checksum UDP fornece a única maneira de garantir que os dados chegaram intactos e devem ser usados.*

Os iniciantes normalmente questionam o que acontece com as mensagens UDP para as quais o checksum calculado é zero. Um valor calculado igual a zero é possível porque o UDP usa o mesmo algoritmo de checksum do IP: ele divide os dados em quantidades de 16 bits e calcula o complemento da sua soma do complemento. De modo surpreendente, zero não é um problema porque a aritmética de complemento a um possui duas representações para zero: todos os bits definidos como zero ou todos os bits definidos como um. Quando o checksum calculado é zero, o UDP usa a representação com todos os bits definidos como um.

10.6 Computação checksum e pseudocabeçalho UDP

O checksum UDP abrange mais informações do que as que existem apenas no datagrama UDP. São extraídas informações do cabeçalho IP e o checksum abrange a informação extra assim como o cabeçalho UDP e a carga útil UDP. Assim como no ICMPv6, usamos o termo *pseudocabeçalho* para nos referirmos à informação extra. Podemos imaginar que o software checksum UDP extrai o campo pseudocabeçalho, o coloca na memória, anexa uma cópia da mensagem UDP e computa o checksum para o objeto inteiro.

É importante compreender que um pseudocabeçalho só é utilizado para o checksum. Não faz parte da mensagem UDP, não é colocado num pacote, e nunca é enviado através de uma rede. Para realçar a diferença entre um pseudocabeçalho e outros formatos de cabeçalho mostrados ao longo do texto, usamos as linhas tracejadas nas figuras que ilustram um pseudocabeçalho.

O objetivo de usar um pseudocabeçalho é verificar que um datagrama UDP chegou ao seu destino correto. A chave para entender o pseudocabeçalho encontra-se em perceber que destino correto consiste em uma máquina específica e uma porta de protocolo específico dentro dessa máquina. O cabeçalho UDP propriamente dito especifica somente o número da porta do protocolo. Desse modo, para verificar o destino, o UDP inclui o endereço de IP de destino no checksum, assim como o cabeçalho UDP. No destino final, o software UDP verifica o checksum usando o endereço IP de destino obtido a partir do cabeçalho do datagrama IP que levou a mensagem UDP. Se os checksums combinam, então deve ser verdade que o datagrama chegou ao host de destino pretendido, bem como à porta de protocolo correta dentro desse host.

10.7 Formato do pseudocabeçalho UDP IPv4

O pseudocabeçalho usado na computação do checksum UDP para IPv4 consiste de 12 octetos de dados organizados como a Figura 10.3 ilustra.

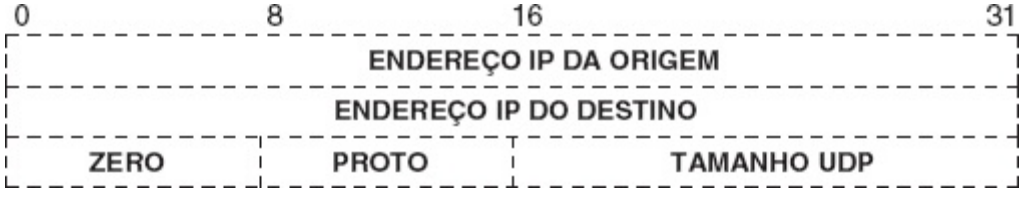


Figura 10.3 Os 12 octetos de um pseudocabeçalho IPv4 usado durante computação checksum UDP.

Os campos do pseudocabeçalho rotulado ENDEREÇO IP DA ORIGEM e ENDEREÇO IP DE DESTINO contêm os endereços IPv4 de origem e de destino que serão colocados em um datagrama IPv4 ao se enviar a mensagem UDP. O campo *PROTO* contém o código de tipo de protocolo IPv4 (*17* para UDP), e o campo rotulado TAMANHO UDP contém o tamanho do datagrama UDP (não incluído o pseudocabeçalho). Para verificar o checksum, o destino deve extrair esses campos do cabeçalho IPv4, colocá-los em um formato pseudocabeçalho e computar o checksum.*

10.8 Formato do pseudocabeçalho UDP IPv6

O pseudocabeçalho usado na computação do checksum UDP para IPv6 consiste de 40 octetos de dados organizados como a Figura 10.4 ilustra.

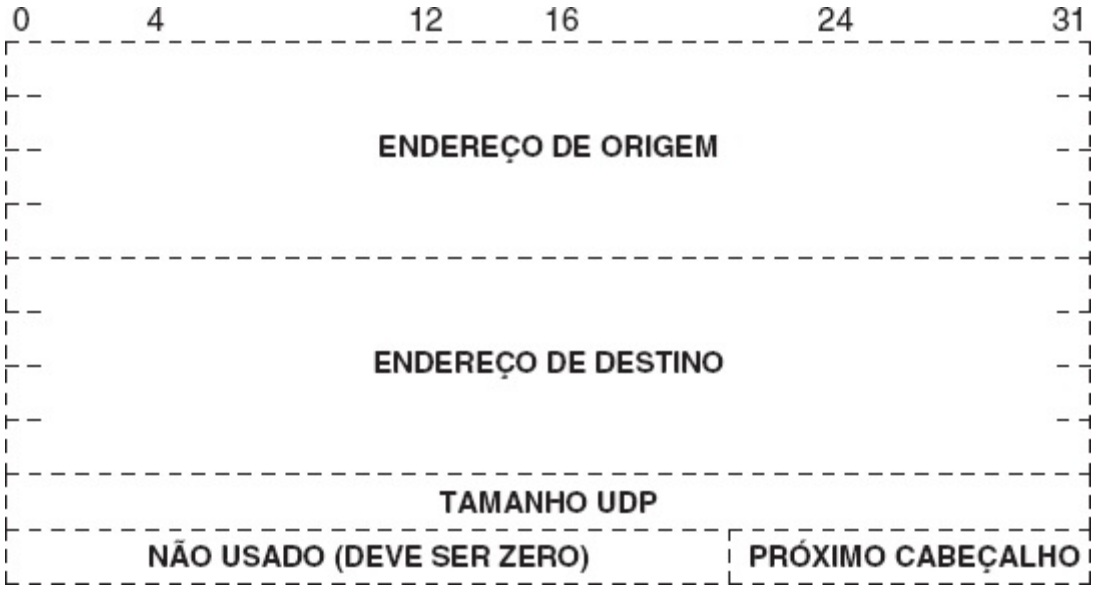


Figura 10.4 Os 40 octetos de um pseudocabeçalho IPv6 usado durante computação checksum UDP.

Claro que o pseudocabeçalho para IPv6 usa os endereços de origem e destino IPv6. As outras alterações do IPv4 são que o campo PROTO é substituído pelo campo PRÓXIMO CABEÇALHO e a ordem dos campos mudou.

10.9 Encapsulamento e camadas de protocolos UDP

UDP fornece o nosso primeiro exemplo de um protocolo de transporte. No modelo de referência TCP/IP 5 camadas, no Capítulo 4, UDP reside na

camada de transporte acima da camada de internet. Conceitualmente, aplicativos acessam UDP, que usa IP para enviar e receber datagramas. A Figura 10.5 ilustra as camadas conceituais.

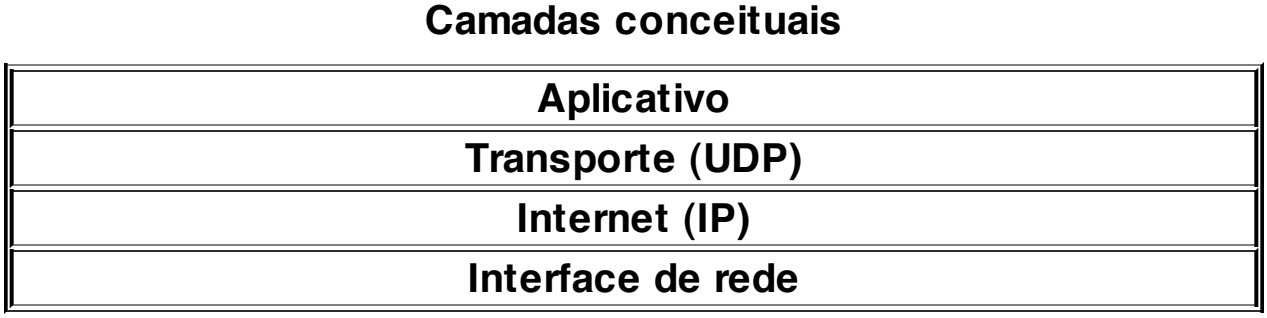


Figura 10.5 As camadas conceituais de UDP entre programas de aplicações e programas IP.

No caso do UDP, as camadas conceituais na figura também implicam encapsulamento. Isto é, porque o UDP está em camadas acima do IP, uma mensagem UDP completa, incluindo o cabeçalho UDP e carga útil, é encapsulada em um datagrama IP, enquanto atravessa uma internet. Claro que o datagrama é encapsulado em um frame de rede enquanto se desloca através de uma rede subjacente, o que significa que existem dois níveis de encapsulamento. A Figura 10.6 ilustra o encapsulamento.

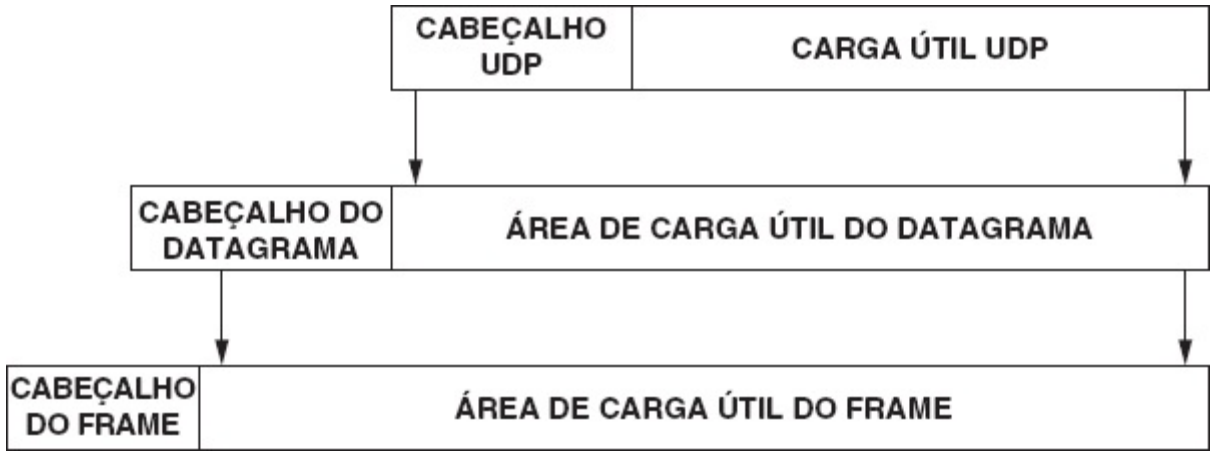


Figura 10.6 Dois níveis de encapsulamento usados quando uma mensagem UDP viaja em um datagrama IP, que viaja em um frame de rede.

Como a figura indica, o encapsulamento irá resultar numa sequência linear de cabeçalhos. Portanto, se um capturar um frame que contenha o UDP, o frame começará com um cabeçalho de frame, seguido de um

cabeçalho de IP seguido por um cabeçalho de UDP. Em termos de construção de um pacote de saída, podemos imaginar que uma aplicação especifica os dados a serem enviados. O UDP adiciona seu cabeçalho aos dados e passa o datagrama UDP para o IP. A camada IP adiciona um cabeçalho IP ao que recebeu do UDP. Finalmente, a camada de interface de rede incorpora o datagrama IP em um frame antes de enviá-lo de uma máquina para outra. O formato do frame depende da tecnologia de rede subjacente, mas, na maior parte das tecnologias, um frame inclui um cabeçalho adicional. A questão é que, quando se olha para um frame, o cabeçalho mais externo corresponde à camada de protocolo mais baixa, enquanto o cabeçalho mais interno corresponde à camada de protocolo mais alta.

Na entrada, um pacote chega quando um dispositivo de driver na camada de interface de rede recebe um pacote a partir do dispositivo de interface de rede e coloca o pacote na memória. O processamento começa uma subida através de camadas sucessivamente mais elevadas de software de protocolo. Conceitualmente, cada camada remove um cabeçalho antes de passar a mensagem para a camada superior seguinte. No momento em que a camada de transporte passa os dados para o processo de destino, todos os cabeçalhos foram removidos. Ao considerar como cabeçalhos são inseridos e removidos, é importante ter em mente o princípio de camadas. Em particular, observa-se que o princípio de camadas aplica-se ao UDP, o que significa que o datagrama UDP recebido do IP na máquina de destino é idêntico ao datagrama que o UDP passou ao IP na máquina de origem. Além disso, os dados que o UDP fornece a um aplicativo na máquina receptora serão exatamente os dados que um aplicativo passou para o UDP na máquina emissora.

A divisão de tarefas entre várias camadas de protocolo é rígida e clara conforme descrito a seguir.

> *A camada IP é responsável só pela transferência de dados entre um par de hosts em uma internet, enquanto a camada UDP é responsável só pela diferenciação entre múltiplas fontes ou destinos dentro de um host.*

Então, só o cabeçalho IP identifica os hosts de origem e destino; só a camada UDP identifica as portas de origem e destino dentro de um host.

10.10 Camadas e cálculo de checksum UDP

Os leitores atentos vão notar uma aparente contradição entre as regras de camadas e o cálculo do checksum UDP. Lembre-se de que o checksum UDP inclui um pseudocabeçalho que possui campos para os endereços IP de origem e destino. Pode-se argumentar que o endereço IP de destino precisa ser conhecido pelo usuário no envio de um datagrama UDP, e o usuário precisa passá-lo para a camada UDP. Assim, a camada UDP pode obter o endereço IP de destino sem interagir com o cabeçalho IP. Porém, o endereço IP de origem depende da rota que o IP escolhe para o datagrama, pois o endereço IP de origem identifica a interface de rede sobre a qual o datagrama é transmitido. Assim, a menos que interaja com a camada IP, o UDP não pode saber o endereço de origem IP.

Consideramos que o UDP pede à camada IP para calcular os endereços IP de origem e (possivelmente) destino, usa-os para construir um pseudocabeçalho, calcula o checksum, descarta o pseudocabeçalho e depois passa o datagrama UDP ao IP para transmissão. Uma técnica alternativa que produz maior eficiência faz com que a camada UDP encapsule o datagrama UDP em um datagrama IP, obtenha o endereço de origem a partir do IP, armazene os endereços de origem e destino nos campos apropriados do cabeçalho do datagrama, calcule o checksum UDP e depois passe o datagrama IP à camada IP, que só precisa preencher os campos restantes do cabeçalho IP.

A forte interação entre UDP e IP viola nossa premissa básica de que as camadas refletem a separação da funcionalidade? Sim. O UDP tem sido bastante integrado ao protocolo IP. Ele é claramente um compromisso de separação pura, feita por motivos totalmente práticos. Desejamos passar por cima da violação de camadas porque é impossível identificar totalmente um programa aplicativo de destino sem especificar a máquina de destino, e queremos tornar eficiente o mapeamento entre os endereços usados pelo UDP e aqueles usados pelo IP. Um dos exercícios examina essa questão de um ponto de vista diferente, pedindo ao leitor para considerar se o UDP deve ser separado do IP.

10.11 Multiplexação, demultiplexação e portas protocolo

Vimos, no Capítulo 10, que o software das camadas de uma hierarquia de protocolos precisa multiplexar ou demultiplexar entre vários objetos na próxima camada. O software UDP oferece outro exemplo de multiplexação e demultiplexação.

- A multiplexação ocorre na saída. Em um determinado computador

host, vários aplicativos podem usar o UDP simultaneamente. Assim, podemos imaginar o software UDP aceitando mensagens enviadas a partir de um conjunto de aplicações, colocando cada uma em um datagrama UDP e passando os datagramas para o IP para transmissão.

- A demultiplexação ocorre na entrada. Podemos imaginar o UDP aceitando datagramas UDP recebidos do IP, escolhendo o aplicativo para o qual o datagrama foi enviado e passando os dados para o aplicativo.

Conceitualmente, toda a multiplexação e demultiplexação entre o software UDP e os programas aplicativos ocorre por meio do mecanismo de porta. Na prática, cada programa aplicativo precisa negociar com o sistema operacional para obter uma porta de protocolo local criar os recursos necessários para enviar e receber mensagens UDP.* Uma vez que o sistema tenha criado os recursos necessários, o aplicativo pode enviar dados; um código UDP no sistema operacional criará um datagrama UDP de saída e colocará o número da porta local no campo PORTA DE ORIGEM UDP.

Conceitualmente, é necessário apenas o número da porta de destino para lidar com demultiplexação. Quando se processa um datagrama de entrada, o UDP aceita o datagrama do software IP, extrai a PORTA DE DESTINO UDP a partir do cabeçalho e passa os dados para a aplicação. A Figura 10.7 ilustra a demultiplexação.

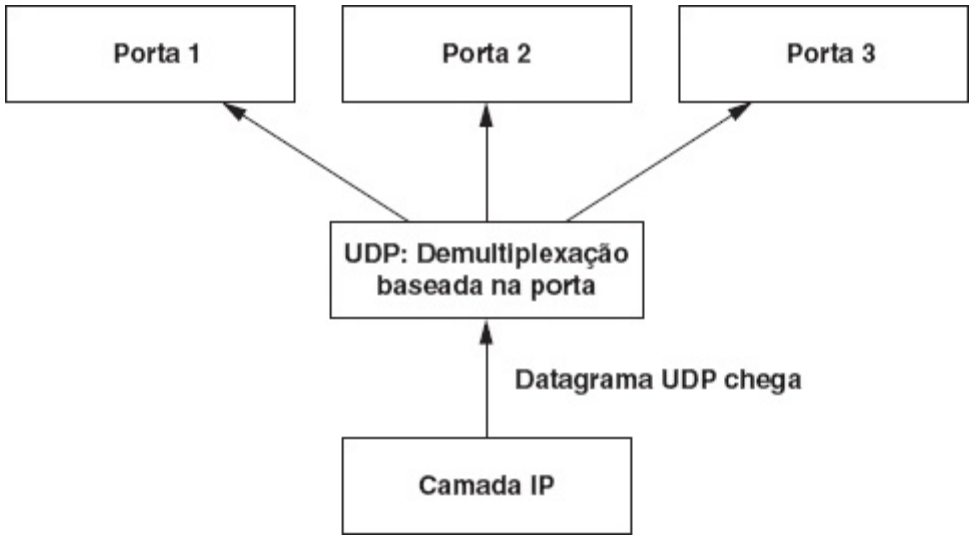


Figura 10.7 Visão conceitual da demultiplexação UDP de datagramas de entrada.

A maneira mais fácil de pensar em uma porta UDP é como uma fila de datagramas de entrada. Na maioria das implementações, quando um

aplicativo negocia com o sistema operacional para atribuir uma porta, o sistema operacional cria uma fila interna necessária para manter os datagramas que chegam. O aplicativo pode especificar ou alterar o tamanho da fila. Quando o UDP recebe um datagrama, ele verifica se o número da porta de destino corresponde a uma das portas em uso.** Se encontrar uma correspondência, o UDP enfileira o novo datagrama na porta onde o programa aplicativo pode acessá-lo. Se nenhuma das portas alocadas corresponder ao datagrama recebido, o UDP envia uma mensagem ICMP para informar à fonte que a porta estava inacessível e descarta o datagrama. Claro, um erro também ocorre se a porta estiver cheia. Em tais casos, o UDP descarta o datagrama e envia uma mensagem ICMP.

10.12 Números de porta UDP reservados e disponíveis

Como os números de porta de protocolo devem ser atribuídos? O problema é importante porque dois computadores precisam combinar a respeito dos números de porta antes que possam interoperar. Por exemplo, quando um usuário no computador *A* decide fazer uma chamada telefônica VoIP para um usuário no computador *B*, o software do aplicativo precisa saber qual porta de protocolo o aplicativo no computador *B* está usando. Existem duas técnicas fundamentais para a atribuição de porta. A primeira técnica usa uma autoridade central. Todos concordam em permitir que uma autoridade central atribua números de porta conforme a necessidade e publique a lista de todas as atribuições. Depois, todo o software é criado de acordo com a lista. Essa técnica às vezes é chamada de *atribuição universal*, e as *atribuições de porta especificadas* pela autoridade são chamadas atribuições de porta bem conhecidas (*well-known port assignments*).

A segunda técnica para atribuição de porta utiliza o vínculo dinâmico, segundo o qual as portas não são conhecidas globalmente. Em vez disso, sempre que um programa precisa de um número de porta de protocolo, o software de protocolo no sistema operacional escolhe um número não usual e o atribui ao aplicativo. Quando um aplicativo precisa saber a porta de protocolo atribuída que está em uso em outro computador, deve enviar uma requisição que pergunte qual a atribuição da porta (por exemplo, "Que porta o serviço de telefonia VoIP está usando?"). A máquina de destino responde dando o número de porta a ser usado.

Os desenvolvedores do TCP/IP adotaram uma técnica híbrida que atribui alguns números de porta *a priori*, mas deixa muitos disponíveis para que

os sites locais ou os programas aplicativos atribuam dinamicamente. Os números de porta bem conhecidos atribuídos pela autoridade central começam com valores baixos e se estendem para cima, deixando grandes quantidades de valores inteiros disponíveis para atribuição dinâmica. A tabela na Figura 10.8 lista exemplos de números bem conhecidos de portas de protocolo UDP.

Porta	Palavra-chave	Descrição
0	-	Reserved
7	echo	Eco
9	discard	Descarte
11	systat	Usuários ativos
13	daytime	Hora do dia
15	netstat	Programa de status da rede
17	qotd	Citação do dia
19	chargen	Gerador de caracteres
37	time	Hora
42	name	Servidor de nome de host
43	whois	Quem é
53	nameserver	Domain Name Server
67	bootps	Servidor BOOTP or DHCP
68	bootpc	Cliente BOOTP or DHCP
69	tftp	Trivial File Transfer
88	kerberos	Serviço de segurança Kerberos
111	sunrpc	ONC Remote Procedure Call (Sun RPC)
123	ntp	Network Time Protocol
161	snmp	Simple Network Management Protocol
162	snmp-trap	Traps SNMP
264	bgmp	Border Gateway Multicast Protocol (BGMP)
389	idap	Lightweight Directory Access Protocol (LDAP)
512	biff	Comsat do UNIX
514	syslog	Log do System
520	rip	Routing Information Protocol (RIP)
525	timed	Daemon de hora
546	dhcpv6-c	Cliente DHCPv6
547	dhcpv6-s	Servidor DHCPv6
944	nsf	Service Network File System (NFS)

Figura 10.8 Exemplos de números de porta de protocolo UDP bem conhecidos.

10.13 Resumo

Modernos sistemas operacionais permitem que vários programas aplicativos sejam executados simultaneamente. O User Datagram Protocol (UDP) distingue vários processos dentro de determinada máquina permitindo que os emissores e receptores atribuam um número de porta de protocolo de 16 bits a cada aplicativo. Uma mensagem UDP inclui dois números de porta de protocolo que identificam uma aplicação no computador remetente e um aplicativo no computador de destino. Alguns números de porta UDP são *bem conhecidos*, sendo atribuídos permanentemente por uma autoridade central e honrados na Internet. Outros números de porta estão disponíveis para quaisquer programas aplicativos utilizarem.

O UDP é um protocolo "enxuto" pelo fato de que não aumenta muita coisa à semântica do IP. Ele simplesmente oferece aos programas aplicativos a capacidade de se comunicar usando o serviço de remessa de pacote em conexão e não confiável do IP. Assim, as mensagens UDP podem ser perdidas, duplicadas, adiadas ou entregues fora da ordem; um par de programas aplicativos que utiliza UDP deve estar preparado para lidar com erros. Se um aplicativo UDP não resolve os erros, ele pode funcionar corretamente em uma LAN confiável, mas não em uma internet de longa distância na qual problemas de atraso e perda são mais comuns.

No esquema de camadas de protocolo, o UDP se encontra na camada de transporte, acima da camada 3, a camada de internet, e abaixo da camada 5, a camada de aplicação. Conceitualmente, a camada de transporte é independente da camada de internet, mas na prática elas interagem fortemente. O checksum UDP inclui os pseudoendereços com os endereços IP da origem e do destino, significando que o software UDP precisa interagir com o software IP para encontrar endereços IP antes de enviar datagramas.

EXERCÍCIOS

10.1 Construa dois programas que usam UDP e meça a velocidade média de transferência com mensagens de 256, 512, 1024, 2048, 4096 e 8192

octetos. Você pode explicar os resultados? (Dica: o que é o MTU da rede que você está usando?)

10.2 Por que o checksum UDP é separado do checksum IP? Você teria objeção a um protocolo que usasse um único checksum para o datagrama IP completo, incluindo a mensagem UDP?

10.3 Não usar checksums pode ser perigoso. Explique como um único broadcast de pacote ARP pela máquina *P* pode tornar impossível alcançar outra máquina, *Q*.

10.4 A noção de protocolos múltiplos, identificada pelas portas de protocolo, deveria ter sido embutida no IP? Sim ou não? Por quê?

10.5 Qual é a principal vantagem de usar números de porta UDP pré-atribuídos? E a principal desvantagem?

10.6 Qual é a principal vantagem do uso de portas de protocolo em vez de identificadores de processo para especificar o destino dentro de uma máquina?

10.7 O UDP fornece comunicação de datagramas não confiável, pois não garante a entrega da mensagem. Elabore um protocolo de datagramas confiável que usa os tempos de espera e reconhecimento para garantir a entrega. Quanto de sobrecarga de rede e atraso a confiabilidade introduz?

10.8 *Registro de nome*. Suponha que você queira permitir que quaisquer pares de programas aplicativos estabeleçam comunicação com o UDP, mas não queira atribuir-lhes números de porta UDP fixos. Em vez disso, você gostaria que correspondentes em potencial fossem identificados por uma string de 64 caracteres ou menos caracteres. Assim, um programa na máquina *A* poderia querer se comunicar com o programa "id-longo-especial" na máquina *B*. Nesse meio-tempo, suponha que um aplicativo na máquina *C* deseja se comunicar com um aplicativo na máquina *A* que escolheu um ID "meu-próprio-id-privado". Mostre que você só precisa atribuir uma porta UDP para tornar essa comunicação possível, projetando o software em cada máquina de modo a permitir que (a) um aplicativo local selecione um número de porta UDP não usada, pela qual se comunicarão, (b) um aplicativo local registre o nome de 64 caracteres ao qual ele responde, e (c) um aplicativo remoto use o UDP para estabelecer comunicação usando apenas o nome de 64 caracteres e o endereço internet de

destino.

10.9 Implemente o software de registro de nome do exercício anterior.

10.10 Envie datagramas UDP através de uma rede de longa distância e meça o percentual perdido e o percentual reordenado. Será que o resultado depende da hora do dia ou da carga da rede?

10.11 A norma define a porta UDP 7 como uma porta de eco – um datagrama enviado para a porta de eco é simplesmente enviado de volta para o remetente. O que um serviço de eco UDP pode dizer a um gerente que um serviço de eco ICMP não pode?

10.12 Considere um projeto de protocolo em que UDP e IPv4 são mesclados e um endereço consiste em 48 bits, que inclui um endereço IPv4 de 32 bits convencional e um número de porta de 16 bits. Qual é a principal desvantagem de um esquema assim?

---

* O autor uma vez teve um problema: um arquivo copiado através de uma Ethernet foi corrompido porque o NIC Ethernet falhou e o aplicativo (NFS) usou UDP sem checksums.

* Na prática, é possível construir o software checksum que executa o cálculo correto sem copiar os campos em um pseudocabeçalho.

* Por agora, vamos descrever os mecanismos abstratamente; o Capítulo 21 fornece um exemplo das primitivas socket que muitos sistemas operacionais usam para criar e usar as portas.

** Na prática, a demultiplexação UDP permite que uma aplicação especifique correspondência em uma porta de origem, bem como em uma porta de destino.

CAPÍTULO 11

Serviço de transporte de fluxo confiável (TCP)

CONTEÚDOS DO CAPÍTULO

11.1 Introdução

Os capítulos anteriores exploram o serviço de entrega de pacote sem conexão não confiável, que forma a base para toda a comunicação da Internet, e o protocolo IP, que a define. Este capítulo apresenta o segundo serviço em nível de rede mais importante e bem conhecido, entrega de fluxo confiável, e o *Transmission Control Protocol* (*TCP*), que o define. Veremos que o TCP acrescenta funcionalidade substancial aos protocolos já discutidos, mas que sua implementação também é substancialmente mais

complexa.

11.2 A necessidade de serviço confiável

No nível mais baixo, as redes de comunicação por computador proveem entrega de pacote não confiável. Os pacotes podem ser perdidos quando os erros de transmissão interferem nos dados ou quando o hardware de rede falha e atrasam quando a rede fica muito carregada. Os sistemas de comutação de pacotes mudam de rotas dinamicamente, o que quer dizer que podem entregar pacotes fora de ordem, entregá-los após um atraso substancial ou entregar duplicatas.

No nível mais alto, os programas aplicativos normalmente precisam enviar grandes volumes de dados de um computador para outro. O uso de um sistema de remessa não confiável sem conexão para grandes transferências de volume se torna tedioso e incômodo; desenvolvedores devem incorporar detecção e recuperação de erro em cada programa aplicativo. Por ser difícil projetar, entender e implementar um software que ofereça confiabilidade corretamente, pesquisadores de rede têm trabalhado para criar uma solução de uso geral – um protocolo único de transferência confiável que todos os aplicativos possam usar. Isso significa que programadores de aplicativos não precisarão incorporar um protocolo de transferência confiável em cada aplicativo.

11.3 Propriedades do serviço de entrega confiável

O serviço de transferência confiável que o TCP fornece aos aplicativos pode ser caracterizado pelas cinco características que são discutidas a seguir:

- orientação de fluxo;
- conexão de circuito virtual;
- transferência em buffer;
- fluxo não estruturado;
- conexão Full Duplex.

Orientação de fluxo. Quando dois programas aplicativos transferem grandes volumes de dados, estes são vistos como um *fluxo de octetos.* O aplicativo na máquina de destino recebe exatamente a mesma sequência de octetos que o emissor passa para ele no host de origem.

Conexão de circuito virtual. Antes de iniciar a transferência de dados, tanto o aplicativo de envio como o de recebimento devem concordar em

estabelecer uma *conexão TCP*. Uma aplicação contata a outra para iniciar uma conexão. Os softwares TCP nos dois hosts se comunicam por meio do envio de mensagens através da internet subjacente. Eles verificam que a transferência é autorizada e ambos os lados estão prontos. Uma vez que todos os detalhes foram resolvidos, os módulos de protocolo informam aos programas de aplicação em cada extremidade que uma conexão foi estabelecida e que a transferência pode começar. O TCP controla a transferência de dados; se a comunicação falhar por algum motivo (por exemplo, porque o hardware da rede ao longo do caminho falhou), os programas aplicativos serão informados. Usamos o termo *circuito virtual* para descrever uma conexão TCP, que atua como um circuito de hardware dedicado, apesar de toda a comunicação ser feita com pacotes.

Transferência em buffer. Os programas aplicativos enviam um fluxo de dados pelo circuito virtual repetidamente, passando octetos de dados ao software de protocolo. Ao transferir dados, uma aplicação usa qualquer tamanho que achar conveniente, que pode ser a partir de um único octeto. O TCP coloca dados em pacotes que envia para o destino. No host receptor, o TCP garante que os dados estejam colocados na ordem original de forma que o aplicativo receba octetos exatamente na mesma ordem que foram enviados.

O TCP é livre para dividir o fluxo em pacotes independentes das partes que o programa aplicativo transfere. Para tornar a transferência mais eficiente e minimizar o tráfego da rede, as implementações normalmente coletam dados suficientes de um fluxo a fim de preencher um datagrama razoavelmente grande antes de transmiti-lo por uma internet. Assim, mesmo que o programa aplicativo gere o fluxo de um octeto de cada vez, a transferência por uma internet pode ser muito eficiente. De modo semelhante, se o programa aplicativo escolher gerar blocos de dados extremamente grandes, o software do protocolo pode escolher dividir cada bloco em partes de modo que cada uma caiba em um único pacote.

Para as aplicações em que os dados devem ser transferidos sem esperar preencher um buffer, o serviço de fluxo oferece um mecanismo *push* que as aplicações usam para forçar a transferência imediata. No lado emissor, um push força o software do protocolo a transferir todos os dados que foram gerados sem esperar preencher um buffer. Ao alcançar o lado receptor, o push faz com que o TCP torne os dados disponíveis à aplicação sem atraso. A função push só garante que todos os dados serão transferidos; ele não oferece limites. Assim, mesmo quando a entrega é forçada, o software do

protocolo pode decidir dividir o fluxo de maneiras inesperadas, ou se um aplicativo de recebimento for lento, os dados de vários pacotes push podem ser entregues ao aplicativo todos de uma vez.

Fluxo não estruturado. O TCP/IP não fornece fluxos de dados estruturados. Por exemplo, não existe um meio de uma aplicação de folha de pagamento identificar o conteúdo do fluxo como sendo dados de folha de pagamento ou marcar os limites entre registros de funcionários. Os programas aplicativos que utilizam o serviço de fluxo precisam entender o conteúdo de fluxo e combinar com um formato do fluxo antes de iniciarem uma conexão.

Conexão full duplex. As conexões fornecidas pelo serviço de fluxo TCP/IP permitem a transferência simultânea nas duas direções. Essas conexões são chamadas de *full duplex*. Conceitualmente, uma conexão full duplex consiste em dois fluxos independentes, indo em direções opostas; do ponto de vista de um processo de aplicação, não há qualquer interação aparente entre os dois. O TCP permite que um processo de aplicação termine o fluxo em uma direção enquanto os dados continuam a fluir na outra direção, tornando a conexão *half duplex*. A vantagem de uma conexão full duplex é que o software do protocolo subjacente pode enviar de volta para a origem, em datagramas transportando dados na direção oposta, informações de controle para um fluxo. Esse *piggybacking* reduz o tráfego da rede.

11.4 Confiabilidade: confirmação e retransmissão

Dissemos que o serviço de entrega de fluxo confiável garante a entrega de um fluxo de dados enviados de uma máquina para outra sem duplicação ou perda de dados. A questão é: "como o protocolo pode oferecer transferência confiável se o sistema de comunicação subjacente oferece apenas a remessa de pacotes não confiável?". A resposta é complicada, mas a maioria dos protocolos confiáveis usa uma única técnica fundamental, conhecida como *confirmação positiva com retransmissão* (*Positive Acknowledgement with Retransmition* – PAR). A técnica requer que um destinatário se comunique com a origem, enviando de volta uma mensagem de *confirmação* (ACK) cada vez que os dados chegam com sucesso. Quando envia um pacote, o software de envio inicia um timer. Se uma confirmação chega antes de o timer expirar, a origem cancela o timer e se prepara para

mandar mais dados. Se o timer expira antes da confirmação chegar, o emissor *retransmite* o pacote.

Antes de podermos entender o mecanismo de retransmissão TCP, devemos considerar alguns princípios básicos. O esquema de retransmissão mais simples possível espera que um determinado pacote seja reconhecido antes de enviar o próximo. Conhecido como *send-and-wait*, a abordagem só pode enviar um pacote de cada vez. A Figura 11.1 ilustra a troca de pacotes ao usar send-and-wait (enviar e esperar).

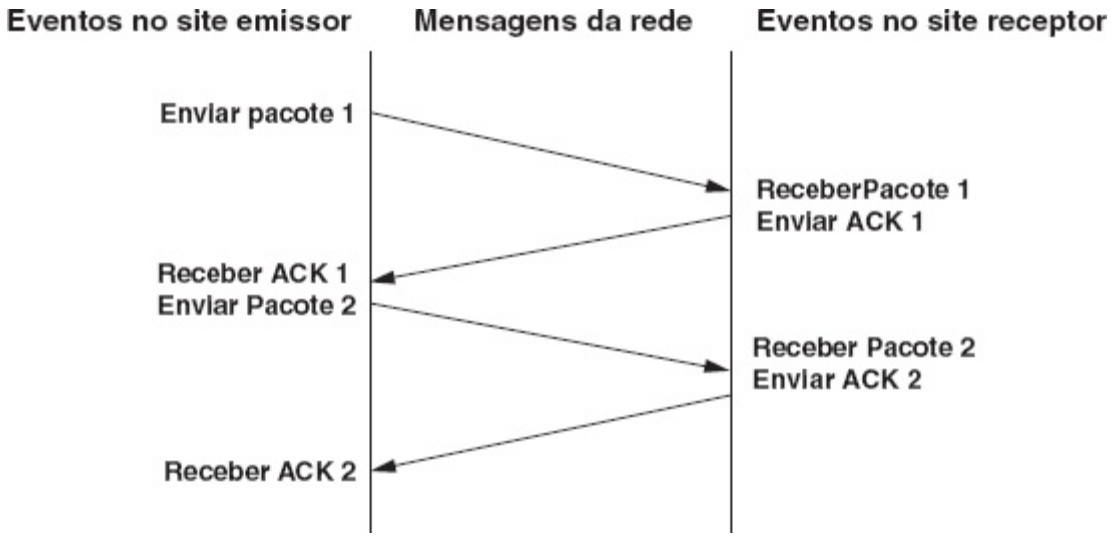


Figura 11.1 A troca de pacote para um protocolo básico send-and-wait. O tempo segue o sentido descendente na figura.

O lado esquerdo da figura lista os eventos no host de envio, e o lado direito lista os eventos no host receptor. Cada linha diagonal cruzando o meio mostra a transferência de um pacote ou um ACK.

A Figura 11.2 utiliza o mesmo formato da Figura 11.1 para mostrar o que acontece quando um pacote é perdido. O remetente envia um pacote e inicia um timer. O pacote é perdido, o que significa que nenhum ACK vai chegar. Quando o timer expira, o remetente retransmite o pacote perdido e deve iniciar um temporizador após a retransmissão para o caso de a segunda cópia também se perder. Na figura, a segunda cópia chega intacta, o que significa que o receptor envia um ACK. Quando o ACK chega ao lado de envio, o remetente cancela o timer.

Em nossa descrição anterior, o remetente deve manter uma cópia de um pacote que foi transmitido para o caso de o pacote precisar ser retransmitido. Na prática, o remetente só precisa manter os dados que vão

no pacote, juntamente com informação suficiente para permitir que o remetente reconstrua os cabeçalhos dos pacotes. A ideia de manter os dados *não reconhecidos* é importante em TCP.

Embora lide com a perda de pacotes ou atraso excessivo, o mecanismo de reconhecimento descrito anteriormente não resolve todos os problemas: um pacote pode ser duplicado. Duplicatas podem surgir se um atraso excessivo fizer a origem retransmitir desnecessariamente. Resolver duplicação requer uma reflexão cuidadosa, porque ambos, os pacotes e as confirmações, podem ser duplicados. Normalmente, os protocolos confiáveis detectam pacotes duplicados, atribuindo a cada um deles um número sequencial e exigindo que o receptor lembre qual sequência de números recebeu. Para evitar ambiguidade, os protocolos de confirmação positiva fazem com que cada confirmação contenha o número sequencial do pacote que chegou. Assim, quando uma confirmação chega, ela pode ser facilmente associada a um determinado pacote.

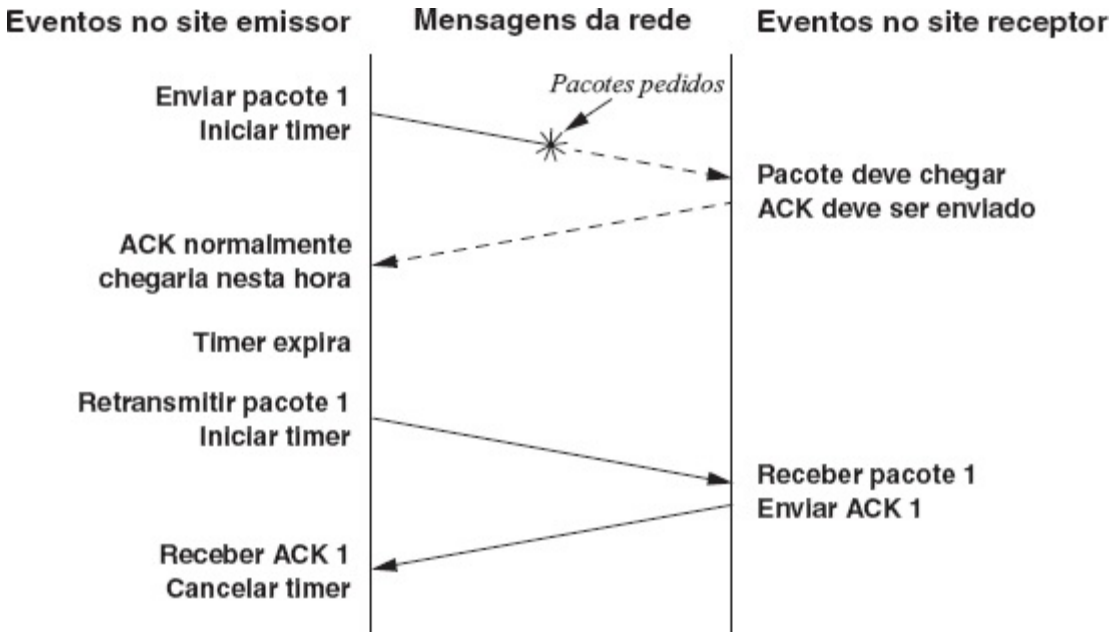


Figura 11.2 Ilustração de timeout e retransmissão quando um pacote é perdido.

11.5 O paradigma das janelas deslizantes

Antes de examinar o serviço de fluxo TCP, temos de explorar um mecanismo adicional que é a base da transmissão confiável. Conhecida como uma *janela deslizante* (*sliding window*), o mecanismo melhora o rendimento geral. Para entender a motivação para janelas deslizantes,

lembre-se da sequência de eventos na Figura 11.1. Para alcançar confiabilidade, o remetente transmite um pacote e, em seguida, espera por uma confirmação antes de transmitir outro. Como a figura mostra, os pacotes de dados fluem entre as máquinas um de cada vez. A rede permanecerá completamente inativa até que a confirmação retorne. Se imaginarmos uma rede com altos atrasos de transmissão, o problema torna-se claro como se explica a seguir.

> *Um protocolo de reconhecimento positivo simples desperdiça uma quantidade substancial de capacidade de rede porque atrasa o envio de um novo pacote até receber uma confirmação para o pacote anterior.*

A técnica de janela deslizante utiliza uma forma mais complexa de confirmação positiva e retransmissão. A ideia fundamental é a de que uma janela deslizante permite a um remetente transmitir vários pacotes antes de esperar por uma confirmação. A maneira mais fácil de visualizar uma janela deslizante em ação é pensar em uma sequência de pacotes a serem transmitidos, como mostra a Figura 11.3. O protocolo coloca uma pequena *janela* de tamanho fixo na sequência e transmite todos os pacotes que se encontram no interior dela.

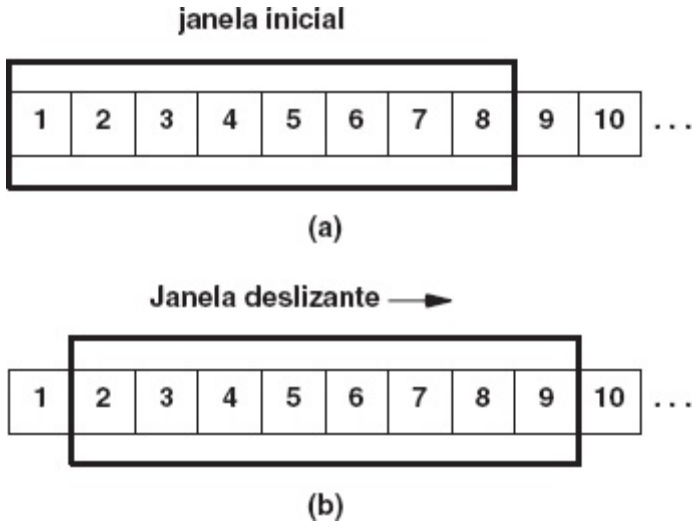


Figura 11.3 (a) Uma janela deslizante com oito pacotes na janela e (b) a janela deslizando para que o pacote *9* possa ser enviado, pois uma confirmação foi recebida para o pacote *1*

Dizemos que um pacote é *não confirmado* se tiver sido transmitido, mas nenhuma confirmação tiver sido recebida. Tecnicamente, o número de pacotes que podem ser não confirmados em qualquer momento é limitado pelo *tamanho da janela*, que é limitado a um pequeno número fixo. Por exemplo, num protocolo de janela deslizante com o tamanho de janela 8, é permitido que o remetente transmita 8 pacotes antes de receber uma confirmação.

Como a Figura 11.3 mostra, uma vez que o remetente recebe uma confirmação para o primeiro pacote no interior da janela, ele "desliza" a janela para frente e envia o pacote seguinte. A janela desliza para frente a cada vez que um reconhecimento chega.

O desempenho do deslizamento de protocolos da janela depende do tamanho desta e da velocidade com que a rede aceita pacotes. A Figura 11.4 ilustra um exemplo de operação de um protocolo de janela deslizante para um tamanho de janela de três pacotes. Note que o remetente transmite todos os três pacotes antes de receber quaisquer confirmações.

Com um tamanho de janela igual a *1*, um protocolo de janela deslizante é exatamente o mesmo que o nosso protocolo de reconhecimento positivo simples. Aumentando o tamanho da janela, é possível eliminar completamente o tempo ocioso da rede. Ou seja, no estado estacionário, o emissor pode transmitir pacotes tão rápido quanto seja possível para a rede transferi-los. Para ver a vantagem da janela deslizante, compare a taxa com que os dados são transferidos nas Figuras 11.1 e 11.4. O ponto principal é descrito a seguir.

Como um protocolo de janela deslizante bem afinado (well-tuned) mantém a rede completamente saturada com pacotes, ele pode obter rendimento substancialmente maior do que um protocolo de confirmação positiva simples.

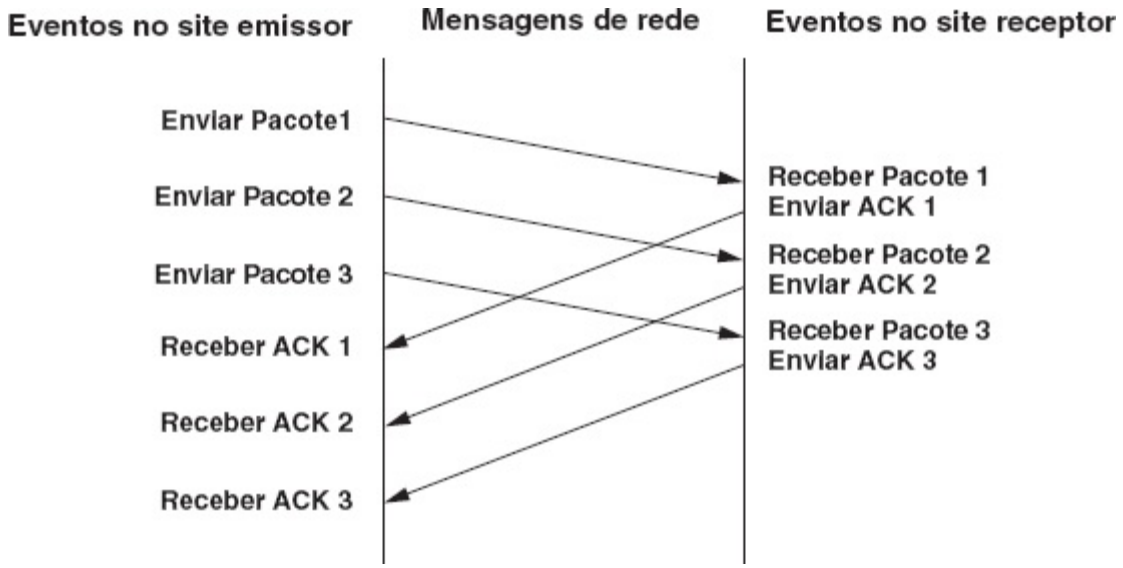


Figura 11.4 Um exemplo de janela deslizante com um janela de tamanho três.

Conceitualmente, um protocolo de janela deslizante sempre lembra quais pacotes foram confirmados e mantém um timer separado para cada confirmação de pacote. Se um pacote é perdido, o timer expira e o remetente retransmite esse pacote. Quando o remetente desliza sua janela, ele ultrapassa todos os pacotes confirmados. Na extremidade receptora, o software de protocolo mantém uma janela análoga, aceitando e confirmando os pacotes à medida que eles chegam. Assim, a janela particiona a sequência de pacotes em três conjuntos: os pacotes à esquerda da janela foram transmitidos, recebidos e confirmados com sucesso; os pacotes à direita ainda não foram transmitidos; e os pacotes que estão na janela estão sendo transmitidos. O pacote de número mais baixo na janela é o primeiro pacote na sequência que não foi reconhecido.

11.6 O Transmission Control Protocol (TCP)

Agora que entendemos o princípio de janelas deslizantes, podemos examinar o *Transmission Control Protocol* (*TCP*), o protocolo que fornece serviço de fluxo confiável. O serviço de fluxo é tão significativo que todo o conjunto de protocolos da Internet é conhecido como TCP/IP.

Vamos fazer uma distinção fundamental entre o protocolo TCP e o software que implementa o TCP. O protocolo TCP fornece uma especificação análoga a um projeto (*blueprint*); o software TCP implementa a especificação. Embora às vezes seja conveniente pensar em TCP como um pedaço de software, os leitores devem reconhecer a distinção

a seguir.

TCP é um protocolo de comunicação, não um pedaço de software.

Exatamente o que o TCP fornece? O TCP é complexo, por isso, não há uma resposta simples. O protocolo especifica o formato dos dados e as confirmações que dois computadores trocam para conseguir uma transferência confiável, bem como os procedimentos que os computadores usam para garantir que os dados chegaram corretamente. Ele especifica como o software TCP distingue múltiplos destinos em uma dada máquina, e como, quando se comunicam, as máquinas se recuperam de erros como perda ou duplicação de pacotes. O protocolo também detalha como dois computadores iniciam uma conexão TCP e como eles aceitam quando ela está completa.

Também é importante entender o que o protocolo não inclui. Embora a especificação TCP descreva como os aplicativos usam o TCP em termos gerais, não define os detalhes de interface entre um programa de aplicação e TCP. Ou seja, a documentação do protocolo só discute as operações que o TCP fornece; não especifica os procedimentos exatos que os aplicativos invocam para acessar as operações. A razão para deixar a interface do aplicativo não especificada é a flexibilidade. Em particular, porque o software TCP faz parte do sistema operacional de um computador, ele precisa empregar qualquer interface que o sistema operacional forneça. Permitir flexibilidade aos implementadores torna possível ter uma única especificação para TCP, que pode ser utilizada para construir software para uma variedade de sistemas de computador.

Por não fazer suposições sobre o sistema de comunicação subjacente, o TCP pode ser usado em uma ampla variedade de redes subjacentes. Ele pode ser executado através de satélite com longos atrasos, uma rede sem fio, onde a interferência causa a perda de muitos pacotes ou uma conexão alugada em que atrasos variam drasticamente dependendo do congestionamento existente. Sua capacidade de acomodar uma grande variedade de redes subjacentes é um dos pontos fortes do TCP.

11.7 Camadas, portas, conexões e extremidades

O TCP, que reside na camada de transporte logo acima do IP, permite que vários programas de aplicação em um determinado computador se comuniquem simultaneamente, e ele demultiplexa o tráfego TCP de

entrada entre as aplicações. Assim, nos termos do modelo de camadas, o TCP é um par conceitual da UDP, como mostra a Figura 11.5.

Apesar de estarem na mesma camada conceitual, TCP e UDP fornecem serviços completamente diferentes. Vamos entender muitas dessas diferenças ao longo do capítulo.

Como o User Datagram Protocol, o TCP usa números de *porta de protocolo* para identificar programas de aplicação. Também como o UDP, um número de porta TCP tem tamanho de 16 bits. A cada porta TCP é atribuído um número inteiro pequeno usado para identificá-lo. É importante entender que as portas TCP são conceitualmente independentes de portas UDP – um aplicativo pode usar a porta UDP 30000 enquanto outro usa a porta TCP 30000.

Dissemos que uma porta UDP consiste de uma fila que mantém datagramas de entrada. Portas TCP são muito mais complexas porque um único número de porta não identifica uma aplicação. Em vez disso, o TCP foi projetado sobre a *abstração da conexão* em que os objetos a serem identificados são conexões TCP, e não portas individuais. Cada conexão TCP é especificada por um par de extremidades que corresponde ao par de aplicativos que se comunicam. Compreender que o TCP usa a noção de conexões é fundamental porque ajuda a explicar o significado e o uso de números de portas TCP.

O TCP usa a conexão, não a porta de protocolo, como sua abstração fundamental; conexões são identificadas por um par de extremidades.

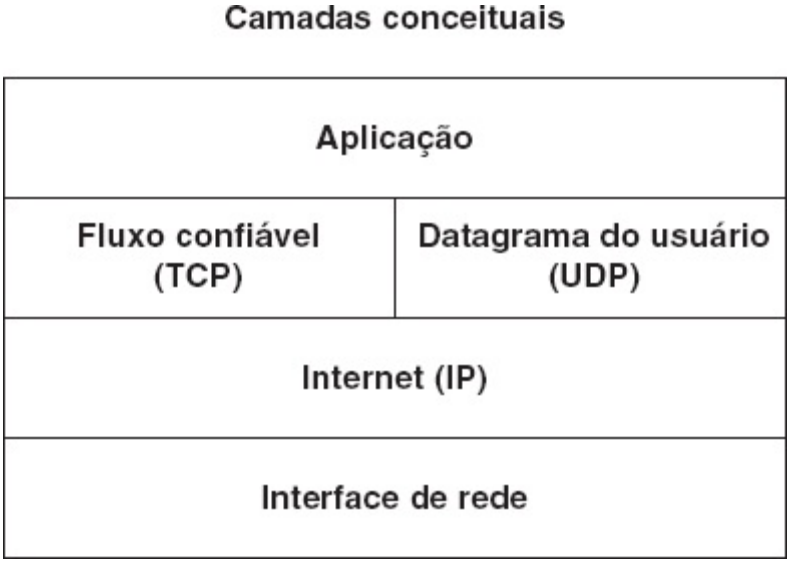


Figura 11.5 As camadas conceituais de UDP e TCP acima do IP.

Exatamente, o que são as "extremidades" de uma conexão TCP? Dissemos que uma conexão consiste de um circuito virtual entre dois programas aplicativos, por isso pode ser natural supor que um programa aplicativo serve como ponto de extremidade de conexão. Não é. Em vez disso, O TCP define uma extremidade como um par de inteiros (*host*, *porta*), em que *host* é o endereço IP para um *host* e *porta* é uma porta TCP nesse host. Por exemplo, uma extremidade IPv4 (*128.10.2.3, 25*) especifica a porta TCP *25* na máquina com o endereço IPv4 *128.10.2.3*.

Agora que definimos extremidades, é fácil de compreender conexões TCP. Lembre-se de que uma conexão é definida por suas duas extremidades. Assim, se há uma conexão da máquina (*18.26.0.36*) no MIT para a máquina (*128.10.2.3*) na Universidade de Purdue, ela pode ser definida pelas extremidades

(*18.26.0.36, 1069*) e (*128.10.2.3, 25*).

Enquanto isso, outra conexão pode estar em andamento a partir da máquina (*128.9.0.32*) no Instituto de Ciências da Informação para a mesma máquina em Purdue, identificada por suas extremidades

(*128.9.0.32, 1184*) e (*128.10.2.3, 53*).

Até agora, os nossos exemplos de conexões foram simples, porque as portas usadas em todas as extremidades têm sido únicas. No entanto, a abstração de conexão permite que várias conexões compartilhem uma extremidade. Por exemplo, poderíamos acrescentar outra conexão às duas listadas anteriormente, a da máquina (*128.2.254.139*) na CMU para a máquina em Purdue usando as extremidades

(*128.2.254.139, 1184*) e (*128.10.2.3, 53*).

Pode parecer estranho que duas conexões possam usar a porta TCP *53* na máquina 128.10.2.3 ao mesmo tempo, mas não há nenhuma ambiguidade. Como o TCP associa entrada de mensagens com uma conexão em vez de uma porta de protocolo, ele usa ambas as extremidades para identificar a conexão apropriada. A ideia importante a lembrar é descrita a seguir.

Como o TCP identifica uma conexão por um par de extremidades, um determinado número de porta TCP pode ser compartilhado por várias conexões na mesma máquina.

Do ponto de vista de um programador, a abstração de conexão é significativa. Isso significa que um programador pode desenvolver um programa que ofereça um serviço simultâneo a múltiplas conexões, sem a necessidade de números de portas locais exclusivos para cada conexão. Por exemplo, a maioria dos sistemas fornece acesso simultâneo ao seu serviço de correio eletrônico, permitindo que vários computadores enviem correio eletrônico simultaneamente para ele. Como ele usa o TCP para se comunicar, o aplicativo que aceita e-mails de entrada só precisa usar uma porta TCP local, mesmo que múltiplas conexões possam atuar simultaneamente.

11.8 Aberturas passivas e ativas

Diferentemente do UDP, o TCP é um protocolo orientado a conexão, que exige que as duas extremidades concordem em participar. Ou seja, antes que o tráfego TCP possa passar por uma rede, os programas aplicativos nas duas extremidades da conexão precisam combinar que a conexão é desejada. Para isso, o programa aplicativo em uma extremidade realiza uma função de *abertura passiva*, contatando seu sistema operacional e indicando que ele aceitará uma conexão que chega para um número de porta específico. O software de protocolo se prepara para aceitar uma conexão nessa porta. O programa aplicativo na outra extremidade executa uma *abertura ativa* requerendo que uma conexão TCP seja estabelecida. Os dois módulos de TCP se comunicam para estabelecer e verificar uma conexão. Quando uma conexão tiver sido criada, os programas aplicativos podem começar a passar dados; os módulos de TCP em cada extremidade trocam mensagens que garantem a entrega confiável. Retornaremos aos detalhes do estabelecimento de conexões depois de examinarmos o formato da mensagem TCP.

11.9 Segmentos, fluxos e números de sequência

O TCP vê o fluxo de dados como uma sequência de octetos ou bytes que ele divide em *segmentos* para transmissão. Normalmente, cada segmento

trafega por uma internet subjacente em um único datagrama IP. O TCP usa um mecanismo especializado de janela deslizante para solucionar dois problemas importantes: transmissão eficiente e controle de fluxo. Assim como o protocolo de janela deslizante descrito anteriormente, o mecanismo de janela TCP possibilita o envio de vários segmentos antes que uma confirmação chegue. Isso aumenta o throughput total, pois mantém a rede ocupada. O formato TCP de um protocolo de janela deslizante também soluciona o problema de *controle de fluxo* de fim a fim, permitindo que o receptor restrinja a transmissão até que tenha espaço de buffer suficiente para acomodar mais dados.

O mecanismo de janela deslizante do TCP opera no nível de octeto, e não no nível de segmento ou pacote. Os octetos do fluxo de dados são numerados sequencialmente, e um emissor mantém três ponteiros associados a cada conexão. Os ponteiros definem uma janela deslizante, conforme a Figura 11.6 ilustra. O primeiro ponteiro marca a esquerda da janela deslizante, separando os octetos que foram enviados e confirmados daqueles que ainda serão confirmados. Um segundo ponteiro marca a direita da janela deslizante e define o octeto mais alto na sequência que pode ser enviado antes que mais confirmações sejam recebidas. O terceiro ponteiro marca o limite dentro da janela que separa aqueles octetos que já foram enviados dos octetos que não foram enviados. O protocolo envia todos os octetos na janela sem atraso, de modo que o limite dentro da janela normalmente se move rapidamente da esquerda para a direita.

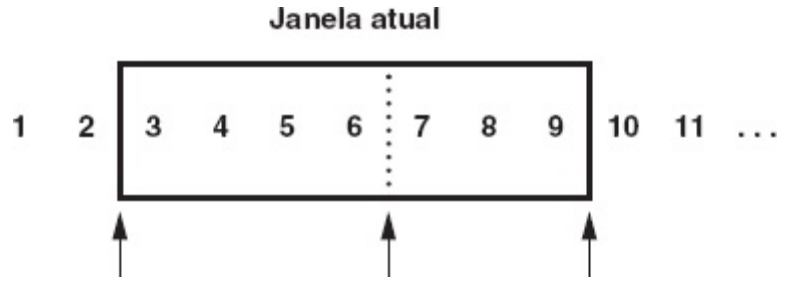


Figura 11.6 Um exemplo da janela deslizante do TCP. Os octetos até 2 foram enviados e confirmados, os octetos 3 a 6 foram enviados, mas não confirmados, os octetos 7 a 9 não foram enviados, mas serão enviados sem atraso, e os octetos 10 em diante não podem ser enviados até que a janela se mova.

Descrevemos como a janela TCP do emissor se desliza e mencionamos que o receptor precisa manter uma janela semelhante para recriar o fluxo. Porém, é importante entender que, como as conexões TCP são full duplex,

duas transferências prosseguem simultaneamente em cima de cada conexão, uma em cada direção. Consideramos as transferências completamente independentes, pois a qualquer momento os dados podem fluir pela conexão em uma direção, ou em ambas as direções. Assim, o TCP em um computador mantém duas janelas por conexão, uma desliza enquanto os dados são enviados, e a outra desliza enquanto os dados são recebidos.

11.10 Tamanho de janela variável e controle de fluxo

Uma diferença entre o protocolo de janela deslizante do TCP e o protocolo de janela deslizante simplificado, apresentado na Figura 11.4,* é que o TCP permite que o tamanho da janela varie com o tempo. Cada confirmação que especifica quantos octetos foram recebidos contém um *anúncio de janela* que especifica quantos octetos adicionais de dados o receptor está preparado para aceitar além dos dados que estão sendo confirmados. Pensamos no anúncio da janela como especificando o tamanho de buffer atual do receptor. Em resposta a um anúncio de janela maior, o emissor aumenta o tamanho de sua janela deslizante e prossegue para enviar octetos que não foram confirmados. Em resposta a um anúncio de janela diminuída, o emissor diminui o tamanho de sua janela e para de enviar octetos além do limite. O TCP não pode contradizer os anúncios anteriores, encurtando a janela além das posições anteriormente aceitáveis no fluxo de octetos. Em vez disso, anúncios menores acompanham confirmações, de modo que o tamanho da janela muda na hora em que ela desliza para a frente.

A vantagem de usar uma janela de tamanho variável é que ela oferece a habilidade de controlar o fluxo. Para evitar receber mais dados do que pode armazenar, o receptor envia anúncios de janela menores enquanto seu buffer se enche. No caso extremo, o receptor anuncia um tamanho de janela zero para interromper todas as transmissões. Mais adiante, quando o espaço do buffer se tornar disponível, o receptor anunciará um tamanho de janela diferente de zero para disparar o fluxo de dados novamente.**

É essencial ter um mecanismo para controle de fluxo em um ambiente no qual os computadores de várias velocidades e tamanhos se comunicam por redes e roteadores de diversas velocidades e capacidades. Existem dois problemas independentes. Primeiro, os protocolos precisam oferecer controle de fluxo de fim a fim entre a origem e o destino final. Por exemplo, quando um smartphone se comunica com um supercomputador poderoso, o smartphone precisa regular o fluxo de dados, ou o protocolo ficaria

rapidamente defasado. Assim, o TCP precisa implementar o controle de fluxo de fim a fim para garantir a entrega confiável. Em segundo lugar, é necessário haver um mecanismo que permita que os sistemas intermediários (ou seja, roteadores) controlem uma origem que envia mais tráfego do que a máquina pode tolerar.

Quando as máquinas intermediárias se tornam sobrecarregadas, a condição é chamada de *congestionamento*, e os mecanismos para solucionar o problema são denominados *mecanismos de controle de congestionamento*. O TCP usa seu esquema de janela deslizante para solucionar o problema de controle de fluxo de fim a fim. Discutiremos o controle de congestionamento mais tarde, mas deve ser observado que um protocolo bem projetado pode detectar e recuperar-se do congestionamento, enquanto um protocolo mal projetado tornará o congestionamento pior. Em particular, um esquema de retransmissão cuidadosamente escolhido pode ajudar a evitar o congestionamento, mas um esquema mal escolhido pode exacerbá-lo retransmitindo agressivamente.

11.11 Formato do segmento TCP

A unidade de transferência entre o TCP nas duas máquinas é chamada de *segmento*. Os segmentos são trocados para se estabelecer conexões, transferir dados, enviar confirmações, anunciar tamanhos de janela e fechar conexões. Como o TCP utiliza o *piggybacking*, uma confirmação trafegando do computador *A* para o computador *B* pode trafegar no mesmo segmento dos dados atravessando do computador *A* para o computador *B*, embora a confirmação se refira aos dados enviados de *B* para *A*.***

Como a maioria dos protocolos, uma mensagem é dividida em duas partes conceituais: um cabeçalho que contém metadados e uma área de carga útil que leva dados. A Figura 11.7 mostra o formato do segmento TCP.

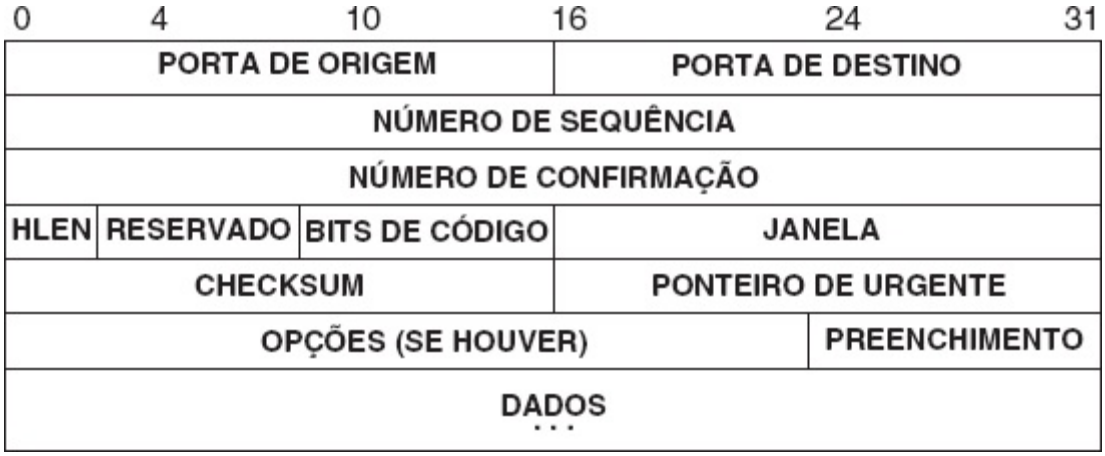


Figura 11.7 Formato do segmento TCP com um cabeçalho TCP seguido por uma carga útil.

O cabeçalho, conhecido como *cabeçalho TCP*, consiste de pelo menos 20 octetos e deve conter mais se o segmento carregar opções. O cabeçalho tem a identificação esperada e informações de controle. Os campos PORTA DE ORIGEM e PORTA DE DESTINO contêm os números de porta TCP que identificam os programas aplicativos nas extremidades da conexão. O campo NÚMERO DE SEQUÊNCIA identifica a posição no fluxo de bytes do emissor dos dados no segmento. O campo NÚMERO DE CONFIRMAÇÃO identifica o número do octeto que a origem espera receber em seguida. Observe que o número de sequência se refere ao fluxo que segue na mesma direção do segmento, enquanto o número de confirmação refere-se ao fluxo que segue na direção oposta do segmento.

O campo *HLEN** contém um inteiro que especifica o tamanho do cabeçalho do segmento medido em múltiplos de 32 bits. Ele é necessário porque o campo OPÇÕES varia em tamanho, dependendo de quais opções foram incluídas. Assim, o tamanho do cabeçalho TCP varia dependendo das opções selecionadas. O campo de 6 bits marcado como RESERVADO é reservado para uso futuro (outra seção descreve um uso proposto).

Alguns segmentos transportam apenas uma confirmação, enquanto outros transportam dados. Já outros, ainda, transportam requisições para estabelecer ou fechar uma conexão. O TCP utiliza um campo de 6 bits rotulado como BITS DE CÓDIGO para determinar a finalidade e o conteúdo do segmento. Os seis bits dizem como interpretar outros campos no cabeçalho, de acordo com a tabela na Figura 11.8.

O TCP anuncia quantos dados ele espera aceitar toda vez que envia um segmento, especificando seu tamanho de buffer no campo JANELA. O

campo contém um inteiro não sinalizado de 16 bits na ordem de bytes padrão da rede. Anúncios de janela oferecem outro exemplo de piggybacking, pois acompanham todos os segmentos, incluindo aqueles transportando dados e também aqueles carregando apenas uma confirmação.

Bit (esquerda para direita)	Significado se bit definido como 1
URG	Campo ponteiro de urgente é válido
ACK	Campo de confirmação é válido
PSH	Este segmento requer um push
RST	Reinicia a conexão
SYN	Sincroniza números de sequência
FIN	Emissor alcançou final do seu fluxo de bytes

Figura 11.8 Bits do campo BITS DE CÓDIGO no cabeçalho TCP.

11.12 Dados fora de faixa (out of band data)

Embora o TCP seja um protocolo orientado para fluxo, às vezes é importante que o programa em uma extremidade de uma conexão envie dados *fora da faixa*, sem esperar que o programa na outra extremidade da conexão consuma octetos já no fluxo. Por exemplo, quando o TCP é usado para uma aplicação em um desktop remoto, o usuário pode decidir enviar uma sequência de teclado que *interrompe* ou *aborta* o programa na outra extremidade. Esses sinais normalmente são necessários quando um programa na máquina remota deixa de operar corretamente. Os sinais precisam ser enviados sem esperar que o programa leia octetos já no fluxo TCP (ou alguém não seria capaz de abordar programas que param de ler a entrada).

Para acomodar a sinalização fora de faixa, o TCP permite que o emissor especifique dados como *urgentes*, significando que o programa receptor deve ser notificado sobre sua chegada o mais rápido possível, independentemente de sua posição no fluxo. O protocolo especifica que, quando dados urgentes são encontrados, o TCP receptor deve notificar o programa aplicativo associado à conexão para entrar no "modo urgente". Depois que todos os dados urgentes tiverem sido consumidos, o TCP diz ao programa aplicativo para retornar à operação normal.

Os detalhes exatos de como o TCP informa a um aplicativo sobre dados

urgentes dependem do sistema operacional do computador. O mecanismo usado para marcar dados urgentes ao transmiti-los em um segmento consiste no bit de código URG e no campo PONTEIRO DE URGENTE no segmento cabeçalho. Quando o bit URG é marcado, o campo PONTEIRO DE URGENTE especifica a posição no segmento onde os dados urgentes terminam.

11.13 Opções do TCP

Como mostra a Figura 11.7, um cabeçalho TCP pode conter zero ou mais *opções*; a próxima seção explica as opções disponíveis. Cada opção começa com um campo de 1 octeto que especifica a opção tipo 5, seguida por um campo de tamanho de 1 octeto que especifica o tamanho da opção em octetos. Se as opções não ocuparem um múltiplo exato de 32 bits, o PREENCHIMENTO é acrescentado ao final do cabeçalho.

11.13.1 Opção de tamanho máximo do segmento

Um emissor pode escolher a quantidade de dados que é colocada em cada segmento. Porém, as duas extremidades de uma conexão TCP precisam combinar sobre um segmento máximo que elas transferirão. O TCP usa uma opção de *tamanho máximo de segmento* (*Maximum Segment Size – MSS*) para permitir que um receptor especifique o segmento de tamanho máximo que ele desejará receber. Um sistema embutido que possui apenas algumas centenas de bytes de espaço em buffer pode especificar um MSS que restringe os segmentos de modo que caibam no buffer. A negociação do MSS é especialmente significativa porque permite que sistemas heterogêneos se comuniquem; um supercomputador pode se comunicar com um pequeno nó de um sensor wireless. Para maximizar o throughput, quando dois computadores se conectam à mesma rede física, o TCP normalmente calcula um tamanho máximo de segmento de modo que os datagramas IP resultantes combinem com a MTU da rede. Se as extremidades não estiverem na mesma rede física, elas poderão tentar descobrir a MTU mínima ao longo do caminho entre elas, ou escolher um segmento de tamanho máximo igual ao tamanho da menor carga útil do datagrama.

Em um ambiente geral de internet, a escolha de um bom tamanho máximo de segmento pode ser difícil, pois o desempenho pode ser fraco para tamanhos de segmento extremamente grandes ou extremamente pequenos. Por um lado, quando o tamanho do segmento é pequeno, a

utilização da rede permanece baixa. Para ver por que razão, lembre-se de que os segmentos TCP trafegam encapsulados nos datagramas IP, que estão encapsulados nos frames físicos da rede. Assim, cada frame leva pelo menos 20 octetos de cabeçalho TCP mais 20 octetos de cabeçalho IP (IPv6 é o maior). Portanto, os datagramas transportando apenas um octeto de dados utilizam no máximo 1/41 da largura de banda da rede subjacente para os dados sendo transferidos (menos para o IPv6).

Por outro lado, tamanhos de segmento extremamente grandes também podem produzir um desempenho fraco. Segmentos grandes resultam em datagramas IP grandes. Quando esses datagramas atravessam uma rede com MTU pequena, o IP precisa fragmentá-los. Diferentemente de um segmento TCP, um fragmento não pode ser confirmado ou retransmitido de forma independente; todos os fragmentos devem chegar, ou o datagrama inteiro precisa ser retransmitido. Como a probabilidade de perder determinado fragmento é diferente de zero, aumentar o tamanho de segmento acima do patamar de fragmentação diminui a probabilidade de o datagrama chegar, o que diminui o rendimento.

Em teoria, o tamanho de segmento ideal, *S*, ocorre quando os datagramas IP transportando os segmentos são os maiores possíveis sem exigir fragmentação em qualquer lugar ao longo do caminho da origem até o destino. Na prática, encontrar *S* significa encontrar o caminho MTU, que envolve sondagem. Para uma conexão TCP de curta duração (ou seja, em que somente alguns pacotes são trocados), a sondagem pode introduzir atraso. Segundo, como os roteadores em uma internet podem mudar de rotas dinamicamente, o caminho que os datagramas seguem entre um par de computadores em comunicação pode mudar dinamicamente, assim como o tamanho em que os datagramas precisam ser fragmentados. Terceiro, o tamanho ideal depende dos cabeçalhos de protocolo de nível inferior (por exemplo, o tamanho do segmento será menor se o datagrama incluir opções IPv4 ou cabeçalhos de extensão IPv6).

11.13.2 Opções de escalas de janela

Como o campo JANELA no cabeçalho TCP tem 16 bits de extensão, o tamanho máximo da janela é de 64 Kbytes. Embora a janela seja suficiente para as primeiras redes, um tamanho de janela maior é necessário para obter o alto rendimento em uma rede como um canal de satélite que possui um produto com grande atraso-largura de banda (informalmente chamado de *long fat pipe*).

Para acomodar tamanhos de janela maiores, uma *opção de escala de janela* foi criada para o TCP. A opção consiste em três octetos: um tipo, um tamanho e um valor de *shift*, *S*. Essencialmente, o valor de shift especifica um fator de escala binário a ser aplicado ao valor da janela. Quando a escala da janela está em vigor, um receptor extrai o valor do campo JANELA, *W*, e desloca *W* à esquerda por *S* bits, a fim de obter o tamanho real da janela.

Diversos detalhes complicam o projeto. A opção pode ser negociada quando a conexão é estabelecida inicialmente, quando todos os anúncios de janela sucessivos são considerados como usando a escala negociada, ou a opção pode ser especificada em cada segmento, quando o fator de escala pode variar de um segmento para outro. Além do mais, se qualquer lado de uma conexão implementar a escala de janela, mas não precisar dimensionar sua janela, esse lado enviará a opção definida como zero, o que cria o fator de escala 1.

11.13.3 Opção de estampa de tempo

A *opção de estampa de tempo* TCP foi inventada para ajudar o TCP a calcular o atraso na rede subjacente, e também é usada para lidar com o caso em que os números de sequência do TCP excedem 2^{32} (conhecidos como números *Protect Against Wrapped Sequence, PAWS*). Além dos campos obrigatórios de tipo e tamanho, uma opção de estampa de tempo inclui dois valores: um valor de estampa de tempo e um valor de estampa de tempo de resposta de eco. Um emissor coloca a hora de seu clock atual no campo de estampa de tempo ao enviar um pacote; um receptor copia o campo de estampa de tempo para o campo de resposta de eco antes de retornar uma confirmação para o pacote. Assim, quando uma confirmação chega, o emissor pode calcular com precisão o tempo gasto total desde que o segmento foi enviado.

11.14 Cálculo do checksum TCP

O campo de CHECKSUM do cabeçalho de TCP é um checksum complementar de um de16 bits utilizado para verificar a integridade dos dados, bem como o cabeçalho do TCP. Tal como acontece com outros checksums, o TCP usa 16 bits aritmético e pega o complemento de um da soma de complementos de um. Para calcular o checksum, o software TCP

na máquina emissora segue um procedimento semelhante ao descrito no Capítulo 10 para UDP. Conceitualmente, o TCP acrescenta um *pseudocabeçalho* para o segmento TCP, acrescenta bits zero suficientes para tornar o segmento um múltiplo de 16 bits, e calcula o checksum de 16 bits ao longo de todo o resultado. Tal como acontece com UDP, o pseudocabeçalho não é parte do segmento, e nunca é transmitido num pacote. No site receptor, o software TCP extrai campos do cabeçalho IP, reconstrói um pseudocabeçalho e realiza o mesmo cálculo do checksum para verificar se o segmento chegou intacto.

A finalidade do uso de um pseudocabeçalho é exatamente a mesma que no UDP. Ele permite que o receptor verifique se o segmento alcançou a extremidade correta, que inclui um endereço IP e também um número de porta de protocolo. Os endereços IP de origem e destino são importantes para o TCP porque ele precisa usá-los para identificar a conexão à qual o segmento pertence. Portanto, sempre que um datagrama chega transportando um segmento TCP, o IP precisa passar ao TCP os endereços IP de origem e destino do datagrama, além do próprio segmento. A Figura 11.9 mostra o formato do pseudocabeçalho usado no cálculo do checksum para IPv4, enquanto a Figura 11.10 mostra o formato para um pseudocabeçalho IPv6.

Claro, o pseudocabeçalho IPv4 usa endereços IPv4 de origem e destino e o pseudocabeçalho IPv6 usa endereços IPv6. Ao campo PROTOCOLO (IPv4) ou ao campo PRÓXIMO CABEÇALHO (IPv6) é atribuído o valor 6, o valor de datagramas que transportam TCP. O campo TAMANHO TCP especifica o tamanho total do segmento TCP, incluindo seu cabeçalho.

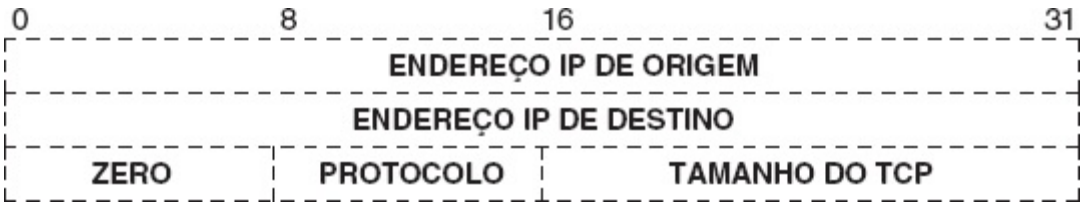


Figura 11.9 O pseudocabeçalho de 12 octetos do IPv4 usado no cálculo do checksum TCP.

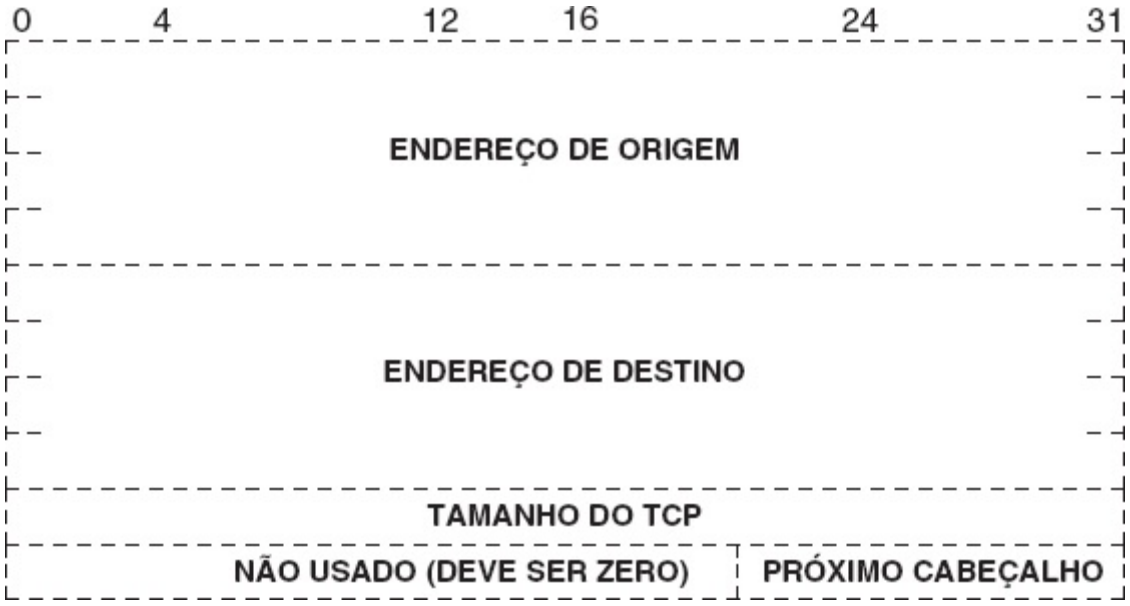


Figura 11.10 O pseudocabeçalho de 40 octetos do IPv6 usado no cálculo do checksum TCP.

11.15 Confirmações, retransmissão e timeouts

Como o TCP envia dados em segmentos de tamanho variável, e como os segmentos retransmitidos podem incluir mais dados do que o original, uma confirmação não pode se referir facilmente a um datagrama ou a um segmento. Em vez disso, uma confirmação se refere a uma posição no fluxo usando os números de sequência de fluxo. O receptor coleta octetos de dados dos segmentos que chegam e reconstrói uma cópia exata do fluxo sendo enviado. Como os fragmentos trafegam nos datagramas IP, eles podem ser perdidos ou entregues fora de ordem; o receptor usa os números de sequência para reordenar segmentos. A qualquer momento, o receptor terá reconstruído zero ou mais octetos contíguos desde o início do fluxo, mas pode ter partes adicionais do fluxo dos datagramas que chegaram fora de ordem. O receptor sempre confirma o prefixo contíguo mais longo do fluxo que foi recebido corretamente. Cada confirmação especifica um valor a mais de sequência que a posição de octeto mais alta no prefixo contíguo que recebeu. Assim, o emissor recebe o feedback contínuo do receptor enquanto prossegue pelo fluxo. Podemos resumir essa ideia importante a seguir.

Uma confirmação TCP especifica o número de sequência do próximo octeto que o receptor espera receber.

O esquema de confirmação do TCP é chamado de *cumulativo* porque informa quanto do fluxo foi acumulado. As confirmações cumulativas possuem vantagens e desvantagens. Uma vantagem é que as confirmações são fáceis de gerar e não ambíguas. Outra vantagem é que as confirmações perdidas não necessariamente forçam a retransmissão. Uma desvantagem importante é que o emissor não recebe informações sobre todas as transmissões bem-sucedidas, mas somente sobre uma única posição no fluxo que foi recebido.

Para entender por que a falta de informação sobre todas as transmissões bem-sucedidas torna as confirmações cumulativas menos eficientes, pense em uma janela que se espalha por *5 mil* octetos, começando na posição *101* no fluxo, e suponha que o emissor tenha transmitido todos os dados na janela enviando cinco segmentos. Suponha ainda que o primeiro segmento seja perdido, mas todos os outros cheguem intactos. À medida que cada segmento chega, o receptor envia uma confirmação, mas cada confirmação especifica o octeto *101*, o próximo octeto contíguo mais alto que ele espera receber. Não existe um meio de o receptor dizer ao emissor que a maioria dos dados para a janela atual chegou.

No nosso exemplo, quando ocorre um timeout no lado do emissor, este precisa escolher entre dois esquemas potencialmente ineficazes. Ele pode decidir retransmitir um segmento ou todos os cinco segmentos. Nesse caso, retransmitir todos os cinco segmentos é ineficaz. Assim que o primeiro segmento chegar, o receptor terá todos os dados na janela e confirmará *5101*. Se o emissor seguir o padrão aceito e retransmitir apenas o primeiro segmento não confirmado, ele terá de esperar a confirmação antes que possa decidir o que e quanto será enviado. Assim, ele reverte para um protocolo de confirmação positiva simples e pode perder as vantagens de ter uma janela grande.

Uma das ideias mais importantes e complexas no TCP está incorporada no modo como ele trata o timeout e a retransmissão. Assim como outros protocolos confiáveis, o TCP espera que o destino envie confirmações sempre que receber com sucesso novos octetos do fluxo de dados. Toda vez que envia um segmento, o TCP inicia um timer e espera uma confirmação. Se o timer expirar antes que os dados no segmento tenham sido confirmados, o TCP assumirá que o segmento foi perdido ou adulterado e o retransmitirá.

Para entender por que o algoritmo de retransmissão do TCP difere do algoritmo usado em muitos protocolos de rede, precisamos nos lembrar de

que o TCP foi criado para uso em um ambiente de internet. Em uma internet, um segmento trafegando entre um par de máquinas pode atravessar uma única rede de baixo atraso (por exemplo, uma LAN de alta velocidade), ou pode atravessar várias redes intermediárias, passando por diversos roteadores. Assim, é impossível conhecer *a priori* a rapidez com que as confirmações retornarão à origem. Além do mais, o atraso em cada roteador depende do tráfego, de modo que o tempo total necessário para um segmento atravessar até o destino e uma confirmação retornar à origem pode variar muito de um instante para outro. Para entender, considere o mundo para o qual TCP foi projetado. A Figura 11.11 ilustra a situação mostrando medições de *tempos de ida e volta* (*round trip times* – RTTs) para 100 pacotes consecutivos enviados pela Internet global dos anos 1980.

Embora a maioria das redes modernas não se comporte tão mal, a plotagem mostra as situações em que o TCP é projetado para acomodar: atrasos incrivelmente longos e mudanças na demora das viagens de ida e volta de determinada conexão. Para lidar com essas situações, o TCP usa um *algoritmo de retransmissão adaptativa*. Ou seja, o TCP monitora o tempo de ida e volta em cada conexão e calcula valores razoáveis para os tempos de espera (timeouts). Como o desempenho de uma conexão muda, o TCP revisa seu valor de tempo limite (ou seja, adapta-se à mudança).

Para coletar os dados necessários para um algoritmo adaptativo, o TCP registra o tempo em que cada segmento é enviado e o tempo em que uma confirmação chega para os dados nesse segmento. Dos dois tempos, o TCP calcula um tempo decorrido, conhecido como tempo de ida e volta de amostra, ou *amostra de ida e volta* (*round trip sample*). Sempre que obtém uma nova amostra de ida e volta, o TCP ajusta sua noção do tempo de ida e volta médio para a conexão. Para isso, usa uma média ponderada para estimar o tempo de ida e volta e usa cada nova amostra de tempo de ida e volta para atualizar a média. A técnica original de cálculo da média usava um fator de peso constante, α, onde $0 \quad \alpha < 1$, para pesar a média antiga contra a amostra de ida e volta mais recente:*

$$RTT = \alpha \times RTT + (1 - \alpha) \times Nova\_Amostra\_De\_Ida\_E\_Volta$$

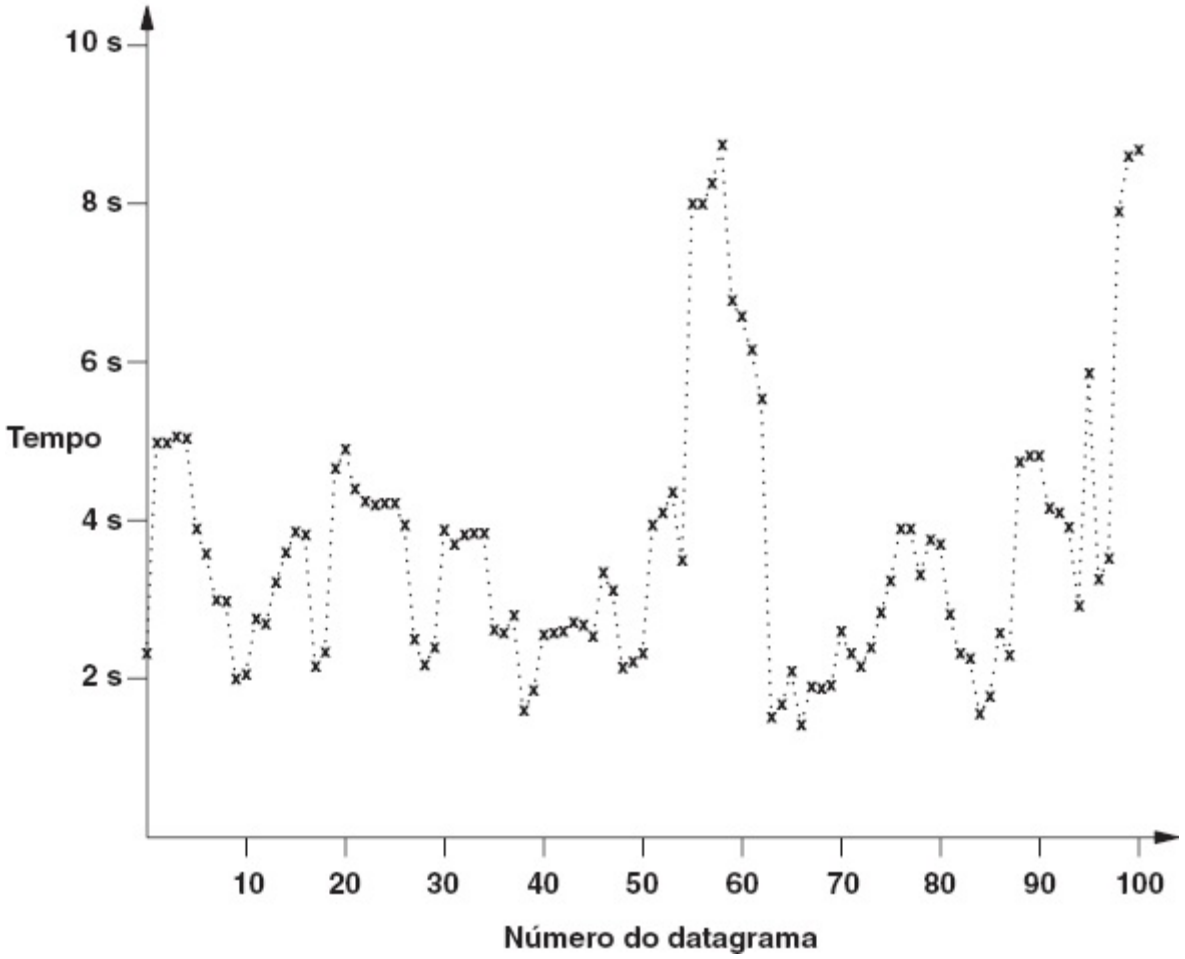


Figura 11.11 Um caso extremo para o TCP: uma plotagem de tempos de ida e volta da Internet dos anos 1980. Embora a Internet atualmente opere com atraso muito menor, atrasos continuam a variar com o tempo.

A ideia é que a escolha de um valor para α perto de *1* torna a média ponderada imune a mudanças que duram um pequeno tempo (por exemplo, um único segmento que encontra um atraso longo). A escolha de um valor para α próximo de *0* faz com que a média ponderada responda a mudanças no atraso muito rapidamente.

Ao enviar um segmento, o TCP calcula um valor de timeout como uma função da estimativa atual de ida e volta. As primeiras implementações do TCP usavam um fator de peso constante, β ($\beta > 1$), e tornavam o timeout maior que a estimativa atual de ida e volta:

$$\mathbf{Timeout = \beta \times RTT}$$

Podemos, a seguir, resumir as ideias apresentadas até agora.

Para acomodar os atrasos variáveis encontrados em um ambiente de internet, o TCP usa um algoritmo de retransmissão adaptativa que monitora atrasos em cada conexão e ajusta seu parâmetro de timeout de forma adequada.

11.16 Medição precisa de amostras de ida e volta

Teoricamente, a medição de uma amostra de ida e volta é trivial – ela consiste em subtrair a hora em que o segmento é enviado da hora em que a confirmação chega. Porém, surgem complicações porque o TCP usa um esquema de confirmação cumulativa, em que uma confirmação se refere aos dados recebidos, e não à instância de um datagrama específico que transportou os dados. Considere uma retransmissão. O TCP forma um segmento, coloca-o em um datagrama e o envia. O timer expira e o TCP envia o segmento novamente em um segundo datagrama. Como os dois datagramas transportam exatamente os mesmos segmentos de dados, o emissor não tem como saber se uma confirmação corresponde ao datagrama original ou ao retransmitido. Esse fenômeno tem sido chamado de *ambiguidade de confirmação*.

O TCP deve assumir que as confirmações pertencem à transmissão mais antiga (ou seja, original) ou à mais recente (ou seja, a retransmissão mais recente)? Surpreendentemente, nenhuma suposição funciona. A associação da confirmação com a transmissão original pode fazer com que o tempo de ida e volta estimado cresça sem limites nos casos em que uma internet perde datagramas*. Se uma confirmação chegar após uma ou mais retransmissões, o TCP medirá a amostra de ida e volta da transmissão original e calculará um novo RTT usando a amostra excessivamente longa. Assim, o RTT crescerá ligeiramente. Da próxima vez que o TCP enviar um segmento, o RTT maior resultará em timeouts ligeiramente maiores, de modo que, se uma confirmação chegar após uma ou mais retransmissões, o próximo tempo de ida e volta de amostra será ainda maior, e assim por diante.

A associação da confirmação com a retransmissão mais recente também pode falhar. Considere o que acontece quando o atraso de fim a fim aumenta de repente. Quando o TCP envia um segmento, ele usa a antiga estimativa de ida e volta para calcular um timeout, que agora é muito

pequeno. O segmento chega e uma confirmação inicia novamente, mas o aumento no atraso significa que o timer expira antes que a confirmação chegue, e o TCP retransmite o segmento. Logo depois que o TCP retransmite, a primeira confirmação chega e é associada à retransmissão. A amostra de ida e volta será muito pequena e resultará em um ligeiro decréscimo do tempo de ida e volta estimado, RTT. Infelizmente, reduzir o tempo de ida e volta estimado garante que o TCP definirá o timeout com um valor muito pequeno para o próximo segmento. Por fim, o tempo de ida e volta estimado pode se estabilizar em um valor, T, de modo que tal tempo correto seja ligeiramente maior que algum múltiplo de T. As implementações do TCP que associam confirmações à retransmissão mais recente têm sido observadas em um estado estável com RTT ligeiramente menor que metade do valor correto (ou seja, o TCP envia cada segmento exatamente duas vezes, embora não ocorra perda alguma).

11.17 Algoritmo de Karn e o timer backoff

Se a transmissão original e a transmissão mais recente deixarem de fornecer tempos de ida e volta precisos, o que o TCP deverá fazer? A resposta aceita é simples: o TCP não deverá atualizar a estimativa de ida e volta para os segmentos retransmitidos. A ideia, conhecida como *algoritmo de Karn*, evita completamente o problema de confirmações ambíguas, ajustando apenas a ida e a volta estimadas para confirmações não ambíguas (confirmações que chegam para segmentos que só foram transmitidos uma vez).

Naturalmente, uma implementação simplista do algoritmo de Karn, que simplesmente ignora os tempos dos segmentos retransmitidos, também pode ocasionar falhas. Considere o que acontece quando o TCP envia um segmento depois de um aumento súbito no atraso. O TCP calcula um timeout usando a estimativa de ida e volta existente. O timeout será muito pequeno para o novo atraso e forçará a retransmissão. Se o TCP ignorar confirmações de segmentos retransmitidos, ele nunca atualizará a estimativa e o ciclo continuará.

Para acomodar essas falhas, o algoritmo de Karn exige que o emissor combine timeouts de retransmissão com uma estratégia de *timer backoff* (recuo de relógio). A técnica de backoff calcula um timeout inicial usando uma fórmula como aquela que mostramos anteriormente. Porém, se o timer expirar e causar uma retransmissão, o TCP aumenta o timeout. Na verdade, toda vez que ele precisa retransmitir um segmento, o TCP aumenta o

timeout (para evitar que os timeouts se tornem ridiculamente longos, a maioria das implementações limita os aumentos a um limite superior que é maior que o atraso ao longo de qualquer caminho na internet).

As implementações utilizam diversas técnicas para calcular o recuo. A maioria escolhe um fator multiplicativo, γ, e define o novo valor como:

$$novo\_timeout = \gamma \times timeout$$

Normalmente, γ é 2. (Argumenta-se que valores de γ menores que 2 levam a instabilidades.) Outras implementações utilizam uma tabela de fatores multiplicativos, permitindo um recuo arbitrário em cada etapa.*

O algoritmo de Karn combina a técnica de backoff com a estimativa de ida e volta para solucionar o problema de nunca aumentar as estimativas de ida e volta.

O algoritmo de Karn, ao calcular a estimativa de ida e volta, ignora as amostras que correspondem a segmentos retransmitidos, mas usa uma estratégia de recuo e retém um valor de timeout de um pacote retransmitido para os pacotes subsequentes até que uma amostra válida seja obtida.

De um modo geral, quando uma internet se comporta mal, o algoritmo de Karn separa o cálculo do valor de timeout da estimativa atual de ida e volta. Ele usa a estimativa de ida e volta para calcular um valor de timeout inicial, mas depois recua o timeout a cada retransmissão, até que possa transferir um segmento com sucesso. Ao enviar segmentos subsequentes, o TCP retém o valor de timeout que resulta do recuo. Finalmente, quando chega uma confirmação correspondente a um segmento que não exibiu retransmissão, o TCP recalcula a estimativa de ida e volta e reinicia o timeout adequadamente. A experiência mostra que o algoritmo de Karn funciona bem mesmo em redes com alta perda de pacotes.**

11.18 Resposta à alta variância no atraso

A pesquisa na estimativa de ida e volta tem mostrado que os cálculos descritos anteriormente não se adaptam a uma grande gama de variação no atraso. A teoria do enfileiramento sugere que o tempo de ida e volta aumenta proporcional a $1/(1-L)$, onde L é a carga atual da rede, $0 \quad L < 1$,

e a variação no tempo de ida e volta, σ, é proporcional a $1/(1-L)2$. Se uma internet estiver rodando a 50% da capacidade, esperamos que o atraso de ida e volta varie por um fator de *4* a partir do tempo médio de ida e volta. Quando a carga atinge 80%, esperamos uma variação por um fator de *25*. O padrão TCP original especificou a técnica para estimar o tempo de ida e volta que descrevemos anteriormente. Usar essa técnica e limitar β ao valor sugerido de 2 significa que a estimativa de ida e volta pode se adaptar a cargas de no máximo 30%.

A especificação de 1989 para o TCP exige que as implementações estimem o tempo médio de ida e volta e a variância e utilizem a variância estimada no lugar do β constante. Como resultado, novas implementações do TCP podem se adaptar a um intervalo maior de variação no atraso e ocasionar um rendimento substancialmente mais alto. Felizmente, as aproximações exigem pouco cálculo; programas extremamente eficientes podem ser derivados das seguintes equações simples:

$$DIFF = SAMPLE - Antigo\_RTT$$

$$RTT\_Suavizado = Antigo\_RTT + \delta \times DIFF$$

$$DEV = Antigo\_DEV + \rho \times (DIFF - Antigo\_DEV)$$

$$Timeout = RTT\_Suavizado + \eta \times DEV$$

Nas quais *DEV* é o desvio médio estimado, δ é uma fração entre *0* e *1* que controla a rapidez com que a nova amostra afeta a média ponderada, ρ é uma fração entre *0* e *1* que controla a rapidez com que a nova amostra afeta o desvio médio e η é um fator que controla o quanto o desvio afeta o timeout de ida e volta.

Para tornar o cálculo eficiente, o TCP escolhe δ e ρ para que cada um seja um inverso de uma potência de *2*, escala o cálculo por 2^n para um n apropriado e usa a aritmética de inteiros. A pesquisa sugere que:

$$\delta = 1/2^3$$

$$\rho = 1/2^2$$

$$\eta = 4$$

O valor original para η no 4.3 BSD UNIX era *2*; após experiência e

medições, ele foi alterado para 4 no 4.4 BSD UNIX.

Para ilustrar como as versões atuais do TCP se adaptam às mudanças no atraso, foi utilizado um gerador de números aleatórios a fim de produzir um conjunto de tempos de ida e volta e alimentando esse conjunto na estimativa de ida e volta descrita acima. A Figura 11.12 ilustra o resultado. A plotagem mostra os tempos de ida e volta individuais plotados como pontos individuais e o tempo limite obtido plotado como uma linha sólida. Observe como o timer de retransmissão varia conforme muda o tempo de ida e volta. Embora os tempos de ida e volta sejam artificiais, eles seguem um padrão observado na prática: os pacotes sucessivos mostram pequenas variações no atraso conforme a média geral sobe ou desce.

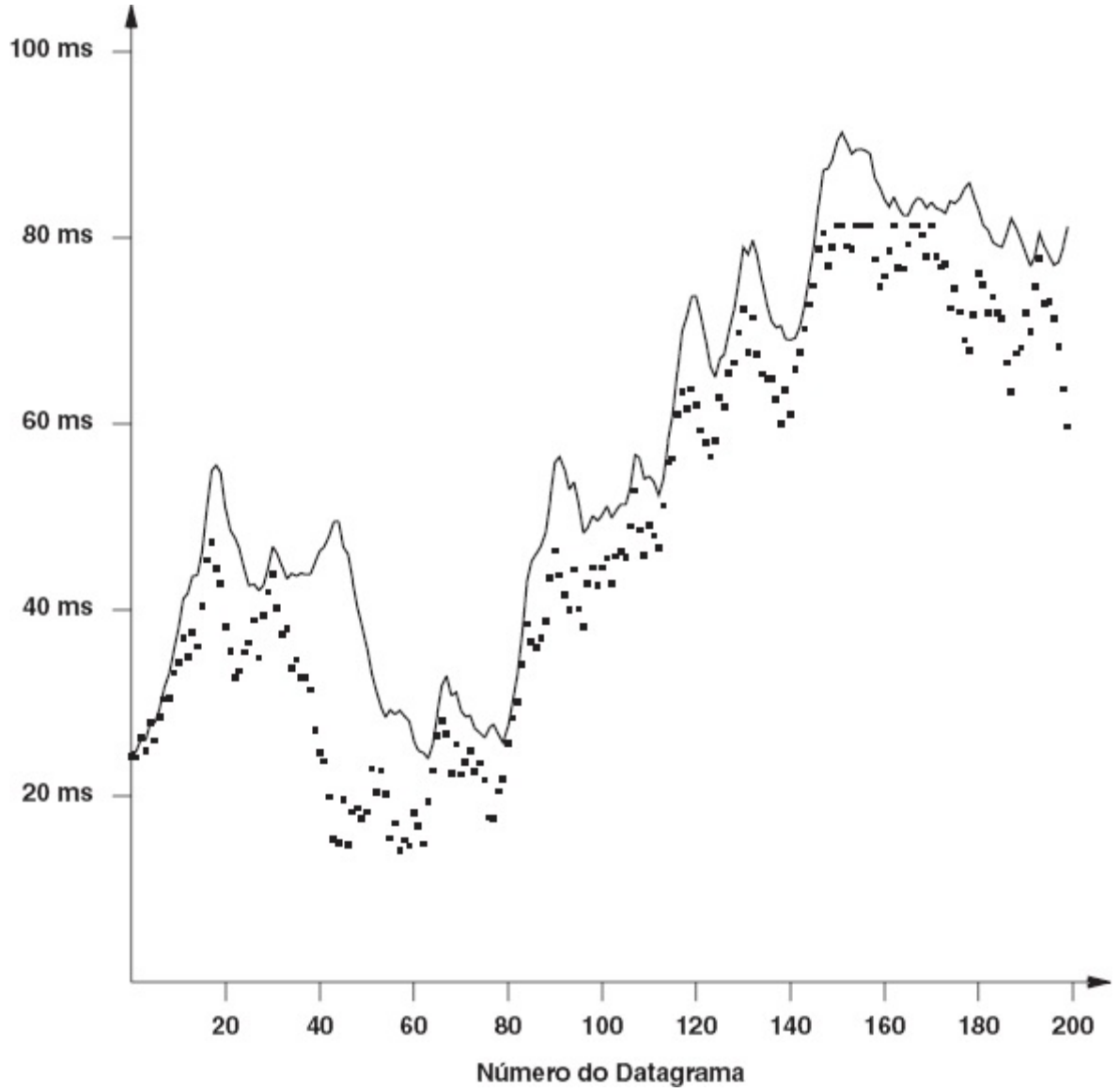


Figura 11.12 Conjunto de 200 tempos(gerados randomicamente) de ida e volta

mostrados como pontos, e o timer de retransmissão TCP mostrado como linha sólida.

Observe que a mudança frequente no tempo de ida e volta, incluindo um ciclo de aumento e diminuição, pode produzir um aumento no timer de retransmissão. Além do mais, embora o timer costume aumentar rapidamente quando o atraso aumenta, ele não diminui tão rapidamente quando o atraso cai.

Ainda que os dados gerados randomicamente ilustrem o algoritmo, é importante ver como o TCP se comporta nos piores casos. A Figura 11.13 utiliza as medições da Figura 11.11 para mostrar como o TCP responde a um caso extremo de variância no atraso. Lembre-se de que o objetivo é fazer com que o timer de retransmissão estime o tempo de ida e volta real o mais perto possível sem subestimar. A figura mostra que, embora responda rapidamente, ele pode subestimar. Por exemplo, entre os dois datagramas sucessivos marcados com setas, o atraso dobra de menos de 4 segundos até mais de 8. Mais importante, a mudança brusca segue um período de relativa estabilidade, em que a variação no atraso é pequena, tornando impossível para qualquer algoritmo antecipar a mudança. No caso do algoritmo do TCP, como o timeout (aproximadamente 5 segundos) subestima substancialmente o atraso grande, ocorre uma retransmissão desnecessária.

Porém, um timer de retransmissão responde rapidamente ao aumento no atraso, significando que pacotes sucessivos chegam sem retransmissão.

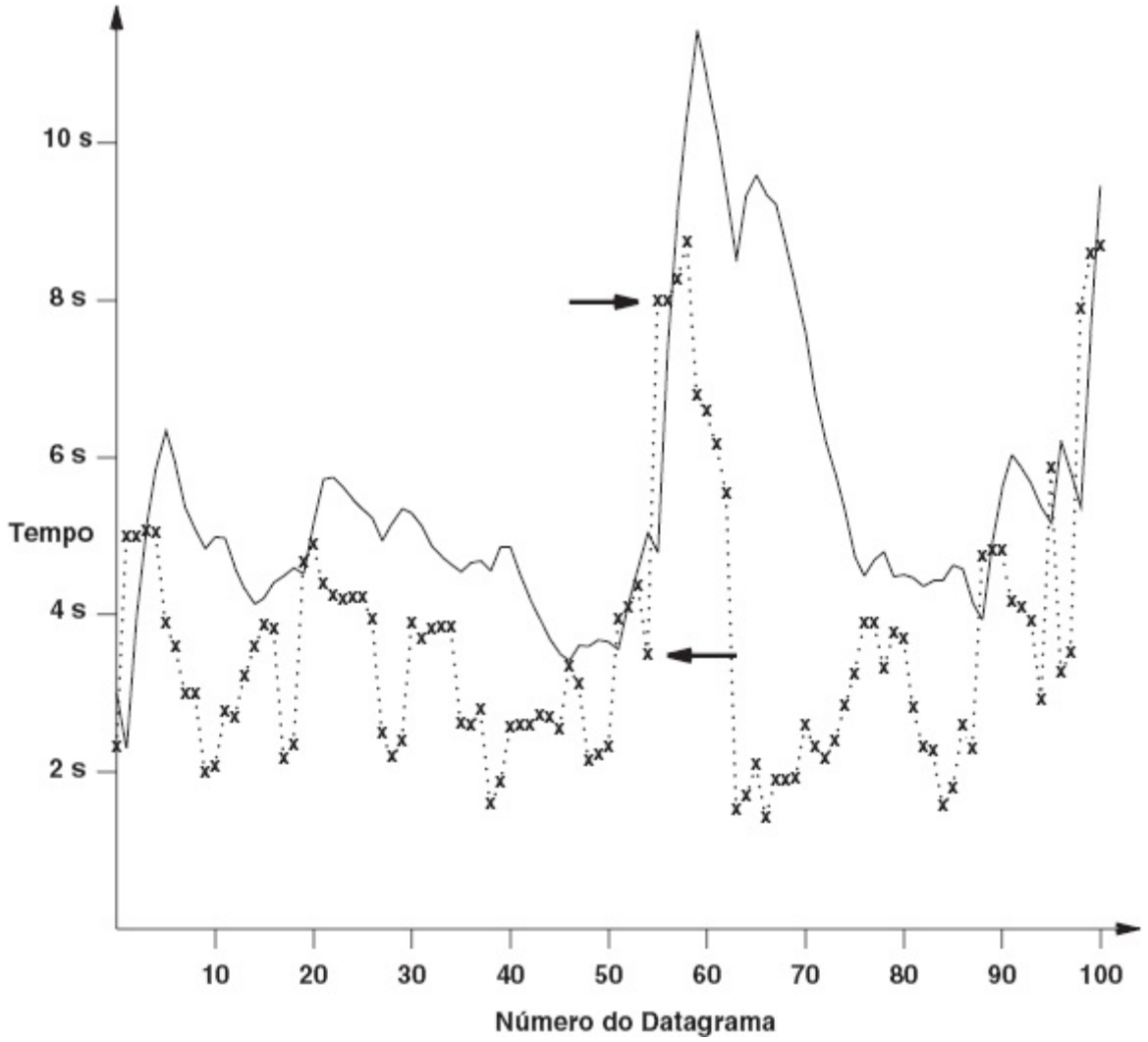


Figura 11.13 O valor do timer de retransmissão do TCP para os dados extremos da Figura 11.11. As setas marcam dois datagramas sucessivos em que o atraso dobra.

11.19 Resposta ao congestionamento

Pode parecer que o TCP poderia ser projetado considerando a interação entre as duas extremidades de uma conexão e os atrasos de comunicação entre essas extremidades. Porém, na prática, o TCP também precisa reagir ao *congestionamento* em uma internet. O congestionamento é uma condição de atraso severo causado por uma sobrecarga de datagramas em um ou mais pontos de comutação (por exemplo, nos roteadores). Quando ocorre o congestionamento, os atrasos aumentam e o roteador começa a enfileirar datagramas até que possa encaminhá-los. Temos de lembrar que cada roteador possui capacidade de armazenamento finita e que os datagramas competem por esse armazenamento (ou seja, em uma internet

baseada em datagrama, não existe pré-alocação de recursos para conexões TCP individuais). No pior caso, o número total de datagramas chegando ao roteador congestionado cresce até que o roteador alcance a capacidade e comece a descartá-los.

As extremidades normalmente não sabem os detalhes de onde ocorreu o congestionamento ou por quê. Para elas, o congestionamento simplesmente significa maior atraso. Infelizmente, a maioria dos protocolos de transporte utiliza timeout e retransmissão, de modo que eles respondem ao maior atraso pela retransmissão de datagramas. As retransmissões agravam o congestionamento, em vez de aliviá-lo. Se não for verificado, o tráfego aumentado produzirá maior atraso, levando a um tráfego mais intenso, e assim por diante, até que a rede se torne inutilizável. A condição é conhecida como *colapso de congestionamento*.

Para evitar o colapso de congestionamento, o TCP precisa reduzir as taxas de transmissão quando o congestionamento ocorre. Os roteadores observam os tamanhos de fila e utilizam técnicas como extinção de origem ICMP para informar aos hosts que ocorreu congestionamento, mas protocolos de transporte como TCP podem ajudar a evitar congestionamento, reduzindo as taxas de transmissão automaticamente sempre que ocorrem atrasos. É natural que os algoritmos para evitar congestionamento precisem ser construídos cuidadosamente, pois até mesmo sob condições de operação normais, uma internet exibirá grande variação nos atrasos de ida e volta.

Para evitar congestionamento, o padrão TCP agora recomenda o uso de duas técnicas: *partida lenta* e *diminuição multiplicativa*. Elas estão relacionadas e podem ser implementadas com facilidade. Dizemos que, para cada conexão, o TCP precisa se lembrar do tamanho da janela do receptor (ou seja, o tamanho do buffer anunciado nas confirmações). Para controlar o congestionamento, o TCP mantém um segundo limite, chamado *limite de janela de congestionamento*, ou *janela de congestionamento*, que usa para restringir o fluxo de dados para menos do que o tamanho do buffer do receptor quando ocorrer congestionamento. Ou seja, a qualquer momento, o TCP atua como se o tamanho da janela fosse:

```
Janela_permitida = min(anúncio_receptor, janela_congestionamento)
```

No estado constante em uma conexão não congestionada, a janela de congestionamento tem o mesmo tamanho da janela do receptor. A redução da janela de congestionamento reduz o tráfego que o TCP injetará na conexão. Para estimar o tamanho da janela de congestionamento, o TCP

considera que a maior parte da perda do datagrama vem do congestionamento e, então, usa a estratégia a seguir.

> *Impedimento de congestionamento por diminuição multiplicativa: na perda de um segmento, reduz a janela de congestionamento pela metade (mas nunca reduz a janela a menos de um segmento). Ao transmitir segmentos que permanecem na janela permitida, recua o timer de retransmissão exponencialmente.*

Como o TCP reduz a janela de congestionamento pela metade para *cada* perda, ele diminui a janela exponencialmente se a perda continuar. Em outras palavras, se o congestionamento for provável, o TCP reduz o volume de tráfego e a taxa de retransmissão exponencialmente. Se a perda continuar, o TCP por fim limita a transmissão a um único datagrama e continua a dobrar os valores de timeout antes de retransmitir. A ideia é oferecer redução de tráfego rápida e significativa para permitir que os roteadores tenham tempo suficiente para limpar os datagramas já em suas filas.

Como o TCP pode se recuperar quando o congestionamento termina? Você poderia suspeitar que o TCP deve reverter a diminuição multiplicativa e dobrar a janela de congestionamento quando o tráfego começar a fluir novamente. Porém, isso produz um sistema instável, que oscila muito entre nenhum tráfego e congestionamento. Em vez disso, o TCP utiliza uma técnica chamada *partida lenta** para aumentar a transmissão.

> *Recuperação por partida lenta (aditiva): sempre que o tráfego estiver se iniciando em uma nova conexão ou houver um aumento de tráfego após um período de congestionamento, inicie a janela de congestionamento com tamanho de um único segmento e aumente a janela de congestionamento em um segmento toda vez que uma confirmação chegar.*

A partida lenta evita a inundação da internet subjacente com tráfego

adicional imediatamente após o congestionamento terminar ou quando novas conexões de repente iniciarem.

O termo *partida lenta* pode ser um erro, pois, sob condições ideais, a partida não é muito lenta. O TCP inicializa a janela de congestionamento em *1*, envia um segmento inicial e espera. Quando a confirmação chega, ele aumenta a janela de congestionamento para *2*, envia dois segmentos e espera. Quando as duas confirmações chegam, cada uma delas aumenta a janela de congestionamento em *1*, de modo que o TCP pode enviar *4* segmentos. As confirmações desses segmentos aumentarão a janela de congestionamento para *8*. Dentro de quatro tempos de ida e volta, o TCP pode enviar *16* segmentos, normalmente o suficiente para atingir o limite de janela do receptor. Até mesmo para janelas extremamente grandes, são necessárias apenas $\log_2 N$ viagens de ida e volta antes que o TCP possa enviar N segmentos.

Para evitar o aumento do tamanho da janela muito rapidamente e congestionamento adicional, o TCP acrescenta uma restrição adicional. Quando a janela de congestionamento atinge a metade do seu tamanho original antes do congestionamento, o TCP entra em uma fase de *impedimento de congestionamento* e diminui a velocidade do incremento. Durante o impedimento do congestionamento, ele aumenta a janela de congestionamento em *1* somente se todos os segmentos na janela tiverem sido confirmados. A técnica geral é conhecida como aumento aditivo diminuição multiplicativa, *Additive Increase Multiplicative Decrease* (*AIMD*).

Juntos, partida lenta, diminuição multiplicativa, aumento aditivo, medição da variação e recuo de timer exponencial melhoram bastante o desempenho do TCP, sem acrescentar qualquer overhead computacional significativo ao protocolo. As versões do TCP que utilizam essas técnicas melhoraram o desempenho das versões anteriores significativamente.

11.20 Recuperação rápida e outras modificações de resposta

Pequenas modificações foram feitas no TCP ao longo dos anos. A versão antiga do TCP, às vezes chamada de *Tahoe*, usava o esquema de

retransmissão descrito anteriormente, esperando um timer expirar antes de retransmitir. Em 1990, apareceu a versão *Reno* do TCP, que introduziu várias mudanças, incluindo uma heurística conhecida como *recuperação rápida* ou *retransmissão rápida*, que possui rendimento mais alto nos casos em que ocorre apenas a perda ocasional. Seguindo a versão *Reno*, pesquisadores exploraram a versão *Vegas*.

O truque usado na recuperação rápida surge do esquema de confirmação acumulativa do TCP: a perda de um único segmento significa que a chegada de segmentos subsequentes fará com que o receptor gere um ACK para o ponto no fluxo onde inicia o segmento que faltava. Do ponto de vista de um emissor, um pacote perdido significa que múltiplas confirmações vão chegar carregando cada uma o mesmo número sequencial. A heurística de retransmissão rápida usa uma série de três *confirmações duplicadas* (ou seja, uma original mais três cópias absolutamente idênticas) para disparar uma retransmissão sem esperar que o timer expire.

No caso em que somente um segmento se perde, esperar que o segmento retransmitido seja confirmado também reduz o rendimento. Portanto, para manter um rendimento mais alto, a heurística de retransmissão rápida continua a enviar dados da janela enquanto espera a confirmação do segmento retransmitido. Além do mais, a janela de congestionamento é inflada artificialmente: a janela de congestionamento é dividida ao meio para a retransmissão, mas depois é aumentada por um segmento de tamanho máximo para cada ACK duplicado que chegou anteriormente ou que chega depois que a retransmissão ocorre. Como resultado, enquanto ocorre a retransmissão rápida, o TCP mantém muitos segmentos "em voo" entre o emissor e o receptor.

Outra otimização da heurística de retransmissão rápida foi incorporada em uma versão posterior do TCP, conhecida como versão *NewReno*. A otimização trata de um caso no qual dois segmentos são perdidos dentro de uma única janela. Essencialmente, quando ocorre a retransmissão rápida, o NewReno registra informações sobre a janela atual e retransmite conforme descrevemos. Quando o ACK chega, para o segmento retransmitido, existem duas possibilidades: o ACK especifica o número de sequência no final da janela (quando o segmento retransmitido foi o único segmento que faltava na janela) ou o ACK especifica um número de sequência mais alto que o segmento que falta, mas menor que o final da janela (quando um segundo segmento da janela também foi perdido). No segundo caso, o

NewReno prossegue para retransmitir o segundo segmento que faltou.

Pequenas modificações no esquema AIMD* também foram propostas e usadas em outras versões do TCP. Para entender, considere como o AIMD muda a janela de congestionamento do emissor, w, em resposta à perda de segmento ou à chegada de uma confirmação:

$$w \leftarrow w - aw \text{ quando a perda é detectada}$$

$$w \leftarrow w \frac{b}{w} \text{ quando um ACK chega}$$

No esquema original, a é 0,5 e b é 1. Pensando em protocolos como STCP, que são usados em sensores de rede, os pesquisadores propuseram a definição de a como 0,125 e b como 0,01, para impedir que a janela de congestionamento oscile e aumentar ligeiramente o rendimento. Outras propostas de modificação (por exemplo, um protocolo conhecido como HSTCP) sugerem tornar a e b funções de w (ou seja, $a(w)$ e $b(w)$). Finalmente, as propostas para controle de congestionamento do TCP, como Vegas e FAST, utilizam o RTT aumentado como uma medida de congestionamento em vez de perda de pacotes, e definem o tamanho da janela de congestionamento como uma função do RTT medido. Normalmente, as modificações só levam a melhorias de desempenho em casos especiais (por exemplo, redes com largura de banda alta e taxas de perda baixas); o controle de congestionamento AIMD original (Reno) é usado para outros casos.

Uma proposta final, relacionada ao controle de congestionamento TCP, refere-se ao UDP. Observe que, embora o TCP reduza a transmissão quando ocorre o congestionamento, o UDP não faz isso, o que significa que, enquanto os fluxos do TCP continuam a recuar, os fluxos do UDP consomem mais da largura de banda. Foi proposta uma solução conhecida como TCP *Friendly Rate Control* (*TFRC*). O TFRC tenta simular o comportamento do TCP fazendo com que um receptor informe a perda de pacotes de volta ao emissor e também que o emissor use os dados de perda relatados para calcular uma velocidade com que o UDP deve ser enviado; o TFRC só tem sido adotado em casos especiais.

11.21 Mecanismos de feedback explícitos (Sack e Ecn)

A maioria das versões do TCP utiliza técnicas *implícitas* para detectar a perda e o congestionamento. Ou seja, o TCP usa timeout e ACKs

duplicados para detectar perda, e mudanças nos tempos de ida e volta para detectar congestionamento. Os pesquisadores têm observado que ligeiras melhorias são possíveis se o TCP incluir mecanismos que ofereçam essas informações *explicitamente*. As próximas duas seções descrevem duas técnicas explícitas que foram propostas.

11.21.1 Confirmação seletiva (Selective Acknowledgement – SACK)

A alternativa ao mecanismo de confirmação acumulativo do TCP é conhecida como mecanismo de *confirmação seletiva*. Essencialmente, as confirmações seletivas permitem que um receptor especifique exatamente quais dados foram recebidos e quais ainda estão faltando. A principal vantagem das confirmações seletivas surge em situações em que ocorre a perda ocasional: as confirmações seletivas permitem que um emissor saiba exatamente quais segmentos retransmitir.

O mecanismo de Selective ACKnowledgement (SACK) proposto para o TCP não substitui completamente o mecanismo de confirmação cumulativa, nem é obrigatório. Em vez disso, o TCP inclui duas opções para o SACK. A primeira opção é usada quando a conexão é estabelecida, para permitir que um emissor especifique que o SACK é permitido. A segunda é usada por um receptor para incluir, em cada confirmação, informações sobre blocos específicos de dados que foram recebidos. As informações para cada bloco incluem o primeiro número de sequência em um bloco (chamado *borda esquerda*) e o número de sequência imediatamente além do bloco (chamado *borda direita*). Como o tamanho máximo de um cabeçalho de segmento é fixo, uma confirmação pode conter no máximo quatro blocos SACK. É interessante que os documentos SACK não especificam exatamente como um emissor responde ao SACK; a maioria das implementações retransmite todos os blocos que faltam.

11.21.2 Notificação explícita de congestionamento

Uma segunda técnica proposta para evitar a medição implícita serve para lidar com o congestionamento na rede. Conhecido como *Explicit Congestion Notification* (*ECN*), o mecanismo requer que os roteadores por uma internet notifiquem o TCP quando ocorre um congestionamento. O mecanismo é conceitualmente simples: quando um segmento TCP passa pela internet, os roteadores ao longo do caminho utilizam um par de bits no

cabeçalho IP para registrar o congestionamento. Assim, quando um segmento chega, o receptor sabe se o segmento experimentou congestionamento em algum ponto. Infelizmente, o emissor, e não o receptor, precisa descobrir o congestionamento. Portanto, o receptor usa o próximo ACK para informar ao emissor que ocorreu o congestionamento. O emissor, então, responde reduzindo sua janela de congestionamento.

O ECN usa dois bits no cabeçalho IP para permitir que os roteadores registrem o congestionamento, e usa dois bits no cabeçalho TCP (tomados da área reservada) para permitir que o TCP emissor e receptor se comuniquem. Um dos bits de cabeçalho do TCP é usado por um receptor para enviar informações de congestionamento a um emissor; o outro bit permite que um emissor informe ao receptor que a notificação de congestionamento foi recebida. Os bits no cabeçalho IP são tomados dos bits não usados no campo TIPO DE SERVIÇO. Um roteador pode decidir definir qualquer bit para especificar que o congestionamento ocorreu (dois bits são usados para tornar o mecanismo mais robusto).

11.22 Congestionamento, descarte de cauda e TCP

Dissemos que os protocolos de comunicação são divididos em camadas para permitir que os projetistas focalizem um único problema de cada vez. A separação da funcionalidade em camadas é tanto necessária quanto útil – significa que uma camada pode ser trocada sem afetar outras e que as camadas operam isoladamente. Por exemplo, por operar de fim a fim, o TCP permanece inalterado quando o caminho entre as extremidades muda (exemplificando, as rotas mudam ou roteadores de redes adicionais são acrescentados). Porém, o isolamento das camadas restringe a comunicação entre estas. Em particular, embora o TCP na origem interaja com o TCP no destino final, ele não pode interagir com elementos da camada inferior ao longo do caminho.* Assim, nem o TCP de envio nem o de recepção recebem relatórios sobre as condições na rede, e nenhuma extremidade informa às camadas inferiores ao longo do caminho antes de transferir dados.

Os pesquisadores observaram que a falta de comunicação entre as camadas significa que a escolha da política ou implementação em uma camada pode ter um efeito dramático sobre o desempenho de camadas mais altas. No caso do TCP, as políticas que os roteadores usam para lidar com datagramas podem ter um efeito significativo sobre o desempenho de uma única conexão TCP e o rendimento agregado de todas as conexões. Por exemplo, se um roteador atrasar alguns datagramas mais do que outros,** o TCP recuará seu timer de retransmissão. Se o atraso ultrapassar o timeout

de retransmissão, o TCP considerará que o congestionamento ocorreu. Assim, embora cada camada seja definida independentemente, os pesquisadores tentam criar mecanismos e implementações que funcionam bem com protocolos em outras camadas.

A interação mais importante entre as políticas de implementação do IP e o TCP ocorre quando um roteador se torna defasado e descarta datagramas. Como um roteador coloca cada datagrama que recebe em uma fila na memória até que possa ser processado, a política focaliza o gerenciamento da fila. Quando os datagramas chegam mais rápido do que podem ser encaminhados, a fila cresce; quando os datagramas chegam mais lentamente do que podem ser encaminhados, a fila encurta. Porém, como a memória é finita, a fila não pode crescer sem limites. Os primeiros roteadores usavam uma política de *descarte de cauda* para gerenciar o estouro da fila.

> *Política de descarte de cauda para roteadores: se uma fila de pacotes estiver cheia quando um datagrama tiver de ser colocado nela, deve-se descartá-lo.*

O nome *descarte de cauda* surge do efeito da política sobre uma sequência de chegada dos datagramas. Quando a fila se enche, o roteador começa a descartar todos os datagramas adicionais. Ou seja, o roteador descarta a "cauda" (tail) da sequência.

O descarte de cauda tem um efeito interessante sobre o TCP. No caso simples em que os datagramas atravessando um roteador transportam segmentos de uma única conexão TCP, a perda faz com que o TCP entre na partida lenta, que reduz o rendimento até que o TCP comece a receber ACKs e aumente a janela de congestionamento. Porém, pode ocorrer um problema mais severo, quando os datagramas atravessando um roteador transportam segmentos de muitas conexões TCP, pois o descarte de cauda pode causar o sincronismo global. Para ver por que motivo, observe que os datagramas normalmente são multiplexados, com datagramas sucessivos chegando de uma origem diferente. Assim, uma política de descarte de cauda torna provável que o roteador descarte um segmento de N conexões em vez de N segmentos de uma conexão. A perda simultânea faz com que todas as N instâncias do TCP entrem em partida lenta ao mesmo tempo.***

11.23 Random Early Detection (RED)

Como um roteador pode evitar o sincronismo global? A resposta está em um esquema inteligente que evita o descarte de cauda sempre que possível. Chamado de *Random Early Detection, Random Early Drop*, ou *Random Early Discard*, o esquema é mais conhecido por seu acrônimo, RED.

A ideia geral por trás do RED está na randomização: em vez de esperar até que a fila encha completamente, um roteador monitora o seu tamanho e, quando ela começa a encher, escolhe datagramas ao acaso para descartar.

Um roteador que implementa o RED executa o algoritmo em cada fila (por exemplo, cada conexão de rede). Para simplificar nossa descrição, vamos apenas discutir uma única fila e assumir que o leitor percebe que a mesma técnica deve ser aplicada a outras filas.

Um roteador usa dois valores de limite para marcar posições na fila: $T_{\min}$ e $T_{\max}$. A operação geral do RED pode ser descrita a seguir por três regras que determinam a disposição de um datagrama que deve ser colocado na fila.

- Se a fila atualmente contém menos de $T_{\min}$ datagramas, acrescente o novo datagrama a ela.
- Se a fila tiver mais do que $T_{\max}$ datagramas, descarte o novo datagrama.
- Se a fila tiver entre $T_{\min}$ e $T_{\max}$ datagramas, descarte aleatoriamente o datagrama de acordo com uma probabilidade, p, que depende do tamanho atual da fila.

A aleatoriedade do RED significa que, em vez de esperar até que a fila estoure e depois passar muitas conexões TCP para partida lenta, um roteador lenta e aleatoriamente descarta datagramas quando o congestionamento aumenta. Podemos resumir desta forma a seguir.

> *A política RED para roteadores: se a fila de entrada estiver cheia, quando um datagrama chegar, descartar o datagrama; se a fila de entrada estiver abaixo de um limite mínimo, adicionar o datagrama para a fila; caso contrário, descartar o datagrama com uma probabilidade que depende do tamanho da fila.*

A chave para fazer o RED funcionar bem está na escolha dos patamares $T_{\min}$ e $T_{\max}$, e a probabilidade de descarte p. $T_{\min}$ precisa ser grande o suficiente para garantir que a fila tenha rendimento alto o suficiente. Por exemplo, se a fila está conectada a um link, ela deve dirigir a rede para alta utilização. Além do mais, como o RED opera como descarte de cauda quando o tamanho da fila ultrapassa $T_{\max}$, o valor precisa ser maior que $T_{\min}$ por mais do que o aumento típico no tamanho da fila durante um tempo de ida e volta do TCP (por exemplo, defina $T_{\max}$ pelo menos duas vezes o tamanho de $T_{\min}$). Caso contrário, o RED pode causar as mesmas oscilações globais que o descarte de cauda (por exemplo, $T_{\min}$ pode ser fixado a metade de $T_{\max}$).

O cálculo da probabilidade de descarte, p, é o aspecto mais complexo do RED. Em vez de usar uma constante, um novo valor de p é calculado para cada datagrama; o valor depende do relacionamento entre o tamanho atual da fila e os patamares. Para entender o esquema, observe que todo o processamento do RED pode ser visto de forma probabilística. Quando o tamanho da fila for inferior a $T_{\min}$, o RED não descartará quaisquer datagramas, tornando a probabilidade de descarte *zero*. De modo semelhante, quando o tamanho da fila for maior que $T_{\max}$, o RED descartará todos os datagramas, tornando a probabilidade de descarte *1*. Para valores intermediários do tamanho da fila (ou seja, aqueles entre $T_{\min}$ e $T_{\max}$), a probabilidade pode variar de *0* até *1* linearmente.

Embora o esquema linear forme a base do cálculo da probabilidade do RED, é preciso fazer uma mudança para evitar reação demasiada. A necessidade de mudança surge porque o tráfego da rede é em rajada, o que resulta em rápidas flutuações na fila de um roteador. Se o RED usasse um esquema linear simples, outros datagramas em cada rajada receberiam uma alta probabilidade de ser descartados (pois chegam quando a fila possui mais entradas). Porém, um roteador não deve descartar datagramas desnecessariamente, pois isso tem um impacto negativo sobre o rendimento do TCP. Assim, se uma rajada for curta, não é sensato descartar datagramas, pois a fila nunca estourará. Naturalmente, o RED não pode adiar o descarte

indefinidamente, porque uma rajada a longo prazo estourará a fila, resultando em uma política de descarte de trailer que tem o potencial de causar problemas de sincronismo global.

Como o RED pode atribuir uma probabilidade de descarte mais alta enquanto a fila se enche sem descartar datagramas de cada rajada? A resposta está em uma técnica que vem do TCP: em vez de usar o tamanho de fila real a qualquer instante, o RED calcula um tamanho de fila com média ponderada, *avg*, e usa o tamanho médio para determinar a probabilidade. O valor de *avg* é uma média ponderada exponencial, atualizada toda vez que um datagrama chega, de acordo com a equação

$$\text{avg} = (1 - \delta) \times \text{Antigo\_avg} + \delta \times \text{Tamanho\_atual\_da\_fila}$$

na qual δ indica um valor entre *0* e *1*. Se δ for pequeno o suficiente, a média acompanhará tendências a longo prazo, mas permanecerá imune a rajadas curtas.*

Além das equações que determinam δ, o RED contém outros detalhes que não mencionamos. Por exemplo, os cálculos do RED podem se tornar extremamente eficientes pela escolha de constantes como potências de dois e pelo uso da aritmética de inteiros. Outro detalhe importante diz respeito à medição do tamanho da fila, que afeta o cálculo do RED e seu efeito geral sobre o TCP. Em particular, como o tempo exigido para encaminhar um datagrama é proporcional ao seu tamanho, faz sentido medir a fila em octetos, em vez de datagramas; isso exige apenas pequenas mudanças nas equações para p e δ. A medição do tamanho da fila em octetos afeta o tipo de tráfego descartado, pois torna a probabilidade de descarte proporcional à quantidade de dados que um emissor coloca no fluxo, em vez do número de segmentos. Pequenos datagramas (por exemplo, aqueles que transportam tráfego de login remoto ou requisições aos servidores) possuem probabilidade menor de serem descartados do que datagramas maiores (por exemplo, aqueles que transportam tráfego de transferência de arquivo). Uma consequência positiva do uso do tamanho é que, quando as confirmações trafegam por um caminho congestionado, elas têm uma probabilidade mais baixa de serem descartadas. Como resultado, se um segmento de dados (grande) chegar, o TCP emissor receberá o ACK e evitará retransmissão desnecessária.

A análise e as simulações mostram que o RED funciona bem. Ele trata do congestionamento, evita o sincronismo que resulta do descarte da trailer e permite rajadas curtas sem descartar datagramas de forma desnecessária.

Consequentemente, O IETF agora recomenda que os roteadores implementem o RED.

11.24 Estabelecendo uma conexão TCP

Para estabelecer uma conexão, o TCP usa um *handshake de três vias.* Ou seja, são trocadas três mensagens que permitem que cada lado concorde em formar uma conexão e saiba que o outro lado concordou. O primeiro segmento de um handshake pode ser identificado porque possui o bit SYN* marcado no campo de código. A segunda mensagem possui os bits SYN e ACK marcados, indicando que ele confirma o primeiro segmento SYN e continua com o handshake. A mensagem de handshake final é apenas uma confirmação, sendo simplesmente usada para informar ao destino que ambos os lados concordam que uma conexão foi estabelecida.

Normalmente, o TCP em uma máquina espera passivamente pelo handshake, e o TCP em outra máquina o inicia. Porém, o handshake é cuidadosamente projetado para funcionar mesmo que as duas máquinas tentem iniciar uma conexão simultaneamente. Assim, uma conexão pode ser estabelecida de qualquer extremidade ou das duas extremidades simultaneamente. Quando a conexão tiver sido estabelecida, os dados podem fluir nas duas direções igualmente bem (ou seja, a conexão é simétrica). Não existe mestre ou escravo, e o lado que iniciou a conexão não tem habilidades ou privilégios especiais. No caso mais simples, procede como a Figura 11.14 ilustra.

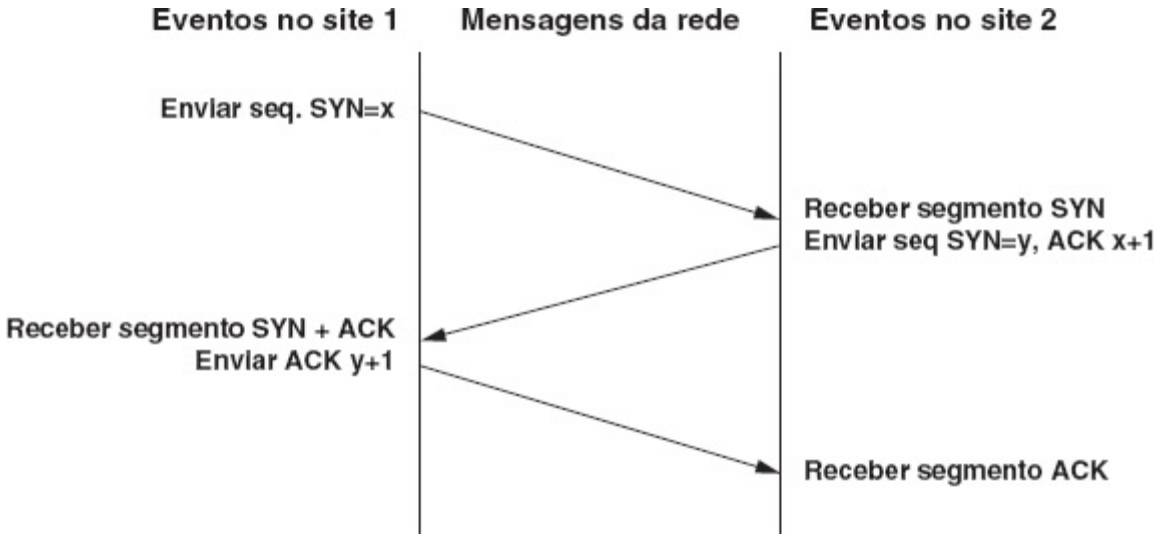


Figura 11.14 A sequência de mensagens em um handshake de três vias. O tempo segue para baixo na página; as linhas diagonais representam segmentos enviados entre os sites.

Pode parecer que a troca de duas mensagens seria suficiente para estabelecer uma conexão. No entanto, o handshake de três vias é necessário e suficiente para a sincronização correta entre as duas extremidades da conexão devido à semântica de entrega da internet. Para entender a razão pela qual o estabelecimento da conexão é difícil, lembre-se de que o TCP usa um serviço de entrega de pacotes não confiável. Portanto, as mensagens podem ser perdidas, atrasadas, duplicadas ou entregues fora de ordem. Para acomodar a perda, o TCP deve retransmitir solicitações. No entanto, podem surgir problemas se a demora excessiva provocar retransmissão, o que significa que tanto a cópia original como a retransmitida chegam enquanto a conexão está sendo estabelecida. Pedidos retransmitidos também podem ser atrasados até que uma conexão seja estabelecida, usada e finalizada! O handshake de três vias e regras que impedem o reinício de uma conexão depois de ter sido finalizada são cuidadosamente projetados para compensar todas as situações possíveis.

11.25 Números sequenciais iniciais

Um handshake de três vias realiza duas funções importantes. Ele garante que os dois lados estejam prontos para transferir dados (e que saibam que ambos estão prontos) e permite que os dois lados concordem sobre números de sequência iniciais. Os números de sequência são enviados e confirmados durante o handshake. Cada máquina precisa escolher um número de sequência inicial aleatoriamente, que usará para identificar os bytes no fluxo que está enviado. Os números de sequência não podem começar sempre com o mesmo valor. Em particular, o TCP não pode simplesmente escolher a sequência *1* toda vez que criar uma conexão (um dos exercícios examina problemas que podem surgir se isso acontecer). Naturalmente, é importante que os dois lados concordem sobre um número inicial, de modo que os números de octeto usados nas confirmações combinem com aqueles usados nos segmentos de dados.

Para ver como as máquinas podem combinar sobre números de sequência para dois fluxos após somente três mensagens, lembre-se de que cada segmento contém um campo de número de sequência e um campo de confirmação. A máquina que inicia um handshake, que chamaremos de *A*, passa seu número de sequência inicial, *x*, no campo de sequência do primeiro segmento SYN do handshake de três vias. A segunda máquina, *B*, recebe o SYN, registra o número de sequência e responde enviando seu número de sequência inicial no campo de sequência, além de uma

confirmação que especifica que *B* espera o octeto x *+1*. Na mensagem final do handshake, *A* "confirma" receber de *B* todos os octetos até *y*. Em todos os casos, as confirmações seguem a convenção de usar o número do próximo octeto esperado.

Descrevemos como o TCP normalmente executa o handshake de três vias, trocando segmentos que contêm uma quantidade mínima de informações. Devido ao projeto do protocolo, é possível enviar dados junto com os números de sequência iniciais nos segmentos de handshake. Nesses casos, o TCP precisa manter os dados até que o handshake termine. Quando uma conexão for estabelecida, o TCP pode liberar dados sendo mantidos e entregá-los rapidamente a um programa aplicativo que esteja aguardando. O leitor deverá consultar a especificação do protocolo para ver os detalhes.

11.26 Fechando uma conexão TCP

Dois programas que usam o TCP para se comunicar podem terminar a conversação de modo controlado usando a operação *close*. Mais uma vez, é importante que ambos os lados concordem em fechar a conexão e também saibam que ela está fechada. Desse modo, o TCP usa o handshake de três vias para fechar a conexão. Para entender o que o handshake usa para fechar a conexão, lembre que as conexões TCP são full duplex e que as vemos como contendo duas transferências de fluxo independentes, cada uma *em uma direção*. Quando um programa aplicativo diz ao TCP que não possui mais dados para enviar, o TCP fecha a conexão em uma direção. Para fechar sua metade de uma conexão, o TCP de envio termina de transmitir os dados restantes, espera que o receptor confirme e depois envia um segmento com o bit *FIN* marcado.* Com o recebimento de um FIN, o TCP envia uma confirmação e então informa ao aplicativo que o outro lado encerrou o envio de dados. Os detalhes dependem do sistema operacional, mas a maior parte dos sistemas usa o mecanismo "fim de arquivo" ("end-of-file").

Uma vez que uma conexão tiver sido fechada em determinada direção, o TCP se recusa a aceitar mais dados para essa direção. Enquanto isso, os dados podem continuar a fluir na direção oposta até que o emissor encerre. Naturalmente, uma extremidade TCP que ainda está recebendo dados deve enviar confirmações, mesmo que a transmissão de dados em reverso tenha terminado. Quando as duas direções tiverem sido fechadas, o TCP em cada extremidade exclui seu registro da conexão.

Os detalhes do fechamento de uma conexão são ainda mais sutis do que o sugerido anteriormente, pois o TCP usa um handshake de três vias modificado para fechar uma conexão. A Figura 11.15 ilustra a mensagem que é trocada para o caso típico em que toda a comunicação terminou e a conexão está fechada em ambas as direções.

A diferença entre os handshakes de três vias usados para estabelecer e fechar conexões ocorre depois que uma máquina recebe o segmento FIN inicial. Em vez de gerar um segundo segmento FIN imediatamente, o TCP envia uma confirmação e depois informa à aplicação quanto à requisição para fechar. Informar ao aplicativo sobre a requisição e obter uma resposta pode levar um tempo considerável (por exemplo, isso pode envolver interação humana). A confirmação impede a retransmissão do segmento FIN inicial durante a espera. Finalmente, quando o programa aplicativo instrui o TCP a encerrar a conexão completamente, o TCP envia o segundo segmento FIN, e o site original responde com a terceira mensagem, um ACK.

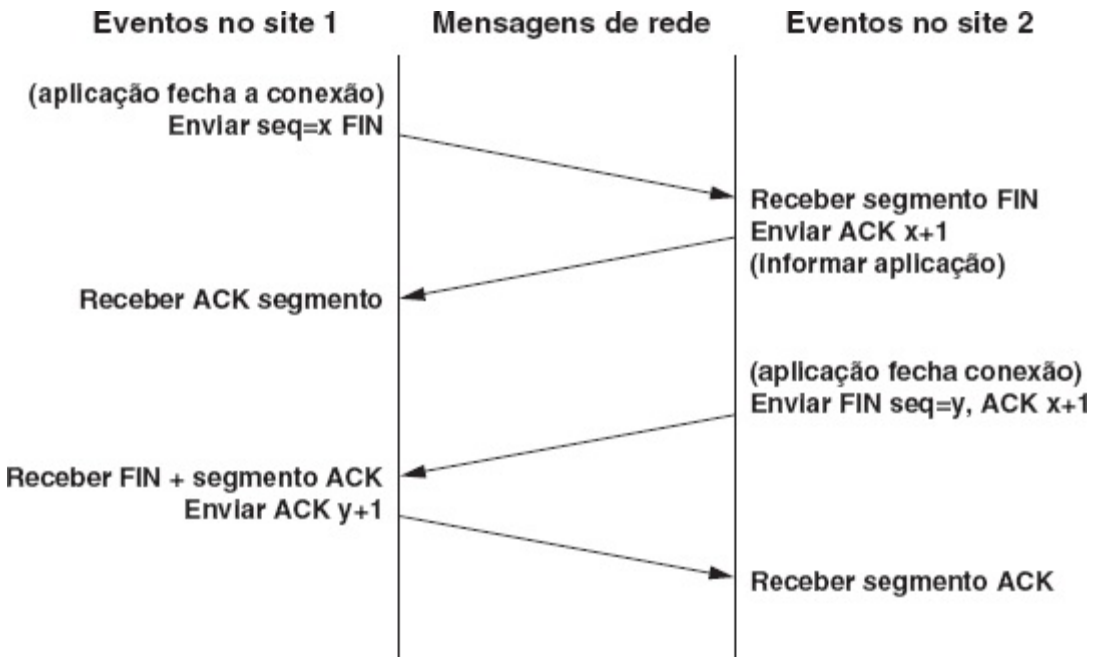


Figura 11.15 O handshake de três vias usado para fechar uma conexão com um ACK extra enviado imediatamente após receber um FIN.

11.27 Reiniciando uma conexão TCP

Normalmente, um programa de aplicação usa a operação *fechar* (close) para encerrar uma conexão ao final do envio de dados. Assim, o fechamento de conexões é considerado uma parte normal de utilização, análogo ao

fechamento de arquivos. Dizemos que a ligação terminou *corretamente* (*gracefully*). No entanto, por vezes, surgem condições anormais que forçam um aplicativo ou o software de rede a quebrar uma conexão sem um desligamento correto. O TCP fornece um recurso de reinício para lidar com desconexões anormais.

Para reiniciar uma conexão, um lado inicia o encerramento enviando um segmento com o bit *RST* (*RESET*) marcado no campo *CODE*. O outro lado responde imediatamente a um segmento reset abortando a ligação. Quando ocorre um re-set, o TCP o informa a qualquer aplicação local que estava usando a conexão. Note-se que um comando reset ocorre imediatamente e não pode ser desfeito. Nós pensamos nisso como um aborto instantâneo que encerra a transferência em ambas as direções e libera os recursos, tais como buffers.

11.28 Máquina de estado TCP

Assim como a maioria dos protocolos, a operação do TCP pode ser mais bem explicada com um modelo teórico chamado *máquina de estado finito* (*finite state machine*). A Figura 11.16 mostra a máquina de estado finito TCP, com círculos representando os estados e setas representando transições entre elas.

Na figura, o rótulo em cada estado de transição mostra o que provoca a transição e que o TCP envia, quando ele realiza a transição. Por exemplo, o software de TCP em cada extremidade começa no estado FECHADO. Os programas de aplicação devem emitir ambos, um comando *passive open* para aguardar uma conexão a partir de outra máquina, ou um comando *active open* para iniciar uma conexão. Um comando active open força uma transição do estado FECHADO para o estado SYN ENVIADO. Quando segue a transição para SYN ENVIADO, O TCP emite um segmento SYN. Quando o outro lado retorna um segmento com o SYN e o bit ACK definido, o TCP passa para o estado ESTABELECIDO, emite um ACK e começa a transferência de dados.

O estado ESPERA TEMPORIZADA revela como o TCP trata de alguns dos problemas que acontecem com a remessa não confiável. O TCP mantém uma noção de *Tempo de Vida Máximo do Segmento* (*MSL – Maximum Segment Lifetime*), o tempo máximo que um segmento antigo pode permanecer vivo em uma internet. Para evitar que os segmentos de uma

conexão anterior interfiram com um atual, o TCP passa para o estado ESPERA TEMPORIZADA depois de fechar uma conexão. Ele permanece nesse estado pelo dobro do tempo de vida máximo do segmento, antes de excluir seu registro da conexão. Se quaisquer segmentos duplicados chegarem para a conexão durante o intervalo de timeout, o TCP os rejeitará. Porém, para lidar com casos em que a última confirmação foi perdida, o TCP confirma os segmentos válidos e reinicia o timer. Como o timer permite que o TCP distinga conexões antigas das novas, ele impede que o TCP envie um reset em resposta aos segmentos atrasados vindos de uma conexão antiga (ou seja, se a outra extremidade retransmitir um segmento FIN).

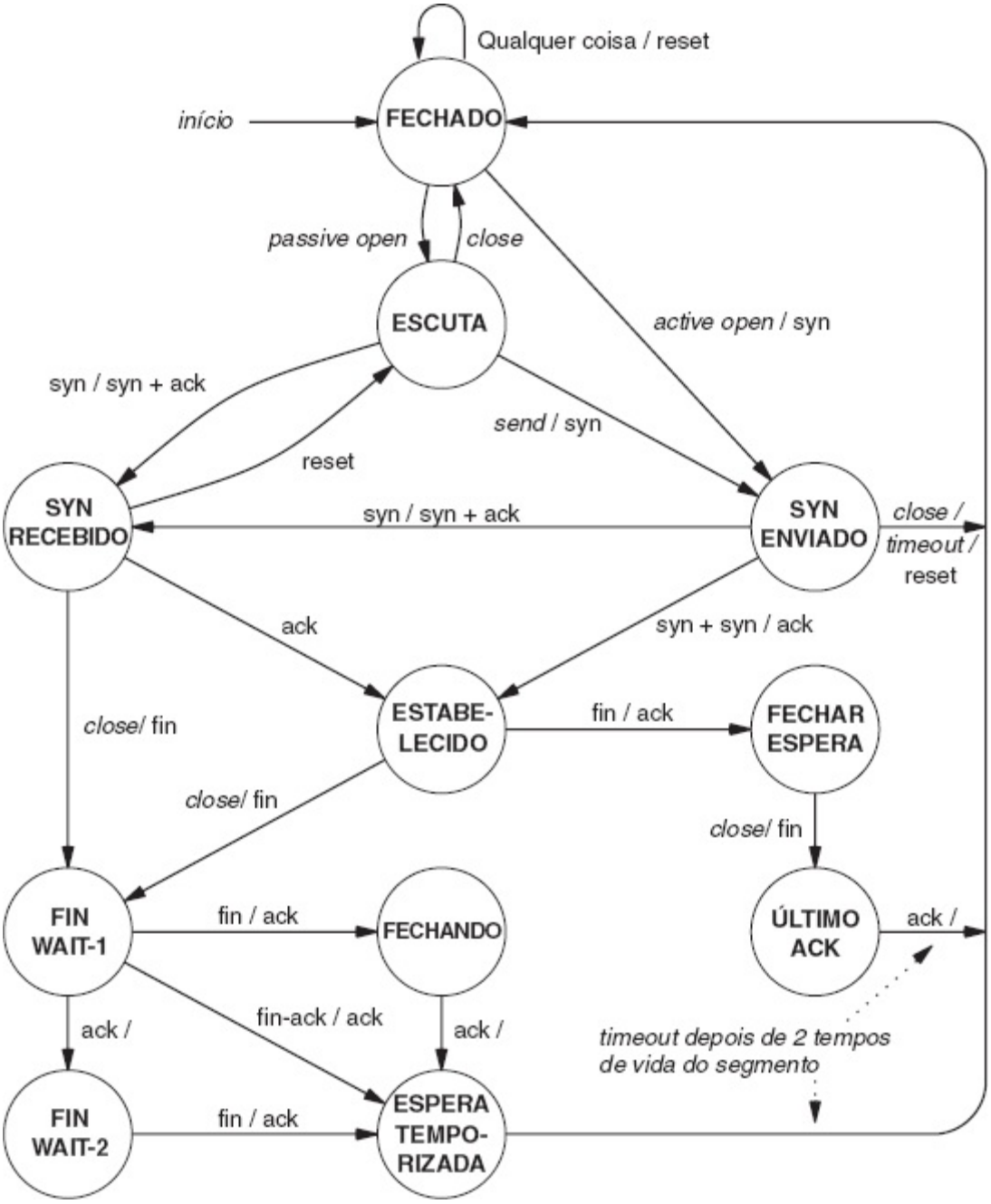


Figura 11.16 A máquina de estado finito para o TCP. Os rótulos nos estados de transição mostram a entrada que causou a transição seguida pela saída, se houver.

11.29 Forçando a entrega de dados

Dissemos que o TCP é livre para dividir o fluxo de dados em segmentos para transmissão sem considerar o tamanho da transferência que os programas aplicativos utilizam. A principal vantagem de permitir que o TCP escolha uma divisão é a eficiência. Ele pode acumular octetos

suficientes em um buffer para tornar os segmentos razoavelmente longos, reduzindo o alto overhead que ocorre quando os segmentos contêm apenas alguns octetos de dados.

Embora o uso de buffers melhore o rendimento da rede, isso pode interferir com algumas aplicações. Considere o uso de uma conexão TCP para passar caracteres de um terminal interativo para uma máquina remota. O usuário espera resposta instantânea para cada toque de tecla. Se o TCP de envio colocar os dados em buffers, a resposta poderá ser adiada, talvez por centenas de toques de tecla. De modo semelhante, como o TCP de recepção pode colocar os dados em buffer antes de torná-los disponíveis ao programa aplicativo em sua extremidade, forçar o emissor a transmitir dados pode não ser suficiente para garantir a entrega.

Para acomodar usuários interativos, o TCP oferece uma operação *push* que um programa aplicativo pode usar para forçar a remessa de octetos atualmente no fluxo sem esperar que o buffer seja preenchido. A operação *push* faz mais do que forçar o TCP local a enviar um segmento. Ela também solicita que o TCP defina o bit *PSH* no campo de código de segmento, de modo que os dados sejam entregues ao programa aplicativo na extremidade receptora. Assim, uma aplicação interativa usa a função *push* depois de cada toque de tecla. De modo semelhante, os programas aplicativos que controlam um display remoto podem usar a função *push* para garantir que os dados foram prontamente enviados através da conexão e passaram imediatamente para uma aplicação do outro lado.

11.30 Números de porta TCP reservados

Tal como o UDP, o TCP usa uma combinação de números de porta de protocolo estática e dinamicamente atribuída. Um conjunto de *portas bem conhecidas* foi atribuído por uma autoridade central para os serviços comumente acessados (por exemplo, servidores web e servidores de correio eletrônico). Outros números de porta estão disponíveis para que um sistema operacional aloque para aplicações locais, conforme necessário. Muitas portas TCP bem conhecidas existem agora. A Figura 11.17 lista algumas das portas TCP atualmente atribuídas.

Porta	Palavra-chave	Descrição
0		Reservado
7	echo	Eco
9	discard	Descartar
13	daytime	Hora do dia
19	chargen	Gerador de caracteres
20	ftp-data	File Transfer Protocol (dados)
21	ftp	File Transfer Protocol
22	ssh	Secure Shell
23	telnet	Conexão de Terminal
25	smtp	Simple Mail Transport Protocol
37	time	Hora
53	domain	Servidor de nome de domínio
80	www	World Wide Web
88	kerberos	Serviço de segurança Kerberos
110	pop3	Post Office Protocol versão 3
123	ntp	Network Time Protocol
161	snmp	Simple Network Management Protocol
179	bgp	Border Gateway Protocol
443	https	HTTP Seguro
860	iscsi	iSCSI (SCSI over IP)
993	imaps	IMAP Seguro
995	pop3s	POP3 Seguro
30301	bittorrent	Serviço BitTorrent

Figura 11.17 Exemplos de números atualmente atribuídos a portas TCP.

Devemos ressaltar que, embora os números de portas TCP e UDP sejam independentes, os desenvolvedores optaram por usar os mesmos números inteiros de porta para qualquer serviço que é acessível a partir de UDP e TCP. Por exemplo, um servidor de nome de domínio pode ser acessado

tanto com TCP como com UDP. Ao usar qualquer um dos protocolos, um aplicativo cliente pode usar o mesmo número de porta, *53*, porque IANA atribuiu *53* como de nome de domínio da porta de serviço tanto para o TCP como para o UDP.

11.31 Síndrome da janela tola e pacotes pequenos

Os pesquisadores que desenvolveram o TCP observaram um sério problema de desempenho que pode acontecer quando aplicações que estão enviando e recebendo operam em velocidades diferentes. Para entender o problema, lembre que o TCP coloca dados em buffer, e considere o que pode acontecer se uma aplicação receptora decidir ler os dados que chegam um octeto de cada vez. Quando uma conexão é inicialmente estabelecida, o TCP receptor aloca um buffer de *K* bytes e usa o campo JANELA em segmentos de confirmação para anunciar ao emissor o tamanho do buffer disponível. Se a aplicação emissora gerar dados rapidamente, o TCP emissor transmitirá segmentos com dados para a janela inteira. Por fim, o emissor receberá uma confirmação que especifica que a janela inteira foi preenchida e não resta qualquer espaço adicional no buffer do receptor.

Quando a aplicação receptora lê um octeto de dados de um buffer cheio, um octeto de espaço torna-se disponível. Dissemos que, quando o espaço se torna disponível em seu buffer, o TCP na máquina receptora gera uma confirmação que usa o campo JANELA para informar ao emissor. No exemplo, o receptor anunciará uma janela de *1* octeto. Ao descobrir que o espaço está disponível, o TCP de envio responde transmitindo um segmento que contém um octeto de dados.

Embora anúncios de janela de único octeto funcionem corretamente para manter o buffer do receptor cheio, eles resultam em uma série de segmentos de dados pequenos. O TCP de envio deve compor um segmento que contenha um octeto de dados, colocar o segmento em um datagrama IP e transmitir o resultado. Quando a aplicação receptora lê outro octeto, o TCP gera outra confirmação, que faz com que o emissor transmita outro segmento que contém um octeto de dados. A interação resultante pode alcançar um estado constante, em que o TCP envia um segmento separado para cada octeto de dados.

Transferir pequenos segmentos desnecessariamente consome largura de banda da rede e introduz overhead de computação. Pequenos segmentos consomem mais largura de banda por octeto de dado do que grandes segmentos, pois cada datagrama tem um cabeçalho. Se o datagrama carrega

somente um octeto de dado; a relação cabeçalho/data é grande. O overhead computacional surge porque o TCP nos computadores emissor e receptor precisa processar cada segmento. O software TCP de envio precisa alocar espaço no buffer, formar um cabeçalho de segmento e calcular um checksum para o segmento. De modo semelhante, o IP na máquina emissora precisa encapsular o segmento em um datagrama, calcular um checksum do cabeçalho, encaminhar o datagrama e transferi-lo para a interface de rede apropriada. Na máquina receptora, o IP precisa verificar o checksum do cabeçalho IP e passar o segmento ao TCP. O TCP precisa verificar o checksum do segmento, examinar o número de sequência, extrair os dados e colocá-los em um buffer.

Embora tenhamos descrito como ocorrem segmentos pequenos quando um receptor anuncia uma janela disponível pequena, um emissor também pode fazer com que cada segmento contenha uma pequena quantidade de dados. Por exemplo, imagine uma implementação do TCP que envie dados agressivamente sempre que estiverem disponíveis, e considere o que acontece se uma aplicação de envio gerar dados um octeto por vez. Depois que a aplicação gera um octeto de dados, o TCP cria e transmite um segmento. O TCP também pode enviar um pequeno segmento se uma aplicação gerar dados em blocos de tamanho fixo com B octetos e o TCP de envio extrair dados do buffer em blocos de tamanho máximo do segmento, M, onde MB, pois o último bloco em um buffer pode ser pequeno.

O problema de o TCP enviar segmentos pequenos se tornou conhecido como *síndrome da janela tola* (*Silly Window Syndrome – SWS*). Implementações TCP iniciais foram assoladas pela SWS. A seguir um resumo.

> *As primeiras implementações do TCP exibiam um problema conhecido como* síndrome da janela tola, *em que cada confirmação anuncia uma pequena quantidade de espaço disponível e cada segmento transporta uma pequena quantidade de dados.*

11.32 Evitando a síndrome da janela tola

As especificações do TCP agora incluem heurística que impede a síndrome da janela tola. Uma heurística usada na máquina emissora evita a transmissão de uma pequena quantidade de dados em cada segmento.

Outra heurística usada na máquina receptora evita o envio de pequenos incrementos em anúncios de janela que podem disparar pequenos pacotes de dados. Embora as heurísticas funcionem bem juntas, fazer com que o emissor e o receptor evitem a janela tola ajuda a garantir o bom desempenho no caso em que uma extremidade de uma conexão deixa de implementar corretamente o impedimento da janela tola.

Na prática, o TCP precisa conter código para impedimento de janela tola no emissor e no receptor. Para entender por que, lembre-se de que uma conexão TCP é full duplex — os dados podem fluir em qualquer direção. Assim, uma implementação do TCP inclui código para enviar dados e também código para recebê-los.

11.32.1 Evitando a janela tola no receptor

A heurística que um receptor usa para evitar a janela tola é simples e mais fácil de entender. Em geral, um receptor mantém um registro interno da janela atualmente disponível, mas atrasa o anúncio de um aumento no tamanho da janela ao emissor até que a janela possa avançar por um espaço significativo. A definição de "significativo" depende do tamanho de buffer do receptor e do tamanho máximo do segmento. O TCP o define como sendo o mínimo de metade do buffer do receptor ou o número de octetos de dados em um segmento de tamanho máximo.

A janela tola no lado do receptor impede anúncios de janela pequena no caso em que uma aplicação receptora extrai octetos de dados lentamente. Por exemplo, quando o buffer de um receptor se enche totalmente, ele envia uma confirmação que contém um anúncio de janela zero. À medida que a aplicação receptora extrai octetos do buffer, o TCP receptor calcula o espaço recém-disponível no buffer. Porém, em vez de enviar um anúncio de janela imediatamente, o receptor espera até que o espaço disponível alcance metade do tamanho total do buffer ou um segmento de tamanho máximo. Assim, o emissor sempre recebe grandes incrementos na janela atual, permitindo que transfira segmentos grandes. A heurística pode ser resumida da maneira a seguir.

> *Evitando a janela total no lado do receptor: antes de enviar um anúncio de janela atualizado depois de anunciar uma janela zero espere até que haja espaço disponível com pelo menos 50% do tamanho total do buffer ou que seja igual a um segmento de tamanho máximo.*

11.32.2 Confirmações adiadas

Duas técnicas têm sido usadas para implementar o impedimento da janela total no lado do receptor. Na primeira delas, o TCP confirma cada segmento que chega, mas não anuncia um aumento em sua janela até que esta atinja os limites especificados pela heurística de impedimento da janela tola. Na segunda técnica, o TCP adia o envio de uma confirmação quando o impedimento da janela tola especifica que a janela não é suficientemente grande para anunciar. Os padrões recomendam o adiamento das confirmações.

As confirmações adiadas possuem vantagens e desvantagens. A principal vantagem surge porque as confirmações adiadas podem diminuir o tráfego e, portanto, aumentar o rendimento. Por exemplo, se dados adicionais chegarem durante o período de espera, uma única confirmação validará todos os dados recebidos. Se a aplicação receptora gerar uma resposta imediatamente após os dados chegarem (por exemplo, o eco de caractere para uma sessão interativa), um pequeno atraso pode permitir que uma confirmação acompanhe um segmento de dados. Além do mais, o TCP não pode mover sua janela até que a aplicação receptora extraia dados do buffer. Nos casos em que a aplicação receptora lê dados assim que eles chegam, um atraso curto permite que o TCP envie um único segmento que confirma os dados e anuncia uma janela atualizada. Sem confirmações adiadas, o TCP confirmará a chegada de dados imediatamente, e mais tarde enviará uma confirmação adicional para atualizar o tamanho da janela.

As desvantagens das confirmações adiadas devem ser claras. Mais importante, caso um receptor adie as confirmações por muito tempo, o TCP de envio retransmitirá o segmento. Retransmissões desnecessárias reduzem o rendimento, pois desperdiçam largura de banda da rede. Além disso, as retransmissões exigem overhead computacional nas máquinas de envio e recepção. Acrescente-se ainda que o TCP usa a chegada de confirmações para estimar os tempos de ida e volta; o atraso das confirmações pode confundir a estimativa e tornar os tempos de retransmissão muito longos.

Para evitar problemas em potencial, os padrões do TCP colocam um limite sobre o tempo que o TCP adia uma confirmação. As implementações não podem adiar uma confirmação por mais do que 500 milissegundos. Além do mais, para garantir que o TCP receba um número suficiente de estimativas de ida e volta, o padrão recomenda que um receptor confirme pelo menos um dentre dois segmentos de dados.

11.32.3 Evitando a janela tola no emissor

A heurística que um TCP de envio usa para evitar a síndrome da janela tola é surpreendente e elegante. Lembre-se de que o objetivo é evitar o envio de segmentos pequenos. Lembre-se também de que uma aplicação de envio pode gerar dados em blocos arbitrariamente pequenos (por exemplo, um octeto de cada vez). Assim, para alcançar o objetivo, um TCP de envio precisa permitir que a aplicação de envio faça várias chamadas a *write* (ou *send*), e precisa coletar os dados transferidos em cada chamada antes de transmiti-la em um único e grande segmento. Ou seja, um TCP de envio precisa adiar o envio de um segmento até que possa acumular uma quantidade razoável de dados. A técnica é conhecida como *clumping*.

Surge uma pergunta: por quanto tempo o TCP deve esperar antes de transmitir dados? Por um lado, se o TCP esperar muito tempo, a aplicação experimenta grandes atrasos. Mais importante, o TCP não pode saber se deve esperar, pois não pode saber se a aplicação gerará mais dados no futuro próximo. Por outro lado, se o TCP não esperar por tempo suficiente, os segmentos serão pequenos, e o rendimento, baixo.

Os protocolos projetados antes do TCP confrontavam o mesmo problema e usavam técnicas para encaixar dados em pacotes maiores. Por exemplo, para conseguir a transferência eficiente por uma rede, os primeiros protocolos de terminal remoto adiavam a transmissão de cada toque de tecla por algumas centenas de milissegundos, para determinar se o usuário continuaria a pressionar teclas. Porém, como o TCP é projetado para ser genérico, ele pode ser usado por um conjunto diversificado de aplicações. Os componentes podem atravessar uma conexão TCP porque um usuário está digitando em um teclado ou porque um programa está transferindo um arquivo. Um atraso fixo não é ideal para todas as aplicações.

Assim como o algoritmo que o TCP usa para a retransmissão e o algoritmo de partida lenta usado para evitar o congestionamento, a técnica que um TCP de envio usa para evitar o envio de pacotes pequenos é adaptativa – o atraso depende do desempenho atual da internet subjacente. Assim como a partida lenta, o impedimento da janela tola no lado do emissor é chamado de *auto clocking*, pois não calcula os atrasos. Em vez disso, o TCP usa a chegada de uma confirmação para disparar a transmissão de pacotes adicionais. A heurística pode ser resumida da forma a seguir.

Evitando envio paralelo de janela tola: quando uma aplicação de envio gera dados adicionais a serem enviados por uma conexão para a qual

dados anteriores foram transmitidos, mas não confirmados, coloque os novos dados no buffer de saída como sempre, mas não envie segmentos adicionais até que haja dados suficientes para preencher um segmento de tamanho máximo. Se ainda estiver esperando para enviar quando uma confirmação chegar, envie todos os dados que foram acumulados no buffer. Aplique a regra mesmo quando o usuário requisitar uma operação push.

Se uma aplicação gerar dados um octeto de cada vez, o TCP enviará o primeiro octeto imediatamente. Porém, até que o ACK chegue, o TCP acumulará octetos adicionais em seu buffer. Assim, se a aplicação for razoavelmente rápida em comparação com a rede (ou seja, uma transferência de arquivo), segmentos sucessivos terão muitos octetos cada um. Se a aplicação for lenta em comparação com a rede (por exemplo, um usuário digitando em um teclado), pequenos segmentos serão enviados sem um atraso longo.

Conhecido como *algoritmo de Nagle*, devido ao nome do seu inventor, a técnica é especialmente elegante, pois exige pouco overhead computacional. Um host não precisa manter timers separados para cada conexão, e nem precisa examinar um clock quando uma aplicação gerar dados. Mais importante, embora a técnica se adapte a quaisquer combinações de atraso da rede, tamanho máximo do segmento e velocidade da aplicação, ela não reduz o rendimento nos casos convencionais.

Para entender por que o rendimento permanece alto para a comunicação convencional, observe que as aplicações otimizadas para um rendimento alto não geram dados um octeto de cada vez (isso geraria overhead desnecessário do sistema operacional). Em vez disso, essas aplicações escrevem grandes blocos de dados a cada chamada. Assim, o buffer TCP de saída começa com dados suficientes para pelo menos um tamanho máximo de segmento. Além do mais, como a aplicação produz dados mais rápido do que o TCP pode transferi-los, o buffer de envio permanece quase cheio, e o TCP não adia a transmissão. Como resultado, o TCP continua a enviar segmentos em qualquer velocidade que a internet básica possa tolerar, enquanto a aplicação continua a preencher o buffer. Podemos fazer o resumo a seguir.

O TCP agora exige que o emissor e o receptor implementem a heurística que evita a síndrome da janela tola. Um receptor evita anunciar uma janela pequena e um emissor usa um esquema adaptativo para adiar a transmissão de modo que possa reunir dados em segmentos grandes.

11.33 Buffer Bloat e seus efeitos em latência

TCP é projetado para maximizar o rendimento, adaptando-se às condições da rede. Como resultado, mantém buffers em dispositivos de rede quase cheios. Ao longo de várias décadas, o preço da memória diminuiu e os fornecedores têm aumentado a quantidade de memória em dispositivos de rede. Mesmo um pequeno roteador Wi-Fi caseiro tem mais memória do que os maiores roteadores da década de 1980.

Pode parecer que a adição de memória para um dispositivo de rede irá sempre melhorar o desempenho, pois o dispositivo pode acomodar pulsos de pacotes (packet bursts) com menos pacotes descartados. No entanto, um sério problema pode surgir quando os dispositivos de rede têm grandes memórias e enviam pacotes através de links lentos: longa latência. O aumento da latência significa que a comunicação em tempo real, tal como as chamadas de telefone VoIP, torna-se inutilizável.

Para apreciar o problema, considere um roteador Wi-Fi em uma casa. Suponha que dois usuários estão usando a Internet: um está fazendo download de um filme e outro usando o Skype para uma chamada telefônica. Imagine que o roteador usa apenas 4 MB de memória como um buffer de pacote (uma estimativa conservadora). Como ele usa TCP e sempre tem dados para enviar, o download do filme vai manter o buffer quase cheio. A conversa pelo Skype envia dados a uma taxa muito mais baixa do que o download. Quando um pacote Skype chega, é colocado no fim do buffer, e não é entregue até que todos os pacotes que estavam esperando no buffer tenham sido enviados. Quanto tempo leva para esvaziar um buffer? Uma conexão Wi-Fi usando 802.11g tem uma taxa de entrega efetiva de aproximadamente 20 Mbps. Um mega-byte de memória contém 8.388.608 bits, então um buffer de 4 megabytes detém 33.554.432 bits. Sabemos que o receptor irá transmitir ACKs, o que significa que o roteador não pode enviar dados do buffer de forma contínua. Para fins de análise, assuma o melhor caso em que a rede não tem nenhum outro tráfego e não há atraso entre pacotes. Mesmo sob as condições idealizadas, o tempo

necessário para transmitir um buffer de dados é:

$$Atraso\ do\ buffer = \frac{3{,}36 \times 10^7\ bits}{2{,}0 \times 10^7\ bits/segundo} = 1{,}68\ segundos$$

Em outras palavras, o usuário que está tentando realizar uma chamada Skype vai experimentar um atraso intolerável. O atraso é perceptível, mesmo que um usuário esteja só navegando na web.

Usamos o termo *buffer bloat* para descrever o uso de buffers muito grandes em dispositivos de rede. O aspecto mais surpreendente do buffer bloat é que o aumento da largura de banda de rede antes do link gargalo não vai melhorar o desempenho e pode piorar a latência. Ou seja, pagar por uma conexão de Internet de maior velocidade não vai resolver o problema. Para mais informações sobre o assunto, consulte o seguinte vídeo (conteúdo em inglês): *http://www.youtube.com/watch?v=-D-cJNtKwuw*.

11.34 Resumo

O Transmission Control Protocol, TCP, define um serviço-chave para comunicação na internet: a remessa por fluxo confiável. O TCP fornece uma conexão full duplex entre duas máquinas, permitindo que elas troquem grandes volumes de dados de forma eficiente.

Por usar um protocolo de janela deslizante, o TCP pode fazer uso eficiente de uma rede. Por fazer poucas suposições sobre o sistema de remessa subjacente, o TCP é flexível o bastante para operar por uma grande variedade de sistemas de remessa. Por fornecer controle de fluxo, o TCP permite que sistemas de velocidades bastante variáveis se comuniquem.

A unidade básica de transferência usada pelo TCP é um segmento. Os segmentos são usados para passar dados ou controlar informações (por exemplo, para permitir que o TCP em duas máquinas estabeleça conexões ou as termine). O formato do segmento permite que uma máquina reúna confirmações para dados fluindo em uma direção, incluindo-as nos cabeçalhos de segmento dos dados fluindo na direção oposta.

O TCP implementa o controle de fluxo fazendo com que o receptor anuncie a quantidade de dados que deseja aceitar. Ele também admite mensagens fora de faixa usando uma facilidade de dados urgentes e força a remessa usando um mecanismo push.

O padrão TCP atual especifica o recuo exponencial para timers de retransmissão e algoritmos de impedimento de congestionamento, como partida lenta, aumento aditivo, e diminuição multiplicativa. Além disso, o

TCP usa a heurística para evitar transferir pacotes pequenos. Finalmente, o IETF recomenda que os roteadores utilizem RED no lugar do descarte de cauda, pois isso evita o sincronismo do TCP e melhora o rendimento.

EXERCÍCIOS

11.1 O TCP utiliza um campo finito para conter números de sequência de fluxo. Estude a especificação do protocolo para descobrir como ele permite que um fluxo de tamanho qualquer passe de uma máquina para outra.

11.2 O texto observa que uma das opções do TCP permite ao receptor especificar o tamanho de segmento máximo que deseja aceitar. Por que o TCP admite uma opção para especificar o tamanho de segmento máximo quando também tem um mecanismo de anúncio de janela?

11.3 Sob que condições de atraso, largura de banda, carga e perda de pacotes o TCP retransmitirá volumes significativos de dados desnecessariamente?

11.4 Uma confirmação TCP perdida não necessariamente força uma retransmissão. Explique por quê.

11.5 Faça experimentos com máquinas locais para determinar como o TCP lida com reinicialização do computador. Estabeleça uma conexão da máquina *X* para a máquina *Y* e deixe a conexão ociosa. Reinicie a máquina *Y* e então force o aplicativo na máquina *X* a enviar um segmento. O que acontece?

11.6 Imagine uma implementação do TCP que descarte segmentos que chegam fora de ordem, mesmo que caiam na janela atual. Ou seja, a versão imaginada só aceita segmentos que estendem o fluxo de bytes que já recebeu. Isso funciona? Como se compara a uma implementação TCP padrão?

11.7 Considere o cálculo de um checksum TCP. Considere que, embora o campo de checksum no segmento *não* tenha sido definido como zero, o resultado do cálculo do checksum *é* zero. O que você pode concluir?

11.8 Quais são os argumentos a favor e contra o fechamento automaticamente de conexões ociosas?

11.9 Se dois programas aplicativos utilizam TCP para enviar dados, mas

somente enviam um caractere por segmento (por exemplo, usando a operação *push*), qual é a porcentagem máxima da largura de banda de rede que eles terão para seus dados com IPv4? Com IPv6?

11.10 Suponha que uma implementação do TCP utilize o número de sequência inicial *1* quando cria uma conexão. Explique como uma falha e um reinício do sistema podem levar um sistema remoto a crer que a conexão antiga permaneceu aberta.

11.11 Descubra como as implementações do TCP precisam solucionar o *problema do segmento sobreposto*. O problema aparece porque o receptor precisa aceitar apenas uma cópia de todos os bytes do fluxo de dados, mesmo que o emissor transmita dois segmentos que sobrepõem parcialmente um ao outro (por exemplo, o primeiro segmento transporta os bytes de 100 a 200, e o segundo transporta os bytes de 150 a 250).

11.12 Rastreie as transições da máquina de estado finito do TCP para duas extremidades de uma conexão. Admita que um lado execute uma abertura passiva e o outro lado execute uma abertura ativa, e percorram o handshake de três vias.

11.13 Leia a especificação TCP para descobrir as condições exatas sob as quais o TCP pode fazer a transição de FIN WAIT-1 para TIMED WAIT.

11.14 Rastreie as transições de estado do TCP para duas máquinas que concordam em fechar uma conexão de forma controlada.

11.15 Considere que o TCP esteja enviando segmentos por meio de um tamanho máximo de janela de 64 Kbytes em um canal que possui largura de banda finita e um tempo de ida e volta médio de 20 milissegundos. Qual é o rendimento máximo? Como o rendimento muda se o tempo de ida e volta aumentar para 40 milissegundos (enquanto a largura de banda permanece infinita)? Você precisa assumir IPv4 ou IPv6 para responder a essa pergunta? Sim ou não? Por quê?

11.16 Você poderia derivar uma equação que expresse o throughput máximo possível do TCP como uma função da largura de banda da rede, o atraso da rede e o tempo para processar um segmento e gerar uma confirmação? (Dica: considere o exercício anterior.)

11.17 Descreva circunstâncias (anormais) que podem deixar uma extremidade de uma conexão no estado *FIN WAIT-2*

indefinidamente. (Dica: pense na perda de datagrama e nas falhas do sistema.)

11.18 Mostre que, quando um roteador implementa RED, a probabilidade de um pacote ser descartado de uma conexão TCP em particular é proporcional à porcentagem de tráfego que a conexão gera.

11.19 Argumente que uma retransmissão rápida poderia ser ainda mais rápida se ela usasse um ACK duplicado como um gatilho. Por que o padrão exige vários ACKs duplicados?

11.20 Para ver se é necessário um esquema SACK na Internet moderna, meça a perda de datagramas em uma conexão TCP de longa duração (por exemplo, um aplicativo de fluxo de vídeo). Quantos segmentos foram perdidos? O que você pode concluir?

11.21 Considere um roteador sem fio com uma conexão de 3 Mbps à Internet e um (bloated) buffer de 256 MB. Se dois usuários estão baixando filmes e um terceiro usuário tenta acessar o *google.com*, qual é o tempo mínimo antes de o terceiro usuário receber uma resposta?

11.22 No exercício anterior, sua resposta muda se a conexão entre o roteador e a Internet for de 10 Mbps? Sim ou não? Por quê?

---

* A Figura 11.4 pode ser encontrada na página 139.

** Há duas exceções à transmissão quando o tamanho da janela é zero: um emissor envia um segmento com o bit urgente marcado quando dados urgentes estão disponíveis e um remetente verifica periodicamente uma janela de tamanho zero no caso de um anúncio diferente de zero ser perdido.

*** Na prática, não costuma ocorrer piggybacking porque a maioria dos aplicativos não envia dados em ambas as direções simultaneamente.

* A especificação TCP diz que o campo HLEN é o *offset* da área dentro do segmento.

* Uma seção posterior explica como o cálculo foi modificado em versões subsequentes do TCP. A fórmula simplista usada em implementações iniciais do TCP faz com que seja fácil de entender o conceito básico de uma estimativa de ida e volta que muda ao longo do tempo.

* A estimativa só pode ter um crescimento arbitrariamente grande se cada segmento for perdido pelo menos uma vez.

* BSD UNIX usa uma tabela de fatores, mas os valores na tabela são equivalentes a usar $\delta = 2$.

** Phil Karn desenvolveu o algoritmo para a comunicação TCP através de uma

conexão radioamadora de alta perda.

* O termo *partida lenta* é atribuído a John Nagle; a técnica era chamada originalmente de *partida macia* (*soft-start*).

* O AIMD foi definido na seção anterior.

* O esquema de Notificação Explícita de Congestionamento mencionado anteriormente ainda não foi adotado.

** Variância em atraso é referida como *jitter*.

*** Curiosamente, a sincronização global não ocorre se o número de conexões TCP que compartilham um link for suficientemente grande (> 500) e os RTTs variarem.

* Um valor de exemplo sugerido para y é 0,002.

* SYN significa sincronismo; a pronúncia é “sin”; o segmento que carrega o SYN é chamado de “segmento sin”.

* Um segmento com o bit FIN marcado é chamado um “segmento fin” (“fin segment”).

CAPÍTULO 12

Arquitetura de roteamento: núcleo, pares e algoritmos

CONTEÚDOS DO CAPÍTULO

12.1 Introdução

Os capítulos anteriores concentram-se nos serviços de comunicação que o TCP/IP oferece para aplicativos e os detalhes dos protocolos nos hosts e roteadores que fornecem esses serviços. Na discussão, consideramos que os roteadores sempre contêm rotas corretas e vimos que um roteador pode usar o mecanismo ICMP Redirect para instruir um host diretamente conectado a mudar uma rota.

Este capítulo considera duas questões básicas: "que valores cada tabela de encaminhamento deve conter, e como esses valores podem ser obtidos?". Para responder à primeira pergunta, consideraremos a relação entre a arquitetura da internet e o roteamento. Em especial, discutiremos internets estruturadas em torno de um backbone e aquelas compostas por múltiplas redes paralelas (peer), e consideraremos as consequências para o roteamento. O primeiro é típico de uma intranet corporativa; este último se aplica à Internet global. Para responder à segunda pergunta, vamos considerar os dois tipos básicos de algoritmos de propagação de rota e ver como cada um fornece informações de roteamento automaticamente.

Começaremos discutindo o encaminhamento em geral. As seções subsequentes concentram-se na arquitetura da internet e descrevem os algoritmos que os roteadores utilizam para trocar informações de roteamento. Os Capítulos 13 e 14 continuam a expandir nossa discussão do roteamento. Eles exploram os protocolos que os roteadores pertencentes a

dois grupos administrativos independentes usam para trocar informações e os protocolos que um único grupo usa entre todos os seus roteadores.

12.2 A origem das tabelas de roteamento

Você deve lembrar, do Capítulo 3, que os roteadores IP fornecem interconexões ativas entre redes. Cada roteador se conecta a duas ou mais redes físicas e encaminha datagramas IP entre elas, aceitando aqueles que chegam por uma interface de rede e enviando-os por outra interface. Com exceção dos destinos em redes diretamente conectadas, os hosts passam todo o tráfego IP para os roteadores, que encaminham datagramas aos seus destinos finais. Geralmente, um datagrama viaja de roteador para roteador até que alcance um que se conecte diretamente à mesma rede do destino final. Portanto, o sistema de roteadores forma a base arquitetônica de uma internet e lida com todo o tráfego, exceto com o de distribuição direta de um host para outro.

O Capítulo 8 descreve o algoritmo que os hosts e roteadores seguem quando encaminham datagramas e mostra como o algoritmo usa uma tabela para tomar decisões. Cada entrada na tabela de encaminhamento usa um endereço e uma máscara para especificar o prefixo de um determinado destino e fornece o endereço do próximo roteador em um caminho usado para alcançar essa rede. Na prática, cada entrada também especifica uma interface de rede local que deve ser usada para alcançar o próximo salto.

Não dissemos como os hosts ou roteadores obtêm as informações para suas tabelas de encaminhamento. A questão tem dois aspectos: *que* valores devem ser colocados nas tabelas e *como* os roteadores obtêm os valores. Ambas as escolhas dependem da complexidade arquitetônica e do tamanho da internet, bem como das políticas administrativas.

Em geral, estabelecer rotas envolve duas etapas: inicialização e atualização. Um host ou roteador deve estabelecer uma tabela inicial de rotas quando de sua inicialização, e deve atualizar a tabela conforme as rotas mudam (por exemplo, quando o hardware falha, tornando uma determinada rede não utilizável). Este capítulo vai se concentrar nos roteadores; o Capítulo 22 descreve como os hosts usam DHCP para obter entradas para uma tabela de encaminhamento.

A inicialização depende do hardware e do sistema operacional. Em alguns sistemas, o roteador lê uma tabela de encaminhamento inicial de um armazenamento secundário na inicialização, seja de um disco ou de uma memória flash. Em outros, o roteador começa com uma tabela vazia que é preenchida executando-se um script durante o boot do roteador. Entre

outras coisas, a instrução de inicialização contém comandos que inicializam o hardware da rede e configuram um endereço IP para cada interface de rede. Finalmente, alguns sistemas iniciam por uma mensagem broadcasting (ou multicasting) que busca vizinhos e requisita que eles forneçam informações sobre endereços que estão sendo usados na rede.

Uma vez que uma tabela de encaminhamento inicial tenha sido construída, um roteador precisa ajustar mudanças nas rotas. Em internets pequenas e que mudam lentamente, os administradores podem estabelecer e modificar rotas manualmente. Em ambientes grandes e que mudam de forma rápida, porém, a atualização manual é incrivelmente lenta e propensa a erros humanos. Métodos automatizados são necessários. Antes que possamos entender os protocolos automáticos usados para troca de informação nos roteadores, precisamos revisar várias ideias básicas. As próximas seções fazem isso, fornecendo o fundamento conceitual necessário para o roteamento principal é descrita a seguir.

12.3 Encaminhamento com informações parciais

A principal diferença entre roteadores e hosts típicos é que os hosts normalmente sabem pouco sobre a estrutura da rede com a qual se conectam. Os hosts não possuem conhecimento completo de todos os endereços de destino possíveis, ou mesmo de todas as redes de destino possíveis. Na verdade, muitos hosts possuem apenas duas entradas em sua tabela de encaminhamento: uma entrada para a rede local e uma entrada-padrão para um roteador diretamente conectado. O host envia todos os datagramas não locais para o roteador local para entrega. A questão principal é descrita a seguir.

> *Um host pode encaminhar datagramas com sucesso mesmo se tiver apenas informações de encaminhamento parciais, pois ele pode se basear em um roteador.*

Os roteadores também podem encaminhar datagramas apenas com informações parciais? Sim, mas somente sob certas circunstâncias. Para entender os critérios, imagine que uma internet seja um país externo cortado por estradas de terra com placas de trânsito colocadas nas interseções. Suponha que você não tenha mapa algum, não possa pedir informações porque não fala o idioma local, não tenha nenhuma ideia sobre

os marcos visíveis, mas precise viajar para um vilarejo chamado *Sussex*. Você parte em sua viagem, seguindo a única estrada que sai da cidade e começa a procurar placas de trânsito. A primeira placa diz:

Norfolk à esquerda; Hammond à direita; outras, em frente.*

Como o destino que deseja não aparece explicitamente, você continua em frente. No jargão do roteamento, dizemos que você segue uma *rota-padrão* (*default route*). Após várias outras placas, finalmente encontra uma que diz:

Essex à esquerda; Sussex à direita; outras, em frente.

Você vira à direita, segue algumas outras placas e sai em uma estrada que leva a Sussex.

Nossa viagem imaginária é análoga a um datagrama atravessando uma internet, e as placas trânsito são análogas às tabelas de encaminhamento nos roteadores ao longo do caminho. Sem um mapa ou outros recursos de orientação, a viagem depende completamente das placas rodoviárias, assim como o encaminhamento de datagrama em uma internet depende inteiramente das tabelas de encaminhamento. Sem dúvida, é possível trafegar ainda que cada placa rodoviária contenha apenas informações parciais.

Uma questão central diz respeito à precisão. Como viajante, você pode perguntar: "como posso estar certo de que seguir as placas me levará ao meu destino seguindo o menor caminho?". Essas perguntas podem parecer especialmente problemáticas se você passar por muitas placas sem encontrar seu destino especificado de maneira direta. É claro, as respostas dependem da topologia do sistema rodoviário e do conteúdo das placas, mas a ideia básica é a de que, quando tomadas como um todo, as informações nas placas devem ser consistentes e completas. Olhando de outra forma, vemos que não é necessário que cada interseção tenha uma placa para cada destino. As placas podem listar caminhos-padrão, já que todas as placas explícitas apontam um caminho mais curto, e as voltas para caminhos mais curtos para todos os destinos estão marcadas. Alguns exemplos explicarão algumas maneiras de como a consistência pode ser obtida.

Em um extremo considere uma topologia simples em forma de estrela de estradas nas quais cada cidade tem exatamente uma estrada que leva a ela, e todas as estradas se encontram em um ponto central. A Figura 12.1 ilustra

essa topologia.

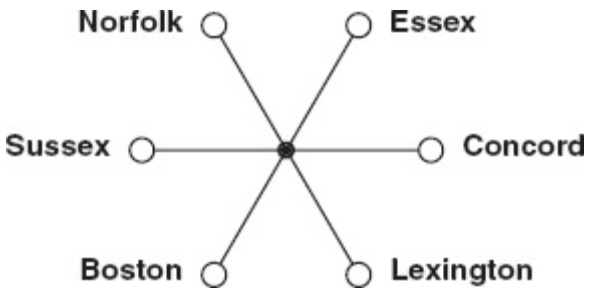


Figura 12.1 Um exemplo de topologia de estradas em formato de estrela conectando cidades.

Imaginamos uma placa na intersecção central que lista cada cidade possível e a estrada para alcançá-la. Em outras palavras, somente a intersecção central tem informação sobre cada destino possível; um viajante sempre vai até a intersecção central a caminho de qualquer destino.

No outro extremo, podemos imaginar um conjunto arbitrário de estradas com sinais em cada intersecção listando todos os destinos possíveis. Para garantir que as placas levem os viajantes pela melhor estrada, precisa ser verdade que, em qualquer interseção, se a placa para o destino D aponta para a estrada R, nenhuma outra estrada diferente de R leva a um caminho mais curto até D.

Nenhum desses extremos arquitetônicos funciona bem para um sistema de roteadores da internet. Por um lado, o método de interseção central falha porque nenhum equipamento é rápido o suficiente para servir como uma chave central através da qual todo o tráfego passa. Por outro lado, ter informações sobre todos os destinos possíveis em todos os roteadores é impraticável porque isso exige propagar grandes volumes de informação sempre que ocorrer uma mudança na internet. Portanto, buscamos uma solução que permita que grupos gerenciem roteadores locais de forma autônoma, acrescentando novas interconexões de rede e rotas sem mudar as informações de encaminhamento em roteadores distantes.

Para entender a arquitetura de roteamento usada na Internet, considere uma terceira topologia em que metade das cidades se encontra na parte leste do país, e metade, na parte oeste. Suponha que uma única ponte atravessa o rio que separa o leste do oeste. Considere que as pessoas que moram na parte leste não gostam das que moram na parte oeste e, portanto, desejam permitir placas rodoviárias que mostrem destinos no leste, mas não no oeste. Imagine que as pessoas que moram no oeste façam o contrário. O

roteamento será consistente se toda placa rodoviária no leste mostrar todos os destinos do leste explicitamente e apontar o caminho-padrão para a ponte, e toda placa no oeste mostrar todos os destinos do oeste explicitamente e apontar o caminho-padrão para a ponte.

Contudo, existe um problema: se um turista que tenha acidentalmente escrito o nome de uma cidade não existente chegar, ele pode cruzar a ponte uma vez e então descobrir que o caminho-padrão o leva de volta para a ponte.

12.4 Arquitetura original da Internet e núcleos (cores)

Uma grande parte do nosso conhecimento dos protocolos de encaminhamento e propagação foi derivada da experiência com a Internet. Quando o TCP/IP foi inicialmente desenvolvido, os sites de pesquisa participantes eram conectados à ARPANET, que servia como o backbone conectando todos os sites na Internet. Durante os testes iniciais, cada site gerenciava tabelas de encaminhamento e, manualmente, instalava rotas para outros destinos. À medida que a Internet bebê começou a crescer, tornou-se claro que a manutenção manual de rotas seria inviável; eram necessários mecanismos automatizados. O conceito de rede backbone continua sendo usado: muitas empresas grandes têm um backbone que conecta sites em sua intranet.

Os desenvolvedores da Internet selecionaram uma arquitetura de roteadores que seguia a topologia em formato de estrela descrita anteriormente. O projeto original usou um pequeno conjunto central de roteadores que mantinham informações completas sobre todos os possíveis destinos, e um grande conjunto de roteadores distantes que mantinham informação parcial. Seguindo nossa analogia, é como designar um pequeno conjunto de intersecções localizadas de forma central que tenham placas que mostrem todos os destinos e permitir que as intersecções distantes mostrem apenas destinos locais. Como a rota-padrão em cada interseção distante aponta para uma das intersecções centrais, os viajantes cedo ou tarde alcançarão seu destino.

O conjunto central de roteadores que mantinha informações completas era chamado o *núcleo* (*core*) da Internet. Como cada roteador core armazena uma rota para cada destino possível, não precisa de uma rota-padrão. Por isso, o conjunto de roteadores-núcleo é referenciado algumas vezes como uma *zona default-free*. A vantagem de particionar o roteamento da Internet em um sistema de duas camadas é que isso permite aos administradores locais gerenciarem as mudanças locais em roteadores

remotos sem afetar outras partes da Internet. A desvantagem é que isso introduz um potencial para inconsistência. Na pior das hipóteses, um erro em um roteador remoto pode tornar rotas distantes inalcançáveis.

Podemos resumir as ideias a seguir.

A vantagem da arquitetura de roteamento core é que, como os roteadores não pertencentes ao core usam informações parciais, um site distante tem autonomia para fazer mudanças de roteamento locais. A desvantagem é que um site pode apresentar inconsistências que tornam alguns destinos inalcançáveis.

As inconsistências entre tabelas de encaminhamento podem surgir de erros nos algoritmos que calculam tabelas de encaminhamento, de dados incorretos fornecidos a esses algoritmos ou de erros que ocorrem durante a transmissão dos resultados a outros roteadores. Os desenvolvedores de protocolo procuram meios de limitar o impacto dos erros, com o objetivo de manter todas as rotas consistentes o tempo todo. Caso as rotas se tornem inconsistentes, os protocolos de roteamento devem ser robustos o bastante para detectar e corrigir os erros de forma rápida. Mais importante, os protocolos devem ser projetados para restringir o efeito dos erros.

A arquitetura da Internet inicial é fácil de entender se nos lembrarmos de que a Internet evoluiu com um backbone remoto, a ARPANET, já estabelecido. Uma grande motivação para o sistema de roteador core veio do desejo de conectar redes locais ao backbone. A Figura 12.2 ilustra a ideia.

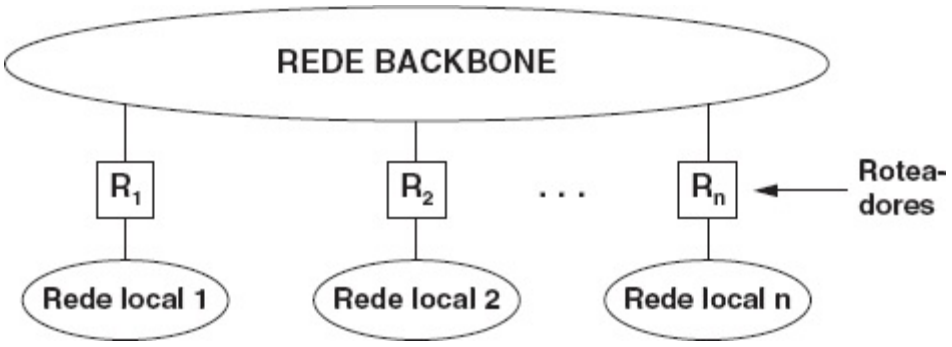


Figura 12.2 O sistema de roteador core da Internet visto como um conjunto de roteadores que conectam redes locais com o backbone. Essa arquitetura é usada hoje em redes de empresas.

Para entender por que os roteadores na Figura 12.2 não podem usar informações parciais, considere o caminho que um datagrama segue se um conjunto de roteadores usar uma rota-padrão. No site de origem, o roteador local verifica se possui uma rota explícita até o destino e, caso não tenha, envia o datagrama ao longo do caminho especificado por sua rota-padrão. Todos os datagramas para os quais o roteador não tem nenhuma rota específica seguem o mesmo caminho-padrão, seja qual for seu destino final. O próximo roteador ao longo do caminho desvia os datagramas para os quais tem uma rota explícita e envia o restante ao longo de sua rota-padrão. Para assegurar consistência global, a cadeia de rotas-padrão precisa alcançar cada roteador em um ciclo gigantesco. Assim, a arquitetura exige que todos os sites locais coordenem suas rotas-padrão.

Há dois problemas com uma arquitetura de roteamento que envolve um conjunto de rotas-padrão. Primeiro, suponha que um computador acidentalmente gere um datagrama para um destino que não existe (ou seja, para um endereço IP que não tenha sido atribuído). O host envia o datagrama para o roteador local, que segue o caminho-padrão até o próximo roteador, e assim por diante. Infelizmente, como as rotas-padrão formam um ciclo, o datagrama circula até o limite salto expirar. Em segundo lugar, se ignorarmos o problema de endereços inexistentes, o encaminhamento é ineficiente. Um datagrama que segue as rotas-padrão pode atravessar $n - 1$ roteadores antes que atinja um roteador, que se liga à rede local do destino.

Para evitar as ineficiências e potenciais loops de roteamento que as rotas-padrão causam, a Internet inicial proibia rotas-padrão em roteadores core. Em vez de usá-las, os desenvolvedores organizaram roteadores para trocar informações de roteamento de modo que cada um sabia como encaminhar datagramas diretamente.

Fazer isso era fácil, uma vez que todos os roteadores se conectavam a uma única rede de backbone, o que significa que eles podiam se comunicar diretamente

12.5 Além da arquitetura básica para backbones peer

A introdução do backbone NSFNET na Internet acrescentou nova complexidade à estrutura de roteamento e obrigou os desenvolvedores a inventarem uma nova arquitetura de roteamento. Mais importante, a mudança na arquitetura anunciou a Internet atual, na qual um conjunto de *Tier-1 ISP* tem, cada um, um backbone de longa distância ao qual sites de usuários se conectam. De várias maneiras, o serviço na NSFNET e a arquitetura de roteamento que foi criada para apoiá-lo foram a chave para o

avanço da arquitetura original da Internet para a arquitetura atual.

A principal mudança que ocorreu com o backbone NSFNET foi a evolução de um backbone central único para um conjunto de redes de *pares de backbones* (*peer backbone networks*), frequentemente chamado *peers* ou *routing peers*. Embora a Internet global tenha agora vários Tier-1 peers, podemos entender a situação de roteamento considerando somente dois. A Figura 12.3 ilustra uma topologia de Internet com um par de redes de backbone.

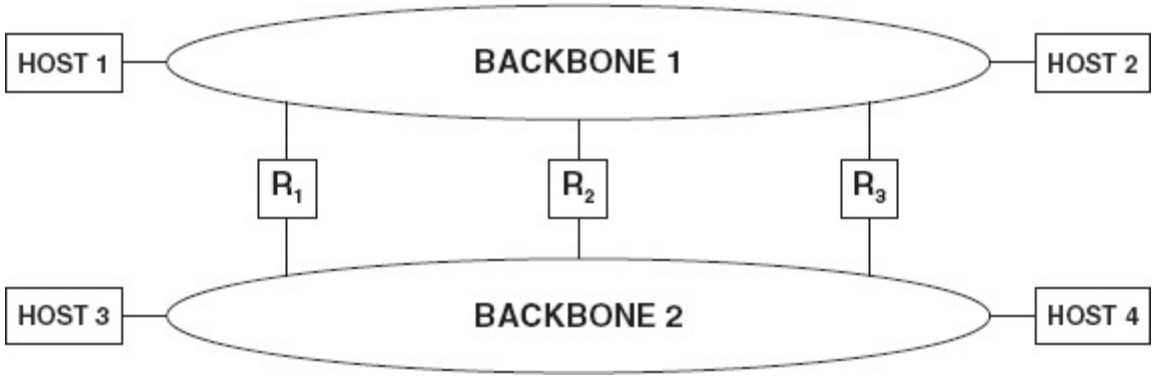


Figura 12.3 Um exemplo de dois pares backbones (peer backbones) interconectados por vários roteadores similares aos dois peer backbones na Internet em 1989.

Para nos ajudar a entender as dificuldades do roteamento IP entre peer backbones, a figura mostra quatro hosts conectados diretamente a backbones. Apesar de essa conexão direta poder parecer não realista, ela simplifica o exemplo. Olhe para a figura e considere rotas do host *3* para o host *2*. Vamos assumir por enquanto que a figura mostra orientação geográfica: o host *3* está na Costa Oeste conectado ao backbone 2, enquanto o host *2* está na Costa Leste conectado com o backbone 1. Ao estabelecer rotas entre os hosts *3* e *2*, os gerentes devem decidir entre três opções a seguir.

(a) Rotear o tráfego do host *3* através do roteador da Costa Oeste, R_1, e então através do backbone 1.

(b) Encaminhar o tráfego do host *3* através do backbone 2, através do roteador de Meio Oeste, R_2, e então através do backbone 1 para o

host *2*.

(c) Rotear o tráfego através do backbone 2, através da Costa Leste do roteador, R_3, e então do host *2*.

Uma rota mais tortuosa também é possível: o tráfego poderia fluir do host *3* através do roteador da Costa Oeste, através de backbone 1, até o roteador do Centro-Oeste, de volta para backbone 2 para o roteador da Costa Leste e, finalmente, através de backbone 1 até o host *2*. Essa rota pode ser aconselhável ou não, dependendo das políticas de uso da rede e da capacidade dos vários roteadores e backbones. No entanto, vamos nos concentrar no roteamento em que um datagrama nunca atravessa uma rede duas vezes (ou seja, nunca se move para uma rede, se move de uma rede, e, então, volta para a rede novamente).

Intuitivamente, gostaríamos de que todo o tráfego tomasse um caminho mais curto. Ou seja, que o tráfego entre um par de hosts geograficamente próximos tomasse um caminho curto, independentemente das rotas escolhidas para o tráfego de longa distância. Por exemplo, é desejável para o tráfego do host *3* para o host *1* fluir através do roteador Costa Oeste, R_1, porque esse caminho minimiza a distância total que o datagrama viaja. Mais importante, se um datagrama deve viajar através de um backbone, um ISP gostaria de manter o datagrama em seu backbone (porque isso é economicamente menos dispendioso do que usar um peer).

As metas anteriores parecem simples e sensatas. No entanto, elas não podem ser traduzidas em um esquema de encaminhamento razoável, por dois motivos. Primeiro, embora o algoritmo de encaminhamento IP padrão use a parte da rede de um endereço IP para escolher uma rota, o encaminhamento ideal em uma arquitetura de backbone peer requer rotas individuais para hosts individuais. Por exemplo, considere a tabela de envio de host *3*. O próximo salto ideal para o host *1* é o roteador da Costa Oeste, R_1, e o próximo salto ideal para o host *2* é o roteador da Costa Leste, R_3. Porém, os hosts *1* e *2* conectam-se ambos ao backbone 1, o que significa que eles têm o mesmo prefixo de rede. Portanto, em vez de usar prefixos de rede, a tabela de encaminhamento do host deve conter rotas específicas de hosts. Segundo, os gerentes dos dois backbones devem concordar em manter rotas consistentes entre todos os roteadores ou pode se desenvolver um *encaminhamento em loop* (*loop de roteamento*), no qual um

conjunto de roteadores encaminham um para o outro em um ciclo.

12.6 Propagação automática de rota e uma FIB

Dissemos que o sistema core da Internet evitava as rotas-padrão porque elas propagavam informações completas sobre todos os destinos possíveis a cada roteador core. Muitas internets corporativas agora usam um esquema semelhante – ele propaga informação sobre cada destino na corporação para todos os roteadores nas suas intranets. A próxima seção discute dois tipos básicos de algoritmos que calculam e propagam informações de roteamento; os capítulos seguintes discutem protocolos que usam os algoritmos.

Os protocolos de roteamento desempenham duas funções importantes. Primeiro, eles calculam um conjunto de caminhos mais curtos. Segundo, eles respondem a falhas de rede ou mudanças de topologia atualizando continuamente as informações de roteamento. Portanto, quando pensamos em propagação de rota, é importante considerarmos o comportamento dinâmico dos protocolos e algoritmos.

Conceitualmente, protocolos de roteamento operam de forma independente dos mecanismos de encaminhamento. Isso que dizer que o software de protocolo de roteamento opera como um processo separado que usa o IP para troca de mensagens com o software do protocolo de roteamento em outros roteadores. Protocolos de roteamento pesquisam os destinos, calculam o menor caminho para cada destino e passam a informação para o software do protocolo de roteamento em outros roteadores.

Embora um protocolo de roteamento calcule o menor caminho, o software do protocolo de roteamento não armazena informação diretamente na tabela de encaminhamento do roteador. Ao contrário, o software de roteamento cria uma *base de informação de encaminhamento* (*Forwarding Information Base – FIB*). A FIB deve conter informações extras, não encontradas em uma tabela de encaminhamento, tais como a fonte da informação de roteamento, quanto tempo tem a informação (ou seja, a última vez que um protocolo de roteamento em outro roteador enviou uma mensagem sobre a rota) e se um gerente substituiu temporariamente uma rota específica.

Quando a FIB muda, o software de roteamento recalcula uma tabela de encaminhamento para o roteador e instala uma nova tabela. Uma etapa crucial acontece entre itens sendo colocados em uma FIB e os itens sendo propagados para a tabela de encaminhamento: as regras da política são aplicadas. As políticas permitem a um gerente controlar quais itens são

instalados automaticamente na tabela de encaminhamento. Dessa maneira, mesmo se o software de roteamento encontra um caminho menor para um destino particular e coloca a informação na FIB, é possível prevenir o caminho de ser adicionado na tabela de roteamento.

12.7 Roteamento por vetor de distância (Bellman-Ford)

O termo *vetor de distância* (*distance vector*)* refere-se a uma classe de algoritmos usados para propagar informações de roteamento. A ideia por trás dos algoritmos de vetor de distância é bastante simples. Cada roteador mantém uma lista de todas as rotas conhecidas em sua FIB. Quando inicia, um roteador inicializa sua FIB para conter uma entrada para cada rede diretamente conectada. Cada entrada na FIB identifica uma rede de destino, um roteador de próximo salto usado para alcançar o destino, e a "distância" até essa rede (de acordo com alguma medida de distância). Por exemplo, algum protocolo de vetor de distância usa o número de saltos de rede como uma medida de distância. Uma rede diretamente conectada está a zero salto de distância; se um datagrama precisa viajar através de N roteadores para alcançar um destino, o destino está a N saltos de distância. A Figura 12.4 ilustra o conteúdo inicial de uma FIB em um roteador que se conecta a duas redes. Na figura, cada entrada corresponde a uma rede diretamente conectada (zero salto distante).

Destino	Distância	Rota
Rede 1	0	direta
Rede 2	0	direta

Figura 12.4 Uma FIB inicial usada com um algoritmo de vetor de distância. Cada entrada contém o endereço IP de uma rede diretamente conectada e um inteiro que é a distância até a rede.

Quando se usa um vetor de distância, o software de roteamento em cada roteador envia uma cópia de sua FIB para qualquer outro roteador que ele possa alcançar diretamente. Logo que um informe chega ao roteador K, vindo do roteador J, K examina o conjunto de destinos relatados e a distância para cada um e aplica três regras descritas a seguir.

- Se *J* lista um destino que *K* não tem em sua FIB, *K* adiciona uma nova entrada à sua FIB com o próximo salto de *J*.
- Se *J* conhece um caminho mais curto para alcançar um destino *D*, *K* substitui o próximo salto em sua entrada FIB para *D* com *J*.
- Se a entrada FIB de *K* para o destino *D* já lista *J* como o próximo salto e a distância de *J* ao destino tiver mudado, *K* substitui a distância em sua entrada FIB.

A primeira regra pode ser interpretada da seguinte maneira: "se meu vizinho sabe um caminho para alcançar um destino que eu não sei, posso usar esse vizinho como um próximo salto". A segunda regra pode ser interpretada assim: "se meu vizinho tem uma rota menor para um destino, posso usar esse vizinho como próximo salto". A terceira regra pode ser interpretada deste modo: "se eu estou usando meu vizinho como próximo salto para um destino e o custo do meu vizinho para alcançar o destino muda, meu custo precisa mudar".

A Figura 12.5 mostra uma FIB existente em uma rota, *K*, e uma mensagem de atualização do vetor de distância de outro roteador, *J*. Três itens na mensagem causam mudanças na FIB. Note que, se *J* informa uma distância *N* saltos para um determinado destino e *K* usa *J* como um próximo salto, a distância armazenada na FIB *K* vai ter a distância $N + 1$ (ou seja, a distância de *J* para o destino mais a distância para alcançar *J*). A terceira coluna em nosso exemplo FIB é rotulada *rota*. Na prática, a coluna contém o endereço IP de um roteador próximo salto. Para tornar fácil de entender, a figura simplesmente lista um nome simbólico (por exemplo, *roteador J*).

Destino	Distância	Rota
Rede 1	0	direta
Rede 2	0	direta
Rede 4	8	Roteador L
Rede 17	5	Roteador M
Rede 24	6	Roteador J
Rede 30	2	Roteador Q
Rede 42	2	Roteador J

(a)

Destino	Distância
Rede 1	2
→ Rede 4	3
Rede 17	6
→ Rede 21	4
Rede 24	5
Rede 30	10
→ Rede 42	3

(b)

Figura 12.5 (a) Uma FIB existente em um roteador *K*, e (b) uma entrada de mensagem de atualização de roteamento do roteador *J* que vai causar mudanças.

O termo *vetor de distância* vem das informações enviadas nas mensagens periódicas. Uma mensagem contém uma lista de pares (D, V), em que D é a distância até um destino e V identifica o destino (chamado o *vetor*). Note que os algoritmos de vetor de distância informam rotas na primeira pessoa (por exemplo, nós pensamos em um anúncio de roteador, "Eu posso alcançar o destino V na distância D"). Nesse projeto, todos os roteadores precisam participar da troca do vetor de distância para que as rotas sejam eficientes e consistentes.

Embora sejam fáceis de implementar, os algoritmos de vetor de distância possuem desvantagens. Em um ambiente completamente estático, eles calculam os caminhos mais curtos e propagam corretamente rotas para todos os destinos. Quando as rotas mudam de forma rápida, no entanto, os cálculos podem não estabilizar. Quando uma rota muda (por exemplo, surge uma nova conexão ou uma conexão antiga falha), a informação se propaga lentamente de um roteador para outro. Enquanto isso, alguns roteadores podem ter informações de roteamento incorretas.

12.8 Confiabilidade e protocolos de roteamento

A maioria dos protocolos de roteamento usa transporte sem conexão – os primeiros protocolos encapsulavam mensagens diretamente no IP; os protocolos de roteamento modernos normalmente encapsulam no UDP.* Infelizmente, o UDP oferece a mesma semântica do IP: as mensagens podem ser perdidas, atrasadas, duplicadas, corrompidas ou entregues fora

de ordem. Assim, um protocolo de roteamento que os utiliza precisa compensar as falhas.

Os protocolos de roteamento usam várias técnicas para cuidar de confiabilidade. Primeiro, os checkusums são usados para lidar com corrupção de dados. A perda é tratada por *soft state*** ou através de confirmação e retransmissão. *Números sequenciais* são usados para resolver dois problemas. Primeiro, eles permitem a um receptor lidar com entrega fora de ordem colocando mensagens de entrada de volta na ordem correta. Segundo, podem ser usados para lidar com *replay*, uma condição que pode ocorrer se uma duplicata de uma mensagem se atrasa e chega muito depois de um update mais novo ter sido processado. O Capítulo 14 ilustra como protocolos de vetor de distância podem exibir convergência lenta e discute técnicas adicionais que eles usam para evitar problemas. Em particular, o capítulo cobre s*plit horizon* e técnicas poison reverse.

12.9 Roteamento por estado do link (SPF)

A principal desvantagem do algoritmo de vetor de distância é que ele não escala bem. Além do problema da resposta lenta mudar, mencionado anteriormente, o algoritmo requer a troca de mensagens grandes – como cada atualização de roteamento contém uma entrada para cada rede possível, o tamanho da mensagem é proporcional ao número total de redes em uma internet. Além disso, como um protocolo de vetor de distância exige que cada roteador participe, o volume das informações trocadas pode ser enorme.

A principal alternativa aos algoritmos de vetor de distância é uma classe de algoritmos conhecida como *estado de link* (*link state*) *link status* ou *Caminhos Mais Curtos Primeiro* (*Shortest Path First – SPF*).*** O algoritmo SPF exige que cada roteador participante calcule informações de topologia. A maneira mais fácil de pensar nas informações de topologia é imaginar que cada roteador possui um mapa que mostra todos os outros roteadores e redes às quais se conectam. Em termos abstratos, os roteadores correspondem a pontos em um gráfico e as redes que conectam roteadores correspondem a linhas. Existe uma linha (link) entre dois pontos se, e apenas se, os roteadores correspondentes podem se comunicar diretamente.

Em vez de enviar mensagens que contenham listas de destinos que um roteador possa alcançar, cada roteador participante em um algoritmo SPF desempenha duas funções descritas a seguir.

- Testa ativamente o estado de todos os roteadores vizinhos. Dois roteadores são considerados vizinhos se eles se conectam a uma rede comum.
- Periodicamente, transmite por broadcast mensagens de estado do link (link-state) no seguinte formato: “o link entre mim e o roteador X está ativo” ou “o link entre mim e o roteador X está inativo”.

Para testar o estado de um vizinho diretamente conectado, os dois vizinhos trocam mensagens curtas que verificam se o outro vizinho está ativo e acessível. Se o vizinho responder, dizemos que o link entre eles está ativo (*up*). Caso contrário, afirmamos que o link está inativo (*down*).* Para informar todos os outros roteadores, cada roteador difunde por broadcast periodicamente uma mensagem que relata o status (estado) de cada um dos seus links. Uma mensagem de estado não especifica rotas – ela simplesmente informa se a comunicação é possível entre pares de roteadores. Quando se usa o algoritmo estado do link, o software deve encontrar um caminho para entregar cada mensagem de estado de link para todos os roteadores, mesmo se a rede subjacente não suportar broadcast. Então, cópias individuais de cada mensagem podem ser encaminhadas ponto a ponto.

Sempre que uma mensagem de estado de link chega, o software em execução no roteador usa as informações para atualizar seu mapa da internet. Primeiro, ele extrai o par de roteadores mencionado na mensagem e se certifica de que o gráfico local contém uma linha entre os dois. Segundo, ele usa o status informado na mensagem para marcar o link como ativo ou inativo. Sempre que a chegada de uma mensagem causa uma mudança na topologia local do gráfico, o algoritmo de estado de link recalcula rotas aplicando o algoritmo de Dijkstra. Ele calcula o menor caminho do roteador local para cada destino. A informação resultante é colocada na FIB e, se a política permitir, usada para mudar a tabela de encaminhamento.

Uma das principais vantagens dos algoritmos SPF é que cada roteador calcula rotas independentemente usando os mesmos dados de estado originais; eles não dependem do cálculo das rotas intermediárias. Compare o método com o algoritmo de vetor de distância no qual cada roteador atualiza sua FIB e então envia a informação atualizada aos vizinhos – se o software em qualquer roteador ao longo do caminho estiver incorreto, todos os roteadores sucessivos vão receber informação incorreta. Com um algoritmo de estado de link, os roteadores não dependem de cálculos intermediários – a mensagem de estado de link se propaga através da rede

sem alteração, tornando mais fácil depurar problemas. Como cada roteador realiza localmente o cálculo do caminho mais curto, existe a garantia de esse cálculo convergir. Finalmente, como cada mensagem de estado de link transporta informações apenas sobre a conexão direta entre um par de roteadores, o tamanho não depende do número de redes na internet subjacente. Assim, os algoritmos SPF escalam melhor do que os algoritmos de vetor de distância.

12.10 Resumo

Para garantir que todas as redes permaneçam acessíveis com alta confiabilidade, uma internet precisa fornecer encaminhamento consistente de forma global. Os hosts e a maioria dos roteadores contêm apenas informações parciais de roteamento; eles dependem de rotas-padrão para enviar datagramas a destinos distantes. Originalmente, a Internet global resolveu o problema do roteamento usando uma arquitetura de roteadores core em que um conjunto de roteadores core continham informações completas sobre todas as redes.

Quando redes de backbone adicionais foram acrescentadas à Internet, uma nova arquitetura de roteamento surgiu para adequar a topologia estendida. Atualmente, existe um conjunto de redes backbone peer gerenciadas separadamente que se interconectam em vários lugares.

Quando trocam informações de roteamento, os roteadores normalmente usam um de dois algoritmos básicos: vetor de distância ou estado de link (também chamado SPF). A principal desvantagem dos algoritmos de vetor de distância é que eles realizam um cálculo de caminho mais curto distribuído que pode não convergir se o status das conexões de rede mudar continuamente. Portanto, para grandes internets, ou para internets em que a topologia subjacente muda rapidamente, os algoritmos SPF são superiores.

EXERCÍCIOS

12.1 Suponha que um roteador descubra que está para encaminhar um datagrama IP de volta pela mesma interface de rede em que o datagrama chegou. O que ele deve fazer? Por quê?

12.2 Após ler a RFC 823 e a RFC 1812, explique o que um roteador core da Internet (por exemplo, um com informações de roteamento completas) deve fazer na situação descrita na questão anterior.

12.3 Como os roteadores em um sistema core usam rotas-padrão para enviar todos os datagramas inválidos para uma máquina específica?

12.4 Imagine que um gerente acidentalmente configura um roteador para anunciar que possui conexões diretas com seis redes específicas quando, na verdade, não possui. Como outros roteadores que recebem o anúncio se protegem de anúncios inválidos enquanto continuam aceitando outras atualizações dos roteadores "não confiáveis"?

12.5 Que mensagens ICMP um roteador gera?

12.6 Considere que um roteador esteja usando transporte não confiável para entrega. Como o roteador pode determinar se um vizinho designado está "ativo" ou "inativo"? (Dica: consulte a RFC 823 para descobrir como o sistema core original resolveu o problema.)

12.7 Suponha que dois roteadores anunciem o mesmo custo, k, para alcançar uma determinada rede, N. Descreva as circunstâncias sob as quais o encaminhamento através de um deles pode precisar de menos saltos totais do que encaminhar através do outro.

12.8 Como um roteador sabe se um datagrama que chega transporta uma mensagem de atualização de roteamento?

12.9 Considere cuidadosamente a atualização de vetor de distância mostrado na Figura 12.5. Para cada item atualizado na tabela, dê o motivo por que o roteador realizará a atualização.

12.10 Considere o uso dos números de sequência para garantir que dois roteadores não se tornem confusos quando datagramas são duplicados, atrasados ou entregues fora de ordem. Como os números de sequência iniciais devem ser selecionados? Por quê?

---

* Felizmente, as placas estão impressas em um idioma que você pode ler.

* Os termos *distância-vetor*, *Ford-Fulkerson*, *Bellman-Ford* e *Bellman* são sinônimos de *vetor de distância*, os últimos três tirados dos nomes dos pesquisadores que publicaram a ideia.

* O próximo capítulo discute uma exceção: um protocolo de roteamento que usa TCP.

** Lembre que soft state se baseia em timeouts para remover informações antigas.

*** Dijkstra, o inventor do algoritmo, cunhou o nome "*Caminhos Mais Curtos Primeiro*" (Shortest Path First), mas ele é enganoso, pois todos os algoritmos de roteamento calculam caminhos mais curtos.

* Na prática, para prevenir oscilações entre os estados ativo e inativo, muitos protocolos usam uma regra *k-out-of-n (k-out-of-n rule)* para testar atividade,

significando que o link permanece ativo até que uma porcentagem significativa de solicitações não tenha resposta, e então ele permanece inativo até que uma porcentagem significativa de mensagens receba uma resposta.

CAPÍTULO 13

Roteamento entre sistemas autônomos (BGP)

CONTEÚDOS DO CAPÍTULO

13.1 Introdução

O capítulo anterior introduz a ideia da propagação de rota. Este capítulo estende nossa compreensão das arquiteturas de roteamento da internet e discute o conceito dos sistemas autônomos. Vamos ver que sistemas autônomos correspondem a grandes ISP ou grandes empresas e que cada sistema autônomo é composto por um grupo de redes e roteadores operando sob uma autoridade administrativa

O tópico central do capítulo é um protocolo de roteamento chamado BGP que é usado para fornecer roteamento entre sistemas autônomos. O BGP é um protocolo de roteamento chave usado no centro da Internet para permitir que cada ISP principal informe outros pares sobre os destinos que ele pode alcançar.

13.2 O escopo do protocolo de atualização de roteamento

Um princípio fundamental cria o projeto de uma arquitetura de roteamento: nenhum protocolo de atualização de roteamento pode escalar para permitir que todos os roteadores na Internet global troquem informações de roteamento. Em vez disso, os roteadores precisam ser divididos em grupos separados e protocolos de roteamento devem ser projetados para operar dentro do grupo. Existem três razões por que roteadores devem ser separados que são descritas a seguir.

- *Tráfego.* Mesmo que cada site consista de uma única rede, nenhum protocolo de roteamento poderá acomodar um número arbitrário de sites porque acrescentar sites aumenta o tráfego de roteamento – se o conjunto de roteadores for grande demais, o tráfego de roteamento começa a ficar sobrecarregado. O protocolo de vetor de distância exige que os roteadores troquem o conjunto inteiro de redes (ou seja, o tamanho de cada atualização é proporcional ao tamanho da tabela de encaminhamento). Os protocolos de estado de link transmitem por broadcast avisos de conectividade entre pares de roteadores (isto é, os broadcasts vão viajar através da Internet).
- *Comunicação indireta.* Como não compartilham uma rede comum na Internet global, não podem comunicar-se diretamente. Em alguns casos, um roteador está muito longe do centro da Internet, o que significa que o único meio que o roteador tem para alcançar todos os outros roteadores é através de vários saltos intermediários.
- *Limites administrativos.* Na Internet, as redes e roteadores não são propriedade e nem gerenciados por uma única entidade. Mais importante, caminhos mais curtos não são sempre usados! Ao contrário, grandes ISP utilizam rotas de tráfego ao longo dos caminhos que geram receita ou têm menor custo financeiro. Portanto, uma arquitetura de roteamento deve fornecer um meio para cada grupo administrativo controlar o roteamento e acessá-lo de forma independente.

As consequências de limitar a interação de roteadores são significativas. A ideia fornece o motivo para grande parte da arquitetura de roteamento usada na Internet, e explica alguns dos mecanismos que vamos estudar. Resumindo, a seguir, esse importante princípio.

> *Embora seja desejável que os roteadores troquem informação de roteamento, é impraticável para todos os roteadores em uma internet arbitrariamente grande, tal como a Internet global, participarem em um único protocolo de atualização de roteamento.*

13.3 Determinando um limite prático para o tamanho de grupo

A discussão anterior deixa muitas questões em aberto. Por exemplo, que tamanho de internet é considerado "grande"? Se apenas um conjunto limitado de roteadores pode participar em uma troca de informações de roteamento, o que acontece com os roteadores que são excluídos? Eles funcionam corretamente? Um roteador que não seja participante pode encaminhar um datagrama para outro roteador que esteja participando? Um roteador participante pode encaminhar um datagrama para um roteador não participante?

A resposta à pergunta do tamanho envolve o entendimento do tráfego que um protocolo específico vai gerar, a capacidade da rede que conecta os roteadores e outros requisitos, tais como se o protocolo exige broadcast e os detalhes do protocolo de roteamento. Existem dois problemas: atraso e sobrecarga (overhead).

O atraso é fácil de entender. Por exemplo, considere o atraso máximo até que todos os roteadores estejam informados de uma mudança quando usarem um protocolo de vetor de distância. Cada roteador precisa receber a nova informação, atualizar sua tabela de roteamento e, então, encaminhar a informação para seus vizinhos. Em uma inter-rede com N roteadores organizados em uma topologia linear, N etapas são necessárias. Portanto, N precisa ser limitado para garantir rápida distribuição das informações.

Atraso. O principal atraso de interesse não é quanto tempo que leva uma única mensagem de atualização de roteamento para se propagar. Em vez disso, a questão diz respeito a tempo de convergência, o máximo atraso até que todos os roteadores sejam informados sobre a mudança. Quando os roteadores usam um protocolo de vetor de distância, cada um deve receber a nova informação, atualizar sua FIB e, então, encaminhar a resposta para seus vizinhos. Em uma internet com N roteadores organizados em uma topologia linear, são necessárias N etapas. Então, N deve ser limitado para garantir rápida distribuição de informação.

Overhead. O problema de overhead também é fácil de entender. Como cada roteador que participa em um protocolo de roteamento deve enviar mensagens, um conjunto grande de roteadores participantes significa mais tráfego de roteamento. Além disso, se as mensagens de roteamento contiverem uma lista de destinos possíveis, o tamanho de cada mensagem crescerá com o aumento de roteadores na rede. Para garantir que o tráfego de roteamento continue sendo uma pequena porcentagem do tráfego total nas redes subjacentes, o tamanho das mensagens de roteamento deve ser limitado.

Na verdade, poucos gerentes de rede têm informação suficiente sobre protocolos de roteamento para uma análise detalhada do atraso ou do overhead. Em vez disso, eles seguem uma diretriz heurística.

> *É seguro permitir que até doze roteadores participem de um único protocolo de informações de roteamento através de uma rede remota; aproximadamente cinco vezes esse número pode participar seguramente em um conjunto de redes locais.*

Naturalmente, a regra dá apenas uma orientação geral e há muitas exceções. Por exemplo, se as redes subjacentes possuírem atraso especialmente baixo e alta capacidade, o número de roteadores participantes poderá ser maior. Da mesma forma, se as redes subjacentes tiverem capacidade atipicamente baixa ou uma alta quantidade de tráfego, o número de roteadores participantes precisará ser menor para evitar sobrecarga das redes com tráfego de roteamento.

Como uma internet não é estática, pode ser difícil estimar quanto tráfego os protocolos de roteamento irão gerar ou que porcentagem da largura de banda subjacente o tráfego de roteamento irá consumir. Por exemplo, à medida que o tráfego de roteamento cresce com o tempo, os aumentos no tráfego gerado consomem mais da capacidade de reserva da rede. Além disso, novas aplicações podem gerar aumento no tráfego. Desse modo, os gerentes de rede não podem se basear unicamente no princípio anteriormente citado ao escolher uma arquitetura de roteamento. Em vez disso, eles normalmente implementam um esquema de *monitoramento de tráfego*. Basicamente, um monitor de tráfego escuta passivamente uma rede e registra estatísticas sobre o tráfego. Em particular, um monitor pode calcular a utilização de rede (ou seja, a porcentagem da largura de banda subjacente sendo usada) e a porcentagem de pacotes transportando mensagens de protocolo de roteamento. Um gerente pode observar as tendências do tráfego fazendo medições por longos períodos (por exemplo, semanas ou meses), e pode usar a saída para determinar se muitos roteadores estão participando em um único protocolo de roteamento.

13.4 Uma ideia fundamental: saltos extras

Embora o número de roteadores que participam em um único protocolo de

roteamento precise ser limitado, fazer isso tem uma importante consequência porque significa que alguns roteadores estarão fora do grupo. Por exemplo, considere uma intranet corporativa com um backbone e um conjunto de roteadores que participam, todos, em um protocolo de atualização de roteamento. Suponha que um novo departamento é adicionado à rede e adquira um roteador. Pode parecer que o novo roteador não precisará participar de um protocolo de atualização de roteamento – o "externo" pode simplesmente usar um membro do grupo como padrão.

A mesma situação ocorreu no início da Internet. Como novos sites foram adicionados, o sistema core funcionava como um mecanismo de roteamento central ao qual os roteadores não core enviavam datagramas para entrega. Os pesquisadores descobriram uma importante lição: se um roteador fora de um grupo usar um membro do grupo como uma rota-padrão, o roteamento ficará abaixo do ótimo. Mais importante, não é necessário um grande número de roteadores ou uma rede remota – o problema pode ocorrer mesmo em uma pequena corporação na qual um roteador não participante usa um roteador participante para entrega.

Para entender como ocorre o encaminhamento abaixo do ótimo, considere a configuração de rede do exemplo na Figura 13.1.

Figura 13.1 Exemplo de uma arquitetura que pode causar o problema do salto extra se um roteador não participante usa um roteador participante como padrão para próximo salto.

Na figura, os roteadores R_1 e R_2 conectam-se às redes de área local *1* e *2*, respectivamente. Cada roteador participa em um protocolo de roteamento e sabe como alcançar ambas as redes. Um novo roteador, R_3, é adicionado. Em vez de configurar R_3 para participar do protocolo de atualização de roteamento, o gerente o configura para usar um dos roteadores existentes, ou seja, R_1, como padrão.

Se R_1 tem um datagrama destinado para um host na rede local *1*, não há problema. Entretanto, se *R3* tem um datagrama destinado para a rede local *2*, ele o envia através do backbone para R_1, que deverá então encaminhar o datagrama de volta através do backbone para o roteador R_2. A rota ótima, claro, requer que R_3 envie datagramas destinados para a rede *2* diretamente para R_2. Note que a escolha de um roteador participante não faz diferença: somente destinos que ficam além do roteador escolhido têm rotas ótimas.

Para todos os outros destinos, um datagrama fará uma segunda viagem, ainda que desnecessária, através da rede backbone. Também lembre que roteadores não podem usar mensagens ICPM redirecionadas para informar R_3 de que ele tem rota não ótima, pois tais mensagens só podem ser enviadas para a origem, e não para roteadores intermediários.

Chamamos a anomalia de roteamento ilustrada na Figura 13.1 de problema do salto extra. O problema é insidioso porque tudo parece funcionar corretamente – os datagramas realmente alcançam seu destino. Entretanto, como o roteamento não é ótimo, o sistema é extremamente ineficaz. Cada datagrama que leva um salto extra consome recursos no roteador intermediário, bem como o dobro da largura de banda do backbone que deveria. O problema do salto extra foi primeiramente descoberto no início da Internet. Resolvê-lo exigiu mudarmos nossa visão da arquitetura conforme descrito a seguir.

Tratar um grupo de roteadores que participa de um protocolo de atualização de roteamento como um sistema de entrega padrão pode introduzir um salto extra para o tráfego de datagrama; é necessário um mecanismo que permita a roteadores não participantes aprenderem rotas dos roteadores participantes de modo que possam escolher rotas ótimas.

13.5 Conceito de sistema autônomo

Como a Internet deveria ser dividida em conjuntos de roteadores que podem executar um protocolo de atualização de roteamento? A chave para a resposta reside em observar que a Internet não consiste em redes

independentes. Ao contrário, as redes e os roteadores são pertencentes a organizações e indivíduos. Como as redes e os roteadores de uma determinada entidade estão sob uma única autoridade administrativa, a autoridade pode garantir que as rotas internas permaneçam consistentes e viáveis. Além disso, a autoridade administrativa pode escolher um ou mais de seus roteadores para avaliar o mundo exterior das redes dentro da organização e aprender sobre redes que são externas à organização.

Para fins de roteamento, um grupo de redes e roteadores controlado por uma única autoridade administrativa é chamado um *Sistema Autônomo* (*AS*). A ideia é que deixaremos cada sistema autônomo escolher seu próprio mecanismo para descobrir, propagar, validar e verificar a consistência das rotas (o próximo capítulo examina alguns dos protocolos que os sistemas autônomos usam para propagar informações de roteamento internamente). Assim, vamos organizar meios para um sistema autônomo resumir informação de roteamento e enviá-lo para o sistema autônomo vizinho.

O sistema autônomo é um ISP? Ele pode ser, mas também pode ser uma grande empresa (isto é, uma grande corporação ou universidade). Embora a definição de sistema autônomo talvez pareça vaga, ela tem a intenção de incluir praticamente qualquer grupo que utiliza uma grande rede. Claro, os limites entre sistemas autônomos devem ser definidos precisamente para permitir que algoritmos automatizados tomem decisões de roteamento, e prevenir os protocolos de atualização de roteamento usados em um sistema autônomo de vazamentos acidentais entre um e outro. Além disso, cada sistema autônomo define um conjunto de políticas. Por exemplo, um sistema autônomo pode preferir evitar rotear pacotes através de sistemas autônomos de concorrentes, mesmo que tal caminho exista. A fim de tornar isso possível, para algoritmos de roteamento automático distinguirem entre sistemas autônomos, a cada um é atribuído um *número de sistema autônomo* pela autoridade central que é encarregada pela atribuição de todos os números da Internet. Quando roteadores em dois sistemas autônomos trocam informação de roteamento, o protocolo se organiza para que cada um aprenda o número do sistema autônomo do outro.

Podemos resumir a ideia conforme descrito a seguir.

A Internet é dividida em sistemas autônomos que são, cada um, pertencentes a uma única autoridade administrativa e operados por

ela. Um sistema autônomo é livre para escolher uma arquitetura de roteamento interna e protocolos.

Na prática, embora algumas grandes organizações tenham obtido números de sistema autônomo para permitir-lhes conectarem-se a múltiplos ISPs, é mais fácil pensarmos em cada sistema autônomo como correspondente a um grande ISP. Uma importante questão é colocada a seguir.

Na Internet atual, cada grande ISP é um sistema autônomo. Durante discussões informais, os engenheiros normalmente usam a expressão "roteamento entre grandes ISPs" quando querem se referir ao roteamento entre sistemas autônomos.

13.6 Protocolos de gateway exterior e alcançabilidade

Dissemos que um sistema autônomo deve resumir informação de seu protocolo de atualização de roteamento e propagar essa informação para outros sistemas autônomos. Para isso, um sistema autônomo configura um ou mais de seus roteadores para se comunicar com roteadores em outros sistemas autônomos. A informação flui em duas direções. Primeiro, o roteador deve coletar informação sobre redes dentro de seu sistema autônomo e passá-la para fora. Segundo, o roteador deve aceitar informação sobre redes em outros sistemas autônomos e disseminar a informação internamente. Tecnicamente, dizemos que o sistema autônomo anuncia a *alcançabilidade da rede* para o mundo externo, e usamos o termo *Exterior Gateway Protocol** (*EGP*) para indicar qualquer protocolo usado para passar informações de alcançabilidade entre dois sistemas autônomos. Estritamente falando, um EGP não é um protocolo de roteamento, porque anunciar alcançabilidade não é o mesmo que propagar informações de roteamento. Na prática, porém, a maioria dos profissionais de rede não faz distinção – provavelmente você ouvirá alguém falar de protocolos de gateway exterior querendo se referir a protocolos de roteamento.

Atualmente, um único EGP é usado para trocar informações de alcançabilidade na Internet. Conhecido como *Border Gateway Protocol*

(*BGP*), ele evoluiu através de quatro (bastante diferentes) versões. Cada versão é numerada, o que deu origem ao nome formal da versão atual: *BGP-4*. Seguindo a prática-padrão na indústria de redes, usaremos o termo *BGP* no lugar de *BGP-4*.

Quando um par de sistemas autônomos concorda em usar BGP para trocar informações de roteamento, cada um precisa designar um roteador** que falará BGP em seu nome; dizemos que os dois roteadores se tornam *pares* (*peers*) *de BGP* um do outro. Como um roteador falando BGP precisa se comunicar com um peer em outro sistema autônomo, faz sentido selecionar uma máquina que esteja próxima da "borda" do sistema autônomo. Assim, a terminologia do BGP chama o roteador de um *gateway de borda* ou *roteador de borda*. A Figura 13.2 ilustra a ideia.

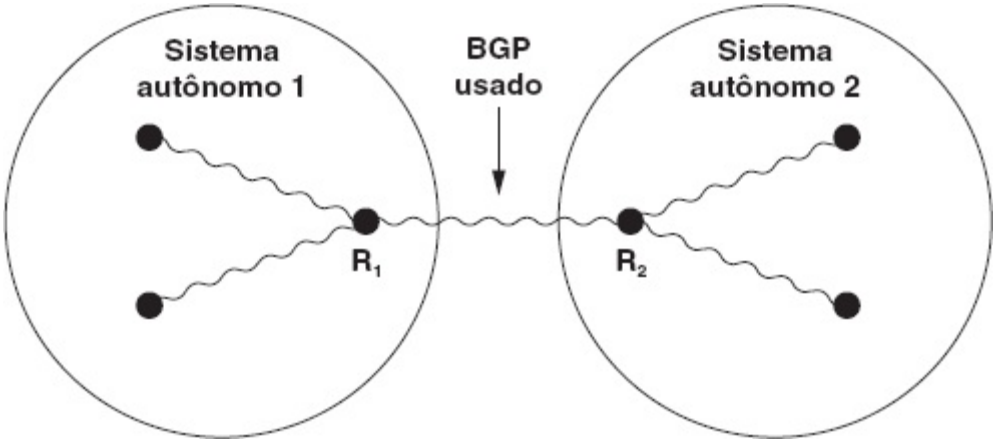


Figura 13.2 Ilustração conceitual de BGP usado entre o roteador R_1 em um sistema autônomo e o roteador R_2 em outro sistema autônomo.

Na figura, as informações fluem em ambas as direções. O roteador R_1 reúne informação sobre redes em sistema autônomo *1* e usa BGP para relatar a informação ao roteador R_2, enquanto o roteador R_2 reúne informação sobre redes no sistema autônomo *2* e usa BGP para relatar a informação para o roteador R_1.

13.7 Características do BGP

O BGP é incomum em vários sentidos. Mais importante, como ele anuncia a alcançabilidade em vez de informações de roteamento, o BGP não usa o algoritmo de vetor de distância nem o algoritmo de estado de link. Em vez disso, o BGP usa uma modificação conhecida como algoritmo *vetor caminho* (*path-vetor*). BGP é caracterizado pelos itens a seguir.

Comunicação de sistema interautônomo. Em virtude de o BGP ser projetado como um exterior gateway protocol, sua principal função é permitir que um sistema autônomo se comunique com outro.

Entre vários interlocutores BGP (BGP speakers). Se um sistema autônomo tiver vários roteadores, cada um se comunicando com um peer em um sistema autônomo externo, uma forma de BGP conhecida como *iBGP* pode ser usada para coordenar entre os roteadores no conjunto para garantir que todos eles propaguem informações consistentes.

Propagação de informações de alcançabilidade. O BGP permite que um sistema autônomo anuncie destinos que são acessíveis nele ou através dele, e aprende essas informações de outro sistema autônomo.

Modelo do próximo salto. Assim como os protocolos de roteamento vetor de distância, o BGP fornece informações de *próximo salto* para cada destino.

Suporte à política. Diferentemente dos protocolos de vetor de distância que anunciam exatamente as rotas na tabela de encaminhamento local, o BGP pode implementar políticas escolhidas pelo administrador local. Em especial, um roteador executando o BGP pode ser configurado para distinguir entre o conjunto de destinos acessíveis pelos computadores dentro de seu sistema autônomo e o conjunto de destinos anunciados a outros sistemas autônomos.

Transporte confiável. O BGP é incomum entre os protocolos que passam informações de roteamento porque ele considera o transporte confiável. Portanto, o BGP usa o TCP para toda comunicação.

Informações de caminho. Em vez de especificar destinos que possam ser alcançados e um próximo salto para cada um, o BGP usa um modelo *vetor caminho* no qual anúncios especificam informações de caminho que permitem a um receptor aprender uma série de sistemas autônomos ao longo de um caminho até o destino.

Atualizações incrementais. Para conservar largura de banda da rede, o BGP não passa informações completas em cada mensagem de atualização. Em vez disso, as informações completas são trocadas uma vez e, depois, mensagens sucessivas carregam mudanças incrementais chamadas *deltas*.

Suporte para IPv4 e IPv6. OBGP suporta endereços IPv4 classless e endereços IPv6. Ou seja, o BGP envia um tamanho de prefixo junto com cada endereço.

Agregação de rota. O BGP conserva largura de banda da rede permitindo que um emissor agregue informações de rota e envie uma única entrada para representar múltiplos destinos relacionados (isto é, várias redes sendo propriedade de um único AS).

Autenticação. O BGP permite que um receptor autentique mensagens (ou seja, verifique a identidade de um emissor).

13.8 Funcionalidade e tipos de mensagem BGP

Os peers BGP desempenham três funções básicas. A primeira delas consiste na aquisição e na autenticação inicial do peer. Os dois peers estabelecem uma conexão TCP e realizam uma troca de mensagens que garanta que os dois lados concordaram em se comunicar. A segunda função forma o foco principal do protocolo – cada lado envia informações de alcançabilidade positiva ou negativa. Ou seja, um emissor pode anunciar que um ou mais destinos estão acessíveis dando um próximo salto para cada um, ou o emissor pode declarar que um ou mais destinos previamente anunciados não estão mais acessíveis. A terceira função fornece verificação constante se os peers e as conexões de rede entre eles estão funcionando corretamente.

Para manipular as três funções descritas anteriormente, o BGP define cinco tipos de mensagem básicos. A Figura 13.3 contém um resumo.

Código de tipo	Tipo de mensagem	Descrição
1	OPEN	Inicializa comunicação.
2	UPDATE	Anuncia ou retira rotas.
3	NOTIFICATION	Resposta a uma mensagem incorreta.
4	KEEPALIVE	Teste ativo da conectividade peer.
5	REFRESH	Requisita novo anúncio do peer.

Figura 13.3 Os cinco tipos de mensagem básicos no BGP.

13.9 Cabeçalho de mensagem do BGP

Cada mensagem BGP começa com um cabeçalho fixo que identifica o tipo de mensagem. A Figura 13.4 ilustra o formato do cabeçalho.

Figura 13.4 O formato do cabeçalho que precede cada mensagem BGP.

Na figura, o campo MARCADOR de 16 octetos contém um valor que ambos os lados concordam em usar para marcar o início de uma mensagem. O campo de TAMANHO de 2 octetos especifica o tamanho total da mensagem medido em octetos. O tamanho mínimo de mensagem é de *19* octetos (para um tipo de mensagem que tem apenas um cabeçalho e não há dados após o cabeçalho), e o tamanho máximo permissível é de *4096* octetos. Finalmente, o campo de TIPO de 1 octeto contém um dos cinco valores para o tipo de mensagem listada na Figura 13.3.

O campo MARCADOR parece incomum. Na mensagem inicial, o marcador consiste em somente 1s; se os peers concordarem em usar um mecanismo de autenticação, o marcador pode conter informações de autenticação. Em qualquer caso, os dois lados precisam concordar no valor para que ele possa ser usado para *sincronização*. Para entender por que a sincronização é necessária, lembre-se de que todas as mensagens BGP são trocadas através de um transporte de fluxo (isto é, TCP), que não identifica o limite entre uma mensagem e a seguinte. Nesse ambiente, um simples erro em qualquer lado pode ter consequências dramáticas. Em particular, se o emissor ou o receptor contar erroneamente os octetos em uma mensagem, um *erro de sincronização* irá ocorrer.

Mais importante, como o protocolo de transporte não especifica limites da mensagem, ele não alertará o receptor para o erro. Portanto, para garantir que o emissor e o receptor permaneçam sincronizados, o BGP coloca um valor de sequência bem conhecido no início de cada mensagem e exige que um receptor verifique que o valor está intacto antes de processar a

mensagem.

13.10 Mensagem BGP OPEN

Assim que dois peers BGP estabelecem uma conexão TCP, cada um deles envia uma mensagem OPEN para declarar o número do seu sistema autônomo e estabelecer outros parâmetros operacionais. Além do cabeçalho-padrão, uma mensagem OPEN contém um valor para um *timer de espera* que é usado para especificar o número máximo de segundos que podem decorrer entre o recebimento de duas mensagens sucessivas. A Figura 13.5 ilustra o formato.

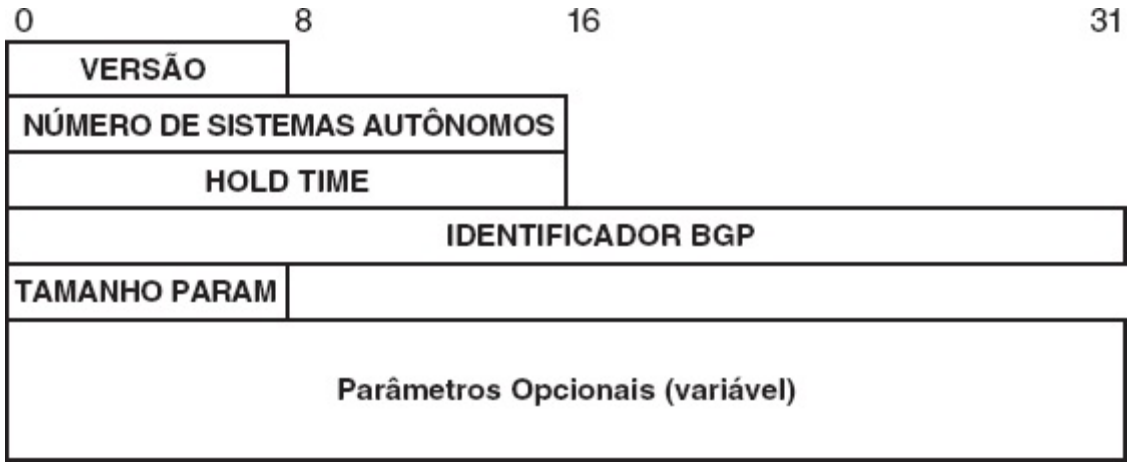


Figura 13.5 Formato da mensagem BGP OPEN IPv4 que é enviada na inicialização. Os octetos mostrados na figura seguem o cabeçalho de mensagem padrão.

A maioria dos campos na figura é simples. O campo VERSÃO identifica a versão do protocolo usada (a versão para o formato mostrado é a versão 4; uma seção mais à frente discute a extensão BGP para o IPv6). Lembre-se de que a cada sistema autônomo é atribuído um número único. O campo NÚMERO DOS SISTEMAS AUTÔNOMOS informa o número do sistema autônomo do sistema do emissor. O campo TEMPO DE ESPERA especifica um tempo máximo que o receptor deve esperar por uma mensagem do emissor. É exigido que o receptor implemente um timer usando o valor especificado. O timer é redefinido toda vez que uma mensagem chega; se o timer expira, o receptor considera que o emissor não está mais disponível (e para de encaminhar datagramas ao longo de rotas recebidas do emissor).

O campo IDENTIFICADOR BGP contém um valor inteiro de 32 bits que identifica unicamente o emissor. Se uma máquina possui múltiplos peers

(por exemplo, talvez em vários sistemas autônomos), deve usar o mesmo identificador em toda a comunicação. O protocolo especifica que o identificador é um endereço IP. Portanto, um roteador precisa escolher um de seus endereços IPv4 para usar com todos os peers BGP.

O último campo de uma mensagem OPEN é opcional. Se presente, o campo TAMANHO PARAM especifica o tamanho medido em octetos, e o campo chamado PARÂMETROS OPCIONAIS contém uma lista dos parâmetros. Ele foi rotulado como *variável* para indicar que o tamanho varia de mensagem para mensagem. Quando os parâmetros estão presentes, cada parâmetro na lista é precedido por um cabeçalho de 2 octetos, com o primeiro octeto especificando o tipo do parâmetro, e o segundo especificando o tamanho. Se não houver parâmetros, o valor de TAMANHO PARAM é zero e a mensagem termina sem dados posteriores.

Ao aceitar uma mensagem que chega com entrada OPEN, um roteador de borda falando BGP responde enviando uma mensagem KEEPALIVE (discutida adiante). Cada peer precisa enviar uma mensagem OPEN e receber uma mensagem KEEPALIVE antes que possam trocar informações de roteamento. Portanto, uma mensagem KEEPALIVE funciona como confirmação para uma mensagem OPEN.

13.11 Mensagem BGP UPDATE

Uma vez que peers BGP tenham criado uma conexão TCP, enviado mensagens OPEN e as reconhecido, os peers usam mensagens UPDATE para anunciar novos destinos que são acessíveis ou para retirar anúncios anteriores quando um destino se tornou inalcançável. A Figura 13.6 ilustra o formato das mensagens UPDATE.

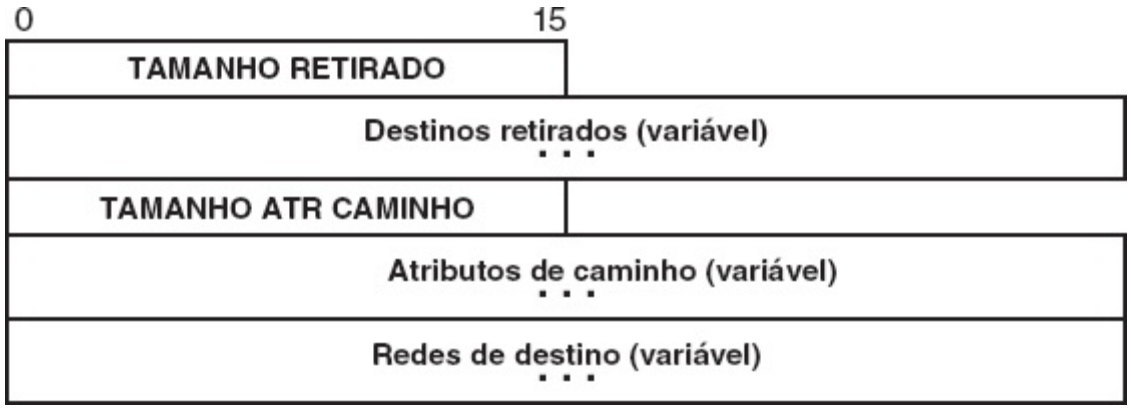


Figura 13.6 Formato da mensagem BGP UPDATE em que áreas de tamanho variável da mensagem podem ser omitidas. Esses octetos seguem o cabeçalho de mensagem-padrão.

Como mostra a figura, cada mensagem UPDATE é dividida em duas partes: a primeira lista destinos anteriormente anunciados que estão sendo retirados e a segunda especifica novos destinos sendo anunciados. A segunda parte lista os atributos dos caminhos seguidos por um conjunto de redes de destino que usam esse atributo. Os campos rotulados como variável não possuem um tamanho fixo. Na verdade, campos de tamanho variado não precisam estar presentes – se a informação não é necessária para uma determinada ATUALIZAÇÃO, o campo correspondente é omitido da mensagem. O campo TAMANHO RETIRADO é de 2 octetos, que especificam o tamanho do campo *destinos retirados* que segue. Se nenhum destino estiver sendo retirado, TAMANHO RETIRADO contém zero. Da mesma forma, o campo TAMANHO ATR CAMINHO especifica o tamanho dos *atributos de caminho* que estão associados a novos destinos sendo anunciados. Se não houver novos destinos, o campo TAMANHO ATR CAMINHO contém zero.

13.12 Pares máscara-endereço IPv4 compactados

Os campos DESTINOS RETIRADOS e REDES DE DESTINO podem conter uma lista de endereços de rede IPv4. Conceitualmente, o BGP poderia enviar uma máscara de endereço com cada endereço IPv4. Em vez de enviar um endereço e uma máscara como quantidades de 32 bits separadas, o BGP usa uma representação compactada para reduzir o tamanho da mensagem. A Figura 13.7 ilustra o formato.

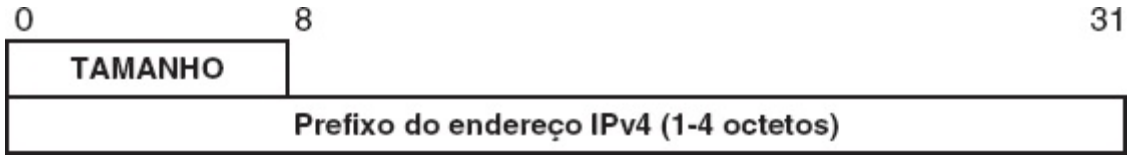


Figura 13.7 O formato compactado que o BGP usa para armazenar um endereço IPv4 de destino e a máscara de endereço associada.

Como a figura mostra, o BGP não envia realmente uma máscara de bits. Em vez disso, ele codifica informações sobre a máscara em um único octeto que precede cada endereço. O octeto de máscara contém um inteiro binário que especifica o número de bits na máscara (os bits de máscara são considerados contíguos). O endereço que segue o octeto de máscara também é compactado – apenas os octetos cobertos pela máscara são incluídos. Portanto, apenas um octeto de endereço segue um valor de

máscara de 8 ou menos, dois seguem um valor de máscara de 9 a 16, três seguem um valor de máscara de 17 a 24 e quatro seguem um valor de máscara de 25 a 32. Curiosamente, o padrão também permite que um octeto de máscara contenha zero (caso em que nenhum octeto de endereço o segue). Uma máscara de tamanho zero é útil porque ela corresponde a uma rota-padrão.

13.13 Atributos de caminho BGP

Dizemos que o BGP não é um protocolo de vetor de distância tradicional. Em vez de simplesmente anunciar uma distância para cada destino, o BGP anuncia informações adicionais, inclusive um caminho. As informações adicionais estão contidas no campo ATRIBUTO DE CAMINHO de uma mensagem de atualização. Um emissor pode usar os atributos de caminho para especificar: um próximo salto para os destinos anunciados, uma lista de sistemas autônomos ao longo do caminho até os destinos e se as informações de caminho foram aprendidas de outro sistema autônomo ou derivadas de dentro do sistema autônomo do emissor.

É importante observar que os atributos de caminho são fatorados para reduzir o tamanho da mensagem UPDATE, significando que os atributos se aplicam a todos os destinos anunciados na mensagem. Assim, se um emissor BGP tenta anunciar caminho para vários conjuntos de destinos cada um com seus próprios atributos, o BGP deve enviar múltiplas mensagens ATUALIZAÇÃO.

Os atributos de caminho são importantes no BGP por três razões. Primeiro, as informações de caminho permitem que um receptor verifique loops de encaminhamento. O emissor especifica um caminho exato de sistemas autônomos a serem usados para alcançar o destino. Se o sistema autônomo do receptor aparece na lista, o anúncio deve ser rejeitado ou haverá um loop de encaminhamento. Segundo, a informação de caminho permite a um receptor implementar restrições de política (por exemplo, rejeitar um caminho que inclua o sistema autônomo de um concorrente). Terceiro, as informações de caminho permitem que um receptor saiba a origem de todas as rotas. Além de possibilitar a um emissor que especifique se as informações vieram de dentro do seu sistema autônomo ou de outro sistema, os atributos de caminho permitem ao emissor declarar se as informações foram coletadas com um protocolo de gateway externo como o BGP ou um protocolo de gateway interno.* Portanto, cada receptor pode decidir se aceita ou rejeita rotas que se originam em sistemas autônomos

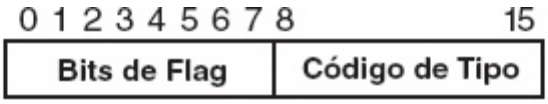
além dos seus peers.

Conceitualmente, o campo ATRIBUTOS DE CAMINHO contém uma lista de itens, cada um deles consistindo em um triplo:

(*tipo, tamanho,valor*).

Em vez de usar campos de tamanho fixo para os três itens, os desenvolvedores escolheram um esquema de codificação flexível que minimiza o espaço ocupado por cada item. O campo *tipo* tem um tamanho fixo de dois octetos, o campo *tamanho* é de um ou dois octetos, e o tamanho do campo *valor* depende do *tamanho*.

A Figura 13.8 ilustra que o campo tipo é dividido em 2 octetos.



(a)

Bits de Flag	Descrição
0	0 para atributo obrigatório; 1 se opcional
1	1 para transitivo; 0 para não transitivo
2	0 para completo; 1 para parcial
3	0 se campo tamanho é um octeto, 1 se
5-7	de dois não usado (deve ser zero)

(b)

Figura 13.8 (a) O campo Código de Tipo de dois octetos que aparece antes de cada item de caminho do atributo BGP e (b) o significado de cada bit de flag.

Cada item no campo ATRIBUTOS DE CAMINHO pode ter um de oito tipos de código possíveis. A Figura 13.9 resume as possibilidades.

Tipo de código	Significado
1	ID da origem das informações do caminho
2	Lista dos sistemas autônomos no caminho até o destino
3	Próximo salto a ser usado para o destino
4	Discriminador usado para múltiplos pontos de saída do sistema autônomo

5	Preferência usada dentro de um sistema autônomo
6	Indicação de que as rotas foram agregadas
7	ID de sistemas autônomos que agregaram rotas
8	ID da comunidade para destinos anunciados

Figura 13.9 Os códigos de tipo de atributo BGP e o significado de cada um.

Para cada item na lista *atributos de caminho*, um campo de TAMANHO segue o campo de TIPO de 2 octetos, e pode ter um ou dois octetos de tamanho. Como mostra a Figura 13.8, o bit de flag 3 especifica o tamanho do campo. Um receptor usa o campo de tipo para determinar o tamanho do campo de tamanho e, depois, usa o conteúdo do campo de tamanho para calcular o tamanho do campo de valor.

13.14 Mensagem BGP KEEPALIVE

Dois peers BGP trocam periodicamente mensagens KEEPALIVE para testar conectividade de rede e verificar se ambos os peers continuam funcionando. Uma mensagem KEEPALIVE consiste no cabeçalho de mensagem-padrão sem quaisquer dados adicionais. Portanto, o tamanho de mensagem total é de *19* octetos (o tamanho mínimo de mensagem BGP).

Existem duas razões por que o BGP usa mensagens KEEPALIVE. Primeiro, a troca periódica de mensagens é necessária porque o BGP usa o TCP para transporte e o TCP não inclui um mecanismo para testar continuamente se uma extremidade de conexão está alcançável. Entretanto, o TCP não informa um erro para uma aplicação se uma tentativa de enviar dados falha. Assim, enquanto ambos os lados enviarem periodicamente um KEEPALIVE, eles serão informados se a conexão TCP falhar. Segundo, mensagens KEEPALIVE conservam largura de banda em comparação com outras mensagens. Muitos protocolos de roteamento antigos usavam troca periódica de informações de roteamento para testar conectividade. Entretanto, como as informações de roteamento raramente mudam, o conteúdo da mensagem dificilmente é alterado. Infelizmente, como as mensagens de roteamento normalmente são grandes, reenviar a mesma mensagem desperdiça largura de banda de rede. Para evitar a ineficiência, o BGP separa a funcionalidade da atualização de rota do teste de conectividade, permitindo que o BGP envie pequenas mensagens KEEPALIVE frequentemente e reservando mensagens UPDATE maiores para situações em que as informações de alcançabilidade mudam.

Lembre-se de que um interlocutor BGP especifica um *tempo de espera* quando abre uma conexão; o tempo de espera define um tempo máximo que o BGP deve esperar sem receber uma mensagem. Como um caso especial, o tempo de espera pode ser zero para especificar que nenhuma mensagem KEEPALIVE é usada. Se o tempo de espera for maior do que zero, o padrão recomenda definir o intervalo de KEEPALIVE em um terço do tempo de espera. Em nenhuma hipótese um BGP peer pode tornar o intervalo de KEEPALIVE menor do que um segundo (o que atende ao requisito de que um timer de espera não zero não pode ser menor do que três segundos).

13.15 Informações da perspectiva do receptor

Um Protocolo de Gateway Externo difere dos protocolos de roteamento tradicionais de uma forma significativa: um peer que usa um protocolo externo não deve simplesmente relatar informação de sua própria FIB. Em vez disso, os protocolos exteriores fornecem informações que são corretas de uma perspectiva externa. Dizemos que um protocolo externo fornece *informação de roteamento de terceiros*. Existem dois problemas: as políticas e as rotas ótimas. O problema da política é óbvio: um roteador dentro de um sistema autônomo pode ter permissão de alcançar um determinado destino, enquanto os externos são proibidos de alcançar o mesmo destino. O problema do roteamento significa que um peer precisa anunciar um próximo salto que seja ótimo da perspectiva externa. A arquitetura na Figura 13.10 pode ser usada para ilustrar a ideia.

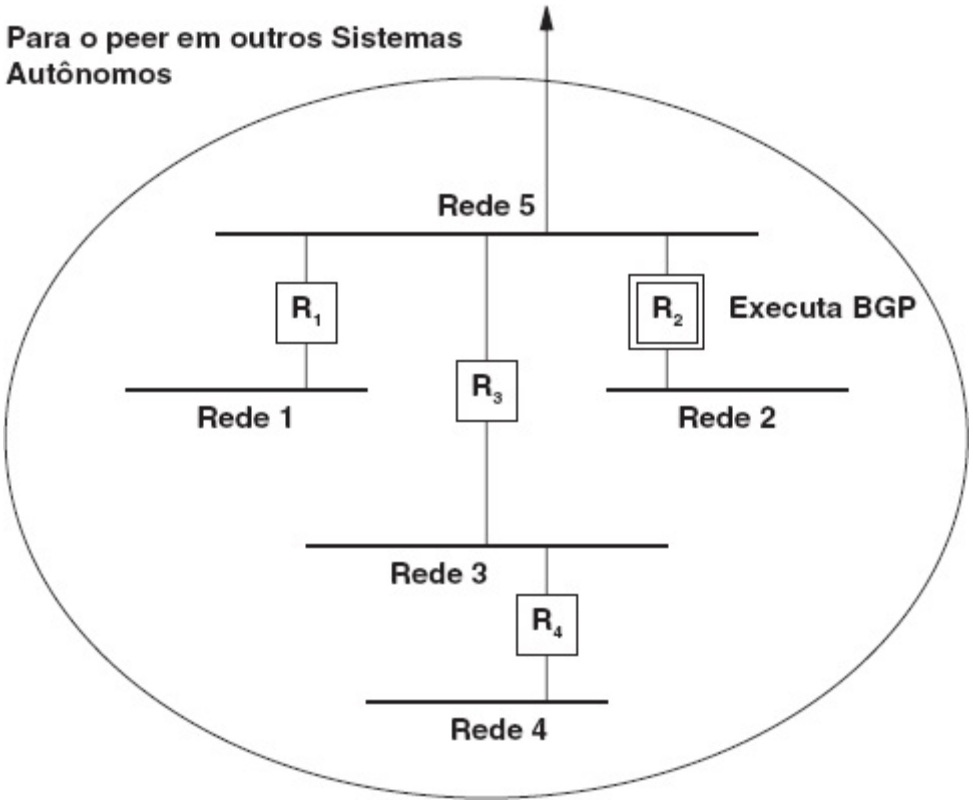


Figura 13.10 Exemplo de um sistema autônomo onde R_2 executa o BGP e relata informações da perspectiva externa, não de sua própria tabela de roteamento.

Na figura, o roteador R_2 foi designado para falar BGP em nome do sistema autônomo. Ele precisa relatar alcançabilidade para as redes de *1* a *4*. Entretanto, ao dar um próximo salto, ele deve relatar a rede *1* como acessível através do roteador R_1, as redes 3 e *4* como acessíveis através do roteador R_3 e a rede *2* como acessível através de R_2 . O ponto chave é que, se R_2 lista a si mesmo com o próximo salto para todos os destinos no sistema autônomo, o roteamento vai ser não ótimo. O peer vai mandar todo o tráfego para o R_2. Em particular, quando um datagrama chega de um peer destinado para as redes *1, 3* ou *4*, o peer vai mandar para R_2 e o datagrama vai ter um salto extra através da rede *5*.

13.16 A restrição básica dos protocolos de gateway

externos

Já vimos que, como os protocolos exteriores seguem restrições de política, as redes que eles anunciam podem ser um subconjunto das redes que podem alcançar. Entretanto, existe uma limitação fundamental imposta sobre o roteamento exterior conforme descrito a seguir.

> *Um protocolo de gateway externo não comunica ou interpreta métrica de distância mesmo se a métrica estiver disponível.*

Embora permitam que um peer declare que um destino se tornou inalcançável ou forneça uma lista dos sistemas autônomos no caminho para o destino, o BGP não pode transmitir ou comparar o custo de duas rotas, a menos que estas venham de dentro do mesmo sistema autônomo. Basicamente, o BGP só pode especificar se existe um caminho até um determinado destino; ele não pode transmitir ou calcular o mais curto de dois caminhos.

Agora podemos ver por que o BGP é cuidadoso em rotular a origem das informações que ele envia. A observação essencial é esta: quando um roteador recebe anúncios para um determinado destino de peers em dois sistemas autônomos diferentes, ele não pode comparar os custos. Portanto, anunciar alcançabilidade com BGP é equivalente a dizer: "meu sistema autônomo fornece um caminho para essa rede.". Não existe um meio de o roteador dizer: "meu sistema autônomo fornece um caminho melhor para essa rede do que outro sistema autônomo.".

Olhar para a interpretação das distâncias nos permite perceber que o BGP não pode ser usado como um algoritmo de roteamento. Suponha que um roteador, R, receba anúncios BGP de dois sistemas autônomos separados. Além disso, suponha que cada um dos dois sistemas autônomos anuncie alcançabilidade para um destino, D. Um deles anuncia um caminho que exige que o datagrama viaje através de três AS, e o outro anuncia um caminho que exige atravessar quatro AS. Qual caminho tem o menor custo? Curiosamente, o roteador R não pode dizer.

Pode parecer que um peer deve usar o tamanho do caminho quando comparando anúncios BGP. Afinal, se um caminho lista sistemas autônomos F, G, H, e I, e outro caminho lista os sistemas autônomos X, Y, e Z, a nossa intuição diz que o último caminho é menor. Mas um

determinado sistema autônomo pode ser grande ou pequeno. Uma vez que um datagrama alcança o sistema autônomo, ele pode precisar atravessar múltiplas redes. Quantas? Quais são as características de atraso e rendimento da rede? Um roteador de borda não pode responder a questão, como podemos ver a seguir.

> *A estrutura interna de um sistema autônomo é oculta, e nenhuma informação sobre o custo dos caminhos internos ao sistema é fornecida pelo BGP.*

A consequência é que um peer não tem como comparar o custo real de dois caminhos se tudo o que o peer recebe é uma lista de sistemas autônomos. Pode acontecer que um caminho com quatro AS envolva redes mais rápidas que um caminho com três AS.

Como o BGP não permite que um sistema autônomo especifique uma métrica com cada rota, o sistema autônomo deve ser cuidadoso para avisar somente a roteadores que o tráfego deve seguir. Podemos fazer o resumo a seguir.

> *Em virtude de um protocolo de gateway externo como o BGP propagar apenas informações de alcançabilidade, um receptor pode implementar restrições de política, mas não pode escolher uma rota de menor custo. Dessa maneira, o BGP só pode ser usado para anunciar caminhos que o tráfego deve seguir.*

O ponto-chave aqui é: um sistema autônomo que usa BGP para fornecer informações de roteamento externo precisa se basear em políticas ou considerar que o trânsito de cada sistema autônomo tem o mesmo custo. Embora possa parecer inócua, a restrição possui algumas consequências surpreendentes.

1. Ainda que o BGP possa anunciar múltiplos caminhos para uma determinada rede, ele não fornece um meio para um sistema autônomo requerer que o tráfego seja encaminhado através de múltiplos caminhos. Ou seja, em qualquer dado instante, todo tráfego roteado de um computador em um sistema autônomo para uma rede

em outro atravessará um caminho, mesmo se várias conexões físicas estiverem presentes. Observe também que um sistema autônomo externo usará apenas um caminho de retorno mesmo que o sistema autônomo divida o tráfego de saída entre dois ou mais caminhos. Como resultado, o atraso e a vazão entre um par de hosts podem ser assimétricos, tornando o tráfego difícil de monitorar ou depurar, e erros dificultam o relato.

2. O BGP não aceita compartilhamento de carga em peers entre sistemas autônomos arbitrários. Se dois sistemas autônomos tiverem vários roteadores conectando-os, será desejável equilibrar o tráfego igualmente entre todos os roteadores. O BGP permite que sistemas autônomos dividam a carga pela rede (por exemplo, para se particionarem em múltiplos subconjuntos e ter múltiplos roteadores anunciando as partições), mas ele não aceita compartilhamento de mais carga geral.
3. Como um caso especial do item 2, o BGP por si só é inadequado para roteamento ótimo em uma arquitetura que tem duas ou mais redes remotas interconectadas em vários pontos. Em vez disso, os gerentes precisam configurar manualmente que redes são anunciadas por cada roteador externo.
4. Para ter um roteamento racionalizado, todos os sistemas autônomos na Internet precisam concordar com um esquema consistente para anunciar alcançabilidade. Ou seja, o BGP sozinho não garantirá consistência global.

13.17 A arquitetura de roteamento e registros da Internet

Para que uma Internet opere sem falhas, as informações de roteamento precisam ser globalmente consistentes. Protocolos individuais como o BGP, que tratam da troca entre um par de roteadores, não garantem consistência global. Portanto, é necessário um esforço adicional para racionalizar as informações de roteamento globalmente. Na arquitetura de roteamento original da Internet, o sistema core garantia informações de roteamento globalmente consistentes porque, em qualquer momento, o core tinha exatamente um caminho para cada destino. Entretanto, o sistema core e seu sucessor (chamado de sistema árbitro de roteamento) foram removidos. Ironicamente, nem um único mecanismo foi criado como um substituto para cuidar da tarefa da racionalização de roteamento – a Internet atual não possui um mecanismo central para validar rotas e garantir consistência global.

Para entender a arquitetura de roteamento atual, precisamos examinar a topologia física. Um par de ISPs pode interconectar-se privadamente (por exemplo, concordando em alugar um circuito entre dois roteadores) ou pode interconectar a *pontos de troca da Internet* (*Internet Exchange Points – XPs*), que também são conhecidos como *pontos de acesso à rede* (*Network Access Points – NAPs*). Dizemos que os ISPs se engajam em *peering privado* (*private peering*) ou que eles entram em um *acordo de peering* (*peering agreement*). Em termos de roteamento, um peering privado representa o limite entre os dois sistemas autônomos. Os dois ISPs definem sua relação, que pode ser vista como *upstream* (um grande ISP concorda em receber tráfego de um ISP menor), *downstream* (um grande ISP transmite tráfego para um ISP menor) ou *trânsito* (um ISP concorda em aceitar e encaminhar tráfego para outros ISPs).

Para ajudar a garantir que as rotas são válidas, os ISPs usam serviços conhecidos como *registros de roteamento*. Basicamente, um registro de roteamento mantém informação sobre que ISPs possuem que blocos de endereços. Portanto, se o ISP *A* envia um anúncio para o ISP *B* solicitando ter alcançabilidade à rede *N*, o ISP *B* pode usar informações de um registro de roteamento para verificar se o endereço *N* foi atribuído ao ISP *A*. Infelizmente, existem muitos registros de roteamento e não existe qualquer mecanismo para validar os dados em um registro. Portanto, podem ocorrer problemas temporários de roteamento, como *buracos negros*, em que um determinado endereço não é alcançável de todas as partes da Internet. É claro, os ISPs e a maioria dos registros de roteamento tentam encontrar e reparar esses problemas rapidamente, mas, sem um registro centralizado e autorizado, o roteamento da Internet não é infalível.

13.18 Mensagem de notificação BGP

Além dos tipos de mensagem OPEN e UPDATE descritas anteriormente, o BGP aceita um tipo de mensagem NOTIFICATION usado para controle ou quando um erro ocorre. Os erros são permanentes – uma vez que detecta um problema, o BGP envia uma mensagem de notificação e, depois, fecha a conexão TCP. A Figura 13.11 ilustra o formato da mensagem.

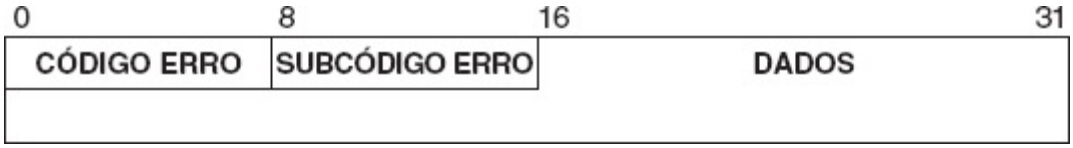


Figura 13.11 Formato da mensagem BGP NOTIFICATION. Esses octetos seguem o cabeçalho de mensagem-padrão.

O campo de 8 bits rotulado como CÓDIGO ERRO especifica uma das possíveis razões listadas na Figura 13.12.

CÓDIGO DE ERRO	Significado
1	Erro no cabeçalho da mensagem
2	Erro na mensagem OPEN
3	Erro na mensagem UPDATE
4	Timer de ESPERA expirado
5	Erro de máquina de estado finito
6	Fim (termina conexão)

Figura 13.12 Os possíveis valores do campo CÓDIGO ERRO em uma mensagem BGP NOTIFICATION.

Para cada CÓDIGO ERRO possível, o campo SUBCÓDIGO ERRO contém uma explicação adicional. A Figura 13.13 lista os valores possíveis.

13.19 Extensões multiprotocolo BGP para IPv6

O BGP foi originalmente projetado para encaminhar informação de roteamento IPv4. Por volta do ano 2000, tornou-se aparente que sistemas autônomos precisavam trocar tipos adicionais de informação de roteamento. Naquela época, as duas necessidades mais urgentes eram IPv6 e MPLS, que são descritos no Capítulo 16. Em vez de criar uma versão de BGP para IPv6 e outra para MPLS, um grupo no IETF criou *extensões multiprotocolo*. A ideia é que, quando anunciando destinos, um emissor possa especificar que os endereços de destino eram de uma *família de endereço* em particular. Para enviar informação IPv6, um emissor especifica a família de endereço IPv6 e, para enviar informação MPLS, um emissor especifica a família de endereço MPLS.

Apenas três itens levados no BGP usam mensagens com endereço IPv4: o

endereço de um *destino* que é anunciado, o endereço de um *próximo salto* usado para alcançar o destino e o endereço de um *agregador* que tem prefixos agregados. As extensões são designadas para permitir que qualquer um dos três itens use um endereço de família arbitrário em vez do IPv4.

Subcódigos para erros de cabeçalho de mensagem

1 Conexão não sincronizada
2 Tamanho de mensagem incorreto
3 Tipo de mensagem incorreto

Subcódigos para erros de mensagem OPEN

1 Número de versão não
2 Sistema autônomo de peer inválido
3 Identificador BGP inválido
4 Parâmetro opcional não
5 Obsoleto (não mais usado)
6 Tempo de espera inaceitável

Subcódigos para erros de mensagem UPDATE

1 Lista de atributos mal-formada
2 Atributo não reconhecido
3 Atributo ausente
4 Erro de flags de atributo
5 Erro de tamanho de atributo
6 Atributo ORIGIN inválido
7 Obsoleto (não mais usado)
8 Próximo salto inválido
9 Erro no atributo opcional
10 Campo de rede inválido
11 Caminho de sistema autônomo mal-formado

Figura 13.13 O significado do campo SUBCÓDIGO ERRO em uma mensagem BGP NOTIFICATION.

Os desenvolvedores escolheram duas propriedades-chave para as extensões multiprotocolo descritas a seguir.

- *Opcional.* Extensões multiprotocolo não são exigidas.

- *Não transitiva.* Um roteador não pode passar as extensões para outros AS.

As decisões são importantes por duas razões. Fazer as extensões opcionais garante retrocompatibilidade (ou seja, o software BGP antigo continuará a funcionar). Se uma implementação do BGP não entender as extensões, ela irá simplesmente ignorá-las, bem como aos roteadores que elas anunciam. A proibição de encaminhamento de extensões mantém o roteamento Internet livre de ser vulnerável a ataque. Se o encaminhamento cego fosse permitido, um Sistema Autônomo que não entendesse as extensões poderia inadvertidamente encaminhar informação incorreta, na qual o próximo AS iria confiar.

As extensões multiprotocolo são levadas nos *atributos de caminho* do BGP. Dois novos tipos de atributo foram criados para permitir a um emissor especificar um lista de destinos não IPv4 que são alcançáveis e uma lista de destinos não IPv4 que são inalcançáveis. Em vez de usar o termo destinos alcançáveis, a extensão usa o termo *informação de alcançabilidade de camada de rede* (*Network Layer Reachability Information – NLRI*). Consequentemente, os dois tipos de atributos são:

- multiprotocolo alcançável NLRI (Tipo 14);
- multiprotocolo não alcançável NLRI (Tipo 15).

13.20 Atributo multiprotocolo NLRI alcançável

Um roteador usa o atributo *multiprotocolo NLRI alcançável* para anunciar destinos alcançáveis, mesmo dentro de seu sistema autônomo, ou destinos alcançáveis através do sistema autônomo. Cada destino no atributo é chamado um *Protocolo de Endereço de Sub-rede* (*Subnetwork Protocol Address – SNPA*). A Figura 13.14 lista os campos em um atributo:

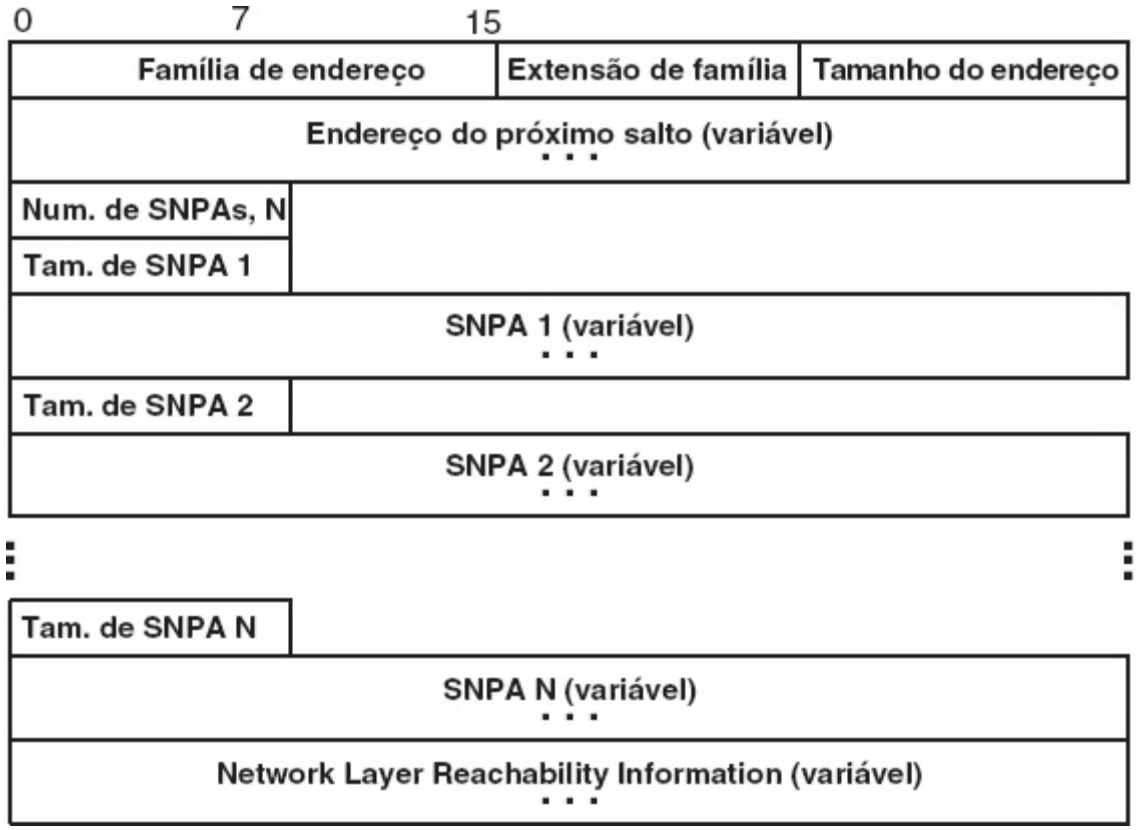


Figura 13.14 O formato de um atributo BGP *multiprotocolo alcançável* NLRI usado para o IPv6 e outro endereço de destino não IPv4.

Como mostra a figura, o atributo começa com campos que fornecem família e tamanho de endereço. O atributo então especifica um endereço de próximo salto e um conjunto de destinos (SNPAs) encontrável através do próximo salto. Cada destino é precedido por um tamanho de 1 octeto.

13.21 Roteamento Internet e economia

Embora as pesquisas em roteamento estejam focadas em encontrar mecanismos que computem caminhos mais curtos, eles não são a primeira preocupação do Tier-1; a economia é. Antes de eles interconectarem suas redes e começarem a passar o tráfego, um par de ISP negocia um contrato comercial. Um contrato típico especifica uma das três possibilidades a seguir.

- ISP 1 é um cliente do ISP 2.
- ISP 2 é um cliente do ISP 1.
- Os dois ISPs são *peers*.

O relacionamento com o *cliente* é definido pelo fluxo de dados: um ISP que recebe mais dados do que ele envia é definido como um cliente e deve pagar uma taxa. A definição é fácil de entender se considerarmos um pequeno ISP. Quando um usuário residencial se torna um cliente de um ISP local (por exemplo, um operador de cabo), ele deve pagar uma taxa porque irá baixar muito mais dados do que eles enviam (por exemplo, cada vez que um usuário navega na web, os dados devem ser enviados a partir do provedor para o usuário). O valor que o cliente paga depende da quantidade de dados de que ele deseja fazer download. No próximo nível da hierarquia de ISP, um provedor local torna-se um cliente de um ISP maior – porque ele faz o download de mais dados do que ele gera, o ISP local deve pagar ao ISP maior.

Que tal dois Tier-1 ISPs no topo da hierarquia? Se eles são verdadeiramente *peers*, os dois ISPs terão cada um o mesmo número de clientes. Assim, em média, esperam que a mesma quantidade de dados viaje em cada sentido entre eles. Então, eles escrevem um contrato em que se comprometem a parear, o que significa que eles vão dividir o custo de uma conexão entre eles. No entanto, eles também concordam em monitorar os dados que passam através da conexão. Se, durante um determinado mês, mais dados passam do ISP 1 para o ISP 2, o contrato estipula que ISP 2 pagará ao ISP 1 uma quantidade que depende da diferença na quantidade de dados.

Uma vez que os contratos foram estabelecidos, os ISPs tentam arranjar roteamento para gerar mais receita. Normalmente, os clientes pagam mais pelos dados. Portanto, se um cliente anuncia acessibilidade para um determinado destino, um ISP vai preferir enviar dados através do cliente, em vez de um peer. Além disso, se um ISP quer evitar a obtenção de dados a partir de um peer, pode organizar mensagens BGP que causam a parada de envio do peer (por exemplo, se um ISP coloca o número AS do peer no caminho em um anúncio, o peer vai rejeitar o caminho como se estivesse havendo loop de roteamento). O ponto principal é descrito a seguir.

> *No centro da Internet, o roteamento é baseado em grande parte na economia em vez de nos caminhos mais curtos. ISPs maiores organizam políticas, preferências e anúncios BGP para forçar os datagramas a irem por rotas que geram o maior retorno, independentemente se a rota é a menor.*

13.22 Resumo

Em uma grande internet, os roteadores precisam ser particionados em grupos ou o volume do tráfego de roteamento será intolerável. A Internet global é composta por um conjunto de sistemas autônomos, sendo que cada um consiste em rotas e redes sob uma autoridade administrativa. Um sistema autônomo usa um protocolo de gateway externo para anunciar rotas a outros sistemas autônomos. Especificamente, um sistema autônomo precisa anunciar a alcançabilidade de suas redes para outro sistema autônomo antes que suas redes sejam atingíveis de origens dentro do outro sistema.

O Border Gateway Protocol (BGP) é o Exterior Gateway Protocol mais usado. O BGP contém cinco tipos de mensagem que são usados para iniciar comunicação (OPEN), enviar informações de alcançabilidade (UPDATE), reportar uma condição de erro (NOTIFICATION), revalidar informações (REFRESH) e garantir que os peers permaneçam em comunicação (KEEPALIVE). Cada mensagem se inicia com um cabeçalho-padrão. O BGP usa TCP para comunicação.

Embora originalmente criado para IPv4, o BGP foi estendido para funcionar com outros protocolos. Em particular, um conjunto de extensões multiprotocolo permite ao BGP passar informação sobre MPLS assim como IPv6.

Na Internet global, cada grande ISP é um sistema autônomo separado, e o limite entre os sistemas autônomos consiste em um acordo de peering entre dois ISPs. Fisicamente, pode ocorrer peering em um ponto de troca da Internet ou sobre um circuito privativo alugado. Um ISP usa BGP para se comunicar com seu peer, tanto para anunciar redes (ou seja, prefixos de endereço) que podem ser atingidas através dele quanto para aprender sobre redes que possam ser atingidas pelo encaminhamento do peer. Embora existam serviços conhecidos como registros de roteamento, que auxiliam os ISPs a validar anúncios, podem ocorrer problemas porque a Internet atualmente não tem um registro centralizado confiável.

No centro da Internet, o roteamento é baseado em custos em vez de caminhos mais curtos. Grandes ISP escolhem rotas que vão maximizar seus rendimentos e minimizar seus custos.

EXERCÍCIOS

13.1 Se seu site executa um Exterior Gateway Protocol como o BGP, quantas rotas você anuncia? Quantas rotas você importa de um ISP?

13.3 Algumas implementações do BGP usam um mecanismo de *hold down* que faz com que o protocolo atrase a aceitação de um OPEN de um peer por um tempo fixo seguinte ao recebimento de uma mensagem *cease request* desse vizinho. Descubra que problema um hold down ajuda a resolver.

13.3 A especificação formal do BGP inclui uma máquina de estado finito que explica como o BGP opera. Desenhe um diagrama da máquina de estado e rotule as transições.

13.4 O que acontece se um roteador em um sistema autônomo enviar mensagens de atualização de roteamento BGP para um roteador em outro sistema autônomo, alegando ter alcançabilidade para todo destino da Internet possível?

13.5 Dois sistemas autônomos podem estabelecer um loop de encaminhamento enviando mensagens de atualização BGP um para o outro? Sim ou não? Por quê?

13.6 Um roteador que usa BGP para anunciar rotas deve tratar o conjunto das rotas anunciadas de modo diferente do conjunto das rotas na tabela de roteamento local? Por exemplo, um roteador deve anunciar alcançabilidade se ele não tiver instalado uma rota para essa rede em sua tabela de roteamento? Por quê? (Dica: Leia a RFC 1771.)

13.7 Com relação à questão anterior, examine cuidadosamente a especificação do BGP-4. É permitido anunciar alcançabilidade para um destino que não esteja listado na tabela de roteamento local?

13.8 Se você trabalha para uma grande empresa, descubra se ela inclui mais de um sistema autônomo. Em caso afirmativo, como eles trocam informações de roteamento?

13.9 Qual é a principal vantagem de dividir uma grande empresa multinacional em vários sistemas autônomos? Qual é a maior desvantagem?

13.10 As empresas A e B usam o BGP para trocar informações de roteamento. Para prevenir que os computadores em B alcancem máquinas em uma de suas redes, N, o administrador de rede na empresa A configura o BGP para omitir N dos anúncios enviados a B. A rede N é segura? Sim ou não? Por quê?

13.11 Como o BGP usa um protocolo de transporte seguro, as mensagens

KEEPALIVE não podem se perder. Faz sentido especificar um intervalo de keepalive como um terço do valor do tempo de espera? Sim ou não? Por quê?

13.12 Consulte as RFCs para saber detalhes do campo *Atributos de Caminho*. Qual é o tamanho mínimo de uma mensagem BGP UPDATE?

---

* A terminologia foi criada em um tempo em que um roteador era chamado *gateway*, e ainda persiste.

** Embora seja possível executar BGP em um sistema diferente de um roteador, a maioria dos sistemas autônomos escolhe executar BGP em um roteador que tem uma conexão direta com outro sistema autônomo.

* O próximo capítulo descreve os protocolos de gateway internos.

CAPÍTULO 14

Roteamento em um sistema autônomo (RIP, RIPng, OSPF, IS-IS)

CONTEÚDOS DO CAPÍTULO

14.1 Introdução

O capítulo anterior apresentou o conceito de sistema autônomo e examinou o BGP, um Protocolo de Gateway Externo que um roteador utiliza para anunciar redes dentro de seu sistema para outros sistemas autônomos. Este capítulo completa nossa sinopse do roteamento de internet examinando como um roteador em um sistema autônomo aprende sobre outras redes dentro de seu sistema autônomo.

14.2 Rotas internas estáticas *versus* dinâmicas

Dizemos que dois roteadores dentro de um sistema autônomo são internos um ao outro. Por exemplo, dois roteadores em um *campus* universitário são considerados internos um ao outro desde que as máquinas no *campus* sejam coletadas em um único sistema autônomo.

Como os roteadores em um sistema autônomo podem aprender sobre redes dentro do sistema autônomo? Em internets pequenas, os gerentes podem estabelecer e modificar rotas manualmente. O administrador mantém uma tabela de redes e a atualiza sempre que uma nova rede é acrescentada no sistema autônomo ou excluída dele. Por exemplo, considere a pequena internet corporativa mostrada na Figura 14.1.

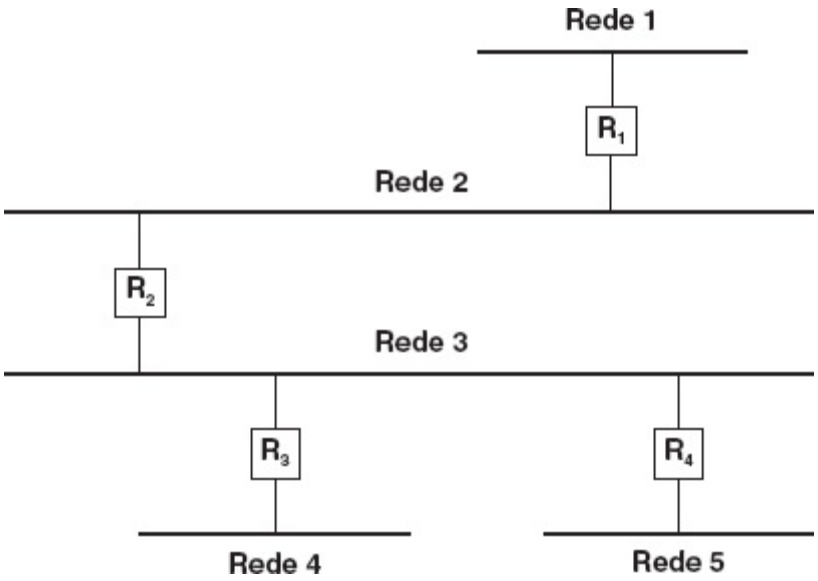


Figura 14.1 Um exemplo de uma pequena internet consistindo de 5 Ethernets e 4 roteadores. Existe apenas uma rota possível entre quaisquer dois hosts no exemplo.

O roteamento para a internet na figura é simples porque existe apenas um caminho entre quaisquer dois pontos. Se uma rota ou roteador falhar, a intranet será desconectada pois não há caminhos redundantes. Entretanto, um gerente pode configurar manualmente as rotas em todos os hosts e roteadores, e nunca precisa mudá-las. Obviamente, se a intranet mudar (por exemplo, se uma rede for acrescentada), o gerente precisa reconfigurar as rotas de acordo com essa mudança.

As desvantagens de um sistema manual são óbvias: os sistemas manuais não podem acomodar crescimento rápido e se baseiam nos humanos para mudar de rota sempre que ocorre uma falha de rede. Na maioria das intranets, os humanos simplesmente não conseguem responder a mudanças rápido o bastante para resolver problemas; são necessários métodos automatizados. Para entender como o roteamento automático pode aumentar a confiabilidade, considere o que ocorre se acrescentarmos um roteador adicional à internet da Figura 14.1, produzindo a internet mostrada na Figura 14.2.

Na figura, existem vários caminhos entre alguns hosts. Nesses casos, um gerente normalmente escolhe um caminho para ser o *caminho primário* (isto é, o caminho que será usado para todo o tráfego). Se um roteador ou rede ao longo do caminho primário falhar, as rotas precisam ser trocadas para enviar tráfego ao longo de um caminho alternativo. A mudança de rota automatizada ajuda de duas maneiras. Primeiro, como os computadores

podem responder a falhas muito mais rapidamente do que os humanos, as mudanças de rota automatizada são menos demoradas. Segundo, como os humanos podem cometer pequenos erros quando inserem endereços de rede, o roteamento automatizado é menos propenso a erros. Portanto, mesmo em pequenas internets, um sistema automatizado é usado para mudar de rota de maneira rápida e confiável.

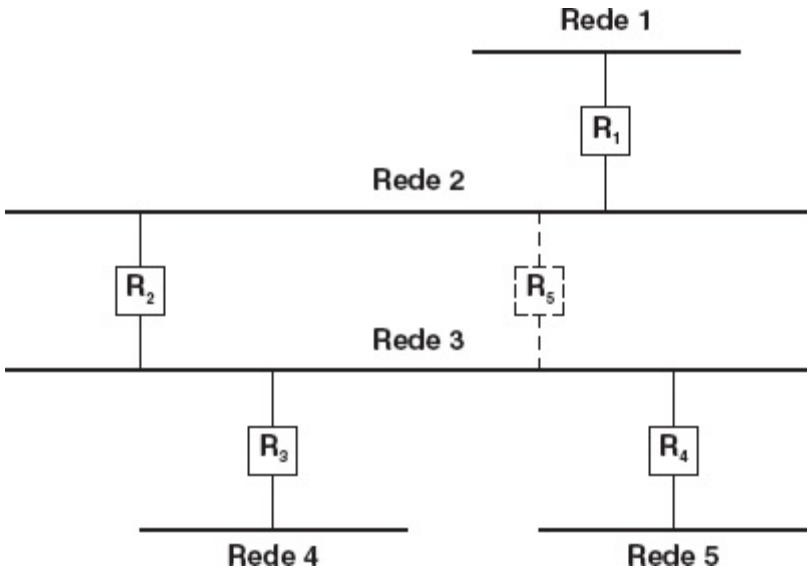


Figura 14.2 A adição de um roteador R_5 introduz um caminho alternativo entre as redes *2* e *3*. Software de roteamento pode se adaptar rapidamente a uma falha e, automaticamente, trocar rotas para o caminho alternativo.

Para automatizar a tarefa de manter as informações de roteamento precisas, os roteadores interiores se comunicam periodicamente a fim de trocar informações de roteamento. Diferentemente da comunicação de roteador exterior, para a qual o BGP fornece um padrão amplamente aceito, nenhum protocolo surgiu para uso dentro de um sistema autônomo ou site. Parte do motivo para a diversidade vem da diversidade em sistemas autônomos. Alguns sistemas autônomos correspondem a grandes empresas (por exemplo, uma corporação) em um único site, enquanto outros correspondem a uma organização com muitos sites conectados por rede de longa distância. Mesmo se considerarmos sites individuais de Internet, as topologias de rede (por exemplo, grau de redundância), tamanhos e tecnologias de rede variam grandemente. Outra razão para a diversidade de protocolos de roteamento interno se origina de compromisso entre simplicidade e funcionalidade, e o tráfego que protocolos impõem nas redes subjacentes – os protocolos que são fáceis de instalar e configurar

podem não fornecer a funcionalidade necessária ou podem impor cargas intoleráveis para as redes. Como resultado, diversos protocolos se tornaram populares, mas nenhum é sempre ótimo.

Embora protocolos internos múltiplos sejam usados, um dado sistema autônomo sempre escolhe limitar o número de protocolos que são implantados. Um pequeno AS tende a escolher um único protocolo e usá-lo exclusivamente para propagar informação de roteamento internamente. Mesmo grandes sistemas autônomos tendem a escolher um conjunto pequeno. Existem duas razões. Primeiro, um dos aspectos mais complexos de roteamento surge da interação de protocolos. Se o protocolo A for usado em alguns roteadores e o protocolo B em outros, no final, um roteador entre os dois conjuntos deverá se comunicar usando ambos os protocolos e ter um meio de transferir informação entre eles. Tais interações são complexas, e muito cuidado deve ser tomado, ou diferenças em protocolos podem levar a consequências inesperadas. Segundo, devido ao fato de os protocolos de roteamento serem difíceis de entender e configurar, cada sistema autônomo deve ter uma equipe que é treinada para instalar, configurar e dar suporte para cada protocolo individual, assim como para software que lidam com interações entre si. O treinamento pode ser dispendioso, portanto, limitar o número de protocolos pode reduzir custos.

Usamos o termo *Protocolo de Gateway Interno* (*Interior Gateway Protocol – IGP*) como uma descrição genérica que se refere a qualquer protocolo que roteadores internos usam quando trocam informação de roteamento. A Figura 14.3 ilustra a ideia geral: dois sistemas autônomos cada um usando uma IGP específica para propagar informação de roteamento entre roteadores internos. O sistema, então, usa O BGP para resumir informação e comunicá-la a outros sistemas autônomos.

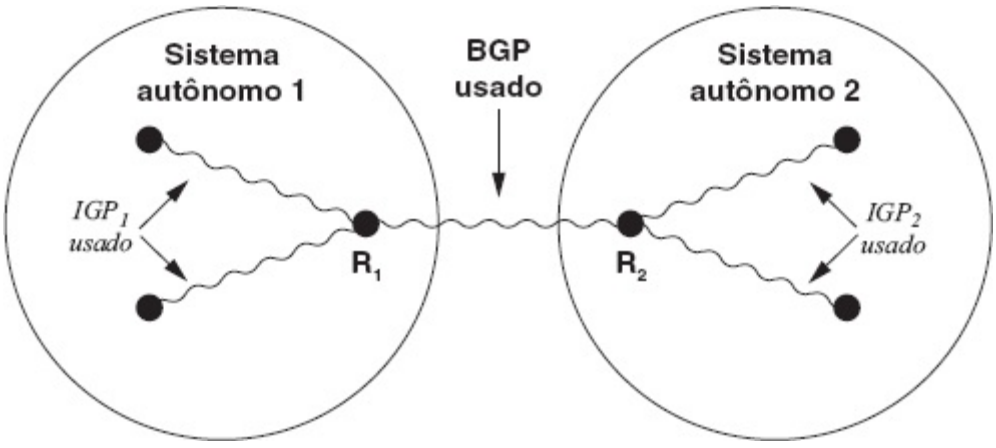


Figura 14.3 Visão conceitual de dois sistemas autônomos, cada um usando

seu próprio IGP internamente, e então usando BGP para comunicar rotas a outro sistema.

Na figura, IGP_1 se refere ao protocolo de roteamento interno usado dentro do sistema autônomo *1* e IGP_2 se refere ao protocolo usado dentro do sistema autônomo *2*. O roteador R_1 usará IGP_1 para obter rotas internamente, resumir a informação, aplicar políticas e, então, usar o BGP para exportar a informação resultante. De forma similar, o roteador R_2 usará IGP_2 para obter informação que ele exporta. Podemos resumir o conceito chave desta maneira a seguir.

Se protocolos múltiplos de roteamento são usados, um único roteador pode executar dois ou mais protocolos de roteamento simultaneamente.

Em especial, os roteadores que executam BGP para anunciar acessibilidade normalmente também precisam executar um IGP a fim de obter informações de dentro de seu sistema autônomo. As próximas seções descrevem protocolos de gateway internos específicos; seções posteriores consideram algumas consequências do uso de protocolos múltiplos.

14.3 Routing Information Protocol (RIP)

14.3.1 História do RIP

O *Routing Information Protocol* (*RIP*) tem se mantido em uso generalizado desde o início da Internet. Originalmente era conhecido pelo nome do programa que o implementa, o *routed*.* O software *routed* foi projetado na Universidade da Califórnia, em Berkeley, para fornecer informações de roteamento consistentes entre máquinas em suas redes locais. O protocolo se baseou em estudos antigos feitos na Xerox Corporation's Palo Alto Researdh Center (PARC). A versão Berkeley do RIP generalizou a versão PARC para abranger múltiplas famílias de redes. O RIP se baseia na transmissão física de rede para fazer trocas de roteamentos rapidamente, e não foi originalmente desenhado para ser usado em grandes

redes de longa distância. Fornecedores desenvolveram posteriormente versões de RIP adaptadas para uso em WANs.

Apesar das pequenas melhorias em relação a seus predecessores, a popularidade do RIP como um IGP não surgiu unicamente de seus méritos técnicos, e sim do fato de a Berkeley distribuir o software *routed* junto com seus populares sistemas UNIX 4BSD. Muitos sites TCP/IP antigos adotaram e instalaram o RIP sem sequer considerar seus méritos ou limitações técnicas. Uma vez instalado e rodando, ele se tornou a base para o roteamento local, e os fornecedores, mais tarde, começaram a oferecer produtos compatíveis com o RIP.

14.3.2 Funcionamento do RIP

O protocolo RIP subjacente é uma implementação simples do roteamento de vetor de distância para redes locais. Ele suporta dois tipos de participantes: *ativos* e *passivos*. Os participantes ativos anunciam suas rotas a outros; os participantes passivos escutam as mensagens RIP e as utilizam para atualizar sua tabela de encaminhamento, mas não anunciam. Apenas um roteador pode executar o RIP no modo ativo, um host usa o modo passivo.

Um roteador executando o RIP no modo ativo difunde uma mensagem de atualização de roteamento a cada 30 segundos. A atualização contém informações tiradas do FIB atual do roteador. Cada atualização contém um conjunto de pares, no qual cada par especifica um endereço de rede IP e uma distância inteira até essa rede. O RIP usa uma *métrica de contagem de saltos* para medir distâncias. Na métrica RIP, um roteador é definido para estar a um salto de uma rede diretamente conectada,** dois saltos de uma rede acessível através de outro roteador e assim por diante. Portanto, o *número de saltos* ou a *contagem de saltos* ao longo de um caminho desde uma determinada origem até um determinado destino se refere ao número de redes que um datagrama encontra ao longo desse caminho. Deveria ser óbvio que usar contagens de saltos para calcular caminhos mais curtos nem sempre produz resultados ótimos. Por exemplo, um caminho com contagem de saltos *3* que cruza três Ethernets pode ser substancialmente mais rápido do que um caminho com contagem de saltos *2* que cruza duas conexões de satélite. Para compensar diferenças nas tecnologias, muitas implementações RIP permitem aos gerentes configurarem contagens de saltos artificialmente altas ao anunciar conexões com redes lentas.

Tanto os participantes RIP ativos quanto os passivos escutam todas as mensagens de difusão (broadcast) e atualizam suas tabelas de encaminhamento de acordo com o algoritmo de vetor de distância descrito anteriormente no Capítulo 12. Por exemplo, se os roteadores na intranet da Figura 14.2 na página 198 usam RIP, o roteador R_1 difundirá por broadcast uma mensagem na rede 2 que contém o par $(1,1)$, significando que ela pode alcançar a rede 1 à distância (isso é, custo) 1. Os roteadores R_2 e R_5 receberão o broadcast e instalarão uma rota para a rede 1 através de R_1 (a custo 2). Posteriormente, os roteadores R_2 e R_5 incluirão o par $(1, 2)$ quando difundirem suas mensagens RIP na rede 3. Futuramente, todos os roteadores e hosts instalarão uma rota para a rede 1.

O RIP especifica algumas regras para melhorar o desempenho e a confiabilidade. Por exemplo, uma vez que o roteador aprende uma rota de outro roteador, ele precisa aplicar *hysteresis*, significando que não substitui a rota por uma rota de custo igual. Em nosso exemplo, se as rotas R_2 e R_5 anunciarem a rede 1 a custo 2, os roteadores R_3 e R_4 instalarão uma rota através da que anuncia primeiro. Podemos fazer o resumo a seguir.

> *Para evitar oscilação entre caminhos de custo igual, o RIP especifica que rotas existentes devem ser mantidas até que uma nova rota tenha custo estritamente inferior.*

O que acontece se um roteador falhar (por exemplo, se ele travar)? O RIP especifica que, quando um roteador instala uma rota em sua tabela de encaminhamento, ele deve iniciar um timer para a entrada. O timer é reiniciado sempre que o roteador recebe outra mensagem RIP anunciando a mesma rota. A rota se torna inválida após 180 segundos sem ser anunciada novamente.

O RIP precisa lidar com três tipos de erros causados pelo algoritmo subjacente. Primeiro, como o algoritmo não detecta explicitamente loops de encaminhamento, o RIP precisa considerar que os participantes podem ser confiáveis ou tomar precauções para prevenir esses loops. Segundo, para

evitar instabilidades, o RIP precisa usar um baixo valor para a distância máxima possível (o RIP usa *16*). Assim, para intranets em que as contagens de saltos legítimas alcançam *16*, os gerentes precisam dividir a internet em seções ou usar um protocolo alternativo.* Terceiro, o algoritmo de vetor de distância usado pelo RIP pode criar um problema conhecido como *convergência lenta ou contagem ao infinito*, em que surgem inconsistências porque as mensagens de atualização de roteamento se propagam lentamente através da rede. Escolher um infinito pequeno (*16*) ajuda a limitar a convergência lenta, mas não a elimina.

14.4 O problema da convergência lenta

As inconsistências de tabela de roteamento e o problema da convergência lenta não são exclusivos do RIP. Eles são problemas fundamentais que podem ocorrer com qualquer protocolo de vetor de distância em que mensagens de atualização transmitem apenas pares de rede de destino e distância para essa rede. Para entender o problema, considere o uso de um protocolo de vetor de distância nos roteadores da Figura 14.2 (página 198). Para simplificar o exemplo, vamos considerar somente três roteadores R_1, R_2, e R_3, e somente as rotas que eles têm para a rede *1*. Para alcançar a rede *1*, R_3, encaminha para R_2, e R_2 encaminha para R_1. A parte (a) da Figura 14.4 ilustra o encaminhamento.

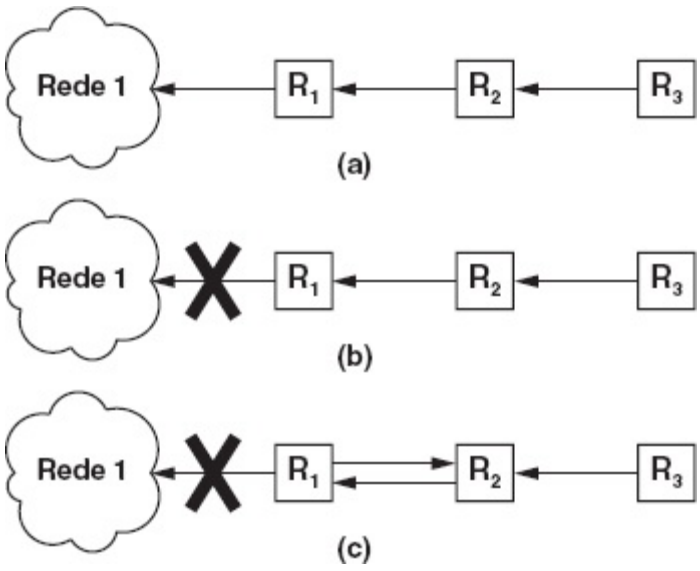


Figura 14.4 Ilustração do problema de convergência com (a) três roteadores

que têm uma rota para a rede *1*, (b) a conexão para a rede *1* falhou e R_1 perdeu seu roteador, e (c) um loop de roteamento causado devido a R_2 anunciar uma rota para a rede *1*.

Na parte (a), assumimos que todos os roteadores estão executando um protocolo de vetor de distância. Assumiremos RIP, mas a ideia se aplica a qualquer protocolo de vetor de distância. O roteador R_1 tem uma conexão direta à rede *1*. Então, quando ele transmite por broadcast o conjunto de destinos de sua FIB, R_1 inclui uma entrada para a rede *1* na distância *1*. O roteador R_2 aprendeu a rota de R_1, instalou a rota em sua tabela de encaminhamento e anunciou a rota a uma distância *2*. Finalmente, R_3 aprendeu a rota de R_2 e anunciou a rota a uma distância *3*.

Na parte (b) da figura, assumimos que uma falha ocorreu e desconectou R_1 da rede *1*. Talvez a conexão entre o roteador R_1 e a rede *1* esteja desligada ou a rede *1* tenha perdido energia. A interface de rede em R_1 vai detectar a perda de conectividade e o IP vai remover a rota para a rede *1* de sua tabela de encaminhamento (ou deixar a entrada, mas colocar a distância ao infinito para que o roteador não seja usado).

Lembre-se de que R_2 transmite periodicamente por broadcast suas informações. Suponha que, imediatamente após, R_1 detecte a falha e remova a rota de sua tabela, R_2 transmitiu sua informação de roteamento. Entre outros itens no broadcast, R_2 vai anunciar uma rota para a rede *1* a uma distância *2*. Entretanto, a menos que o protocolo inclua mecanismos extras para preveni-lo, as regras para roteamento de vetor de distância significam que R_1 vai examinar o broadcast de R_2, encontrar uma rota para a rede *1* e adicionar uma nova rota a sua tabela com distância *3* (a distância R_2 anunciada mais 1) e R_2 como próximo salto.

Infelizmente, a parte (c) da figura mostra o que aconteceu: R_1 instalou uma rota para a rede *1* que passa por R_2, e R_2 tem uma rota que passa por

R_1. Nesse ponto, se tanto R_1 ou R_2 receberem um datagrama destinado para a rede *1*, eles vão roteá-lo para frente e para trás, até que o limite de salto seja alcançado. Podemos dizer a seguir outras palavras.

> *Um algoritmo de vetor de distância convencional pode causar um loop de roteamento após uma falha porque uma informação de roteamento que um roteador enviou pode alcançá-lo novamente.*

O problema persiste, pois os dois roteadores permanecem continuamente confusos sobre o roteamento. Na próxima etapa de trocas de roteamento, R_1 vai transmitir um anúncio que inclui o custo atual para alcançar a rede 1. Quando ele recebe o anúncio de R_1, R_2 vai aprender que a nova distância é *3*. R_2 atualiza sua distância para alcançar a rede *1*, fazendo a distância 4. Na terceira etapa, R_1 recebe uma atualização de roteamento de R_2 que inclui a distância aumentada. R_1 vai aumentar sua distância para *5* e anunciar a maior distância na próxima atualização. Os dois roteadores continuam enviando mensagens de atualização de roteamento para trás e para frente, e a distância aumenta de *1* a cada troca. A atualização continua até que a contagem chegue ao infinito (*16* para o RIP).

14.5 Resolvendo o problema da convergência lenta

Uma técnica conhecida como *atualização de horizonte dividido* (*split horizon update*) foi desenvolvida para permitir que protocolos de vetor de distância, tais como o RIP, solucionem o problema de convergência lenta. Quando usa o horizonte dividido, um roteador não propaga informações sobre a rota pela mesma interface pela qual a rota chegou. No nosso exemplo, o horizonte dividido evita que o roteador R_2 anuncie uma rota para a rede *1* de volta para o R_1; portanto, se R_1 perder conectividade com a rede *1*, ele irá parar de anunciar uma rota. Com o horizonte dividido, nenhum loop de encaminhamento aparece na rede do exemplo. Em vez

disso, após alguns ciclos de atualizações de roteamento, todos os roteadores concordarão que a rede está inatingível. Entretanto, a heurística do horizonte dividido não impede loops de encaminhamento em todas as topologias possíveis como um dos exercícios sugere.

Outra maneira de pensar no problema da convergência lenta é em termos do fluxo de informações. Se um roteador anunciar uma rota curta para a mesma rede, todos os roteadores receptores responderão rapidamente para instalar essa rota. Se um roteador parar de anunciar uma rota, o protocolo precisará depender de um mecanismo de timeout antes que considere a rota inatingível. Uma vez que o timeout ocorra, o roteador encontrará uma rota alternativa e começará a propagar essa informação. Infelizmente, um roteador não pode saber se a rota alternativa depende da rota que acabou de desaparecer. Portanto, as informações negativas nem sempre se propagam rapidamente. Um breve epigrama a seguir captura a ideia e explica o fenômeno.

> *Em protocolos de roteamento, boas novas viajam rapidamente; notícias ruins viajam lentamente.*

Outra técnica usada para resolver o problema da convergência lenta emprega o *hold down*. Ela força um roteador participante a ignorar informações sobre uma rede por um período de tempo fixo após o recebimento de uma mensagem que afirma que a rede é inatingível. Para o RIP, o período de hold down é definido em 60 segundos. Duas vezes maior que o tempo normal de atualização. A ideia é esperar tempo o bastante para garantir que todas as máquinas recebam as más notícias e não aceitem erroneamente uma mensagem que está desatualizada. É importante notar que todas as máquinas participantes em uma troca RIP precisam usar noções idênticas de hold down, ou podem ocorrer loops de encaminhamento. A desvantagem dessa técnica é que, se ocorrerem loops de encaminhamento, eles serão preservados pela duração do período de hold down. Mais importante, a técnica preserva todas as rotas incorretas durante o período de hold down, mesmo quando existem alternativas.

Uma última técnica para resolver o problema da convergência lenta é chamada *poison reverse*. Uma vez que a conexão desaparece, o roteador anunciando a conexão mantém a entrada por vários períodos de atualização e inclui um custo de rota infinito em seus broadcasts. Para tornar o poison

reverse mais eficiente, é necessário combiná-lo com *atualizações engatilhadas* (*triggered updates*). Os mecanismos de atualizações engatilhadas forçam um roteador a enviar um broadcast imediato quando recebe más notícias, em vez de esperar o próximo broadcast periódico. Enviando uma atualização imediatamente, um roteador minimiza o tempo em que fica vulnerável (isto é, o tempo durante o qual os vizinhos podem anunciar rotas curtas, pois ainda não receberam as más notícias).

Infelizmente, embora as técnicas de atualizações engatilhadas, poison reverse, hold down e horizonte dividido resolvam alguns problemas, elas introduzem outros. Por exemplo, considere o que ocorre com as atualizações engatilhadas quando muitos roteadores compartilham uma rede comum. Um único broadcast pode mudar todas as suas tabelas de roteamento, engatilhando um novo ciclo de broadcasts. Se o segundo ciclo de broadcasts mudar as tabelas, irá gerar ainda mais broadcasts, e isso pode resultar em uma avalanche de broadcasts.*

O uso do broadcast, o potencial para loops de encaminhamento, e o uso de hold down para prevenir a convergência lenta podem tornar o RIP extremamente ineficiente em uma rede remota. O broadcast sempre requer largura de banda substancial. Mesmo se nenhum problema de avalanche ocorrer, fazer todas as máquinas difundirem periodicamente significa que o tráfego aumenta à medida que aumenta o número de roteadores. O potencial para loops de encaminhamento também pode ser fatal quando a capacidade da linha é limitada. Uma vez que a linha se torna saturada por pacotes de looping, pode ser difícil ou impossível para os roteadores trocarem as mensagens de roteamento necessárias a fim de interromper os loops. Além disso, em uma rede remota, os períodos de hold down são tão longos que os timers usados pelos protocolos de nível superior podem expirar e levar à interrupção das conexões.

14.6 Formato de mensagem RIP (IPv4)

As mensagens RIP podem ser classificadas em dois tipos gerais: mensagens de informação de roteamento e mensagens usadas para requisitar informações. Os dois tipos usam o mesmo formato, que consiste em um cabeçalho fixo seguido de uma lista opcional de redes e pares de distância. A Figura 14.5 mostra o formato de mensagem usado com a versão 2 do RIP (*RIP2*), a versão atual.

Na figura, o campo COMANDO especifica uma operação; apenas cinco comandos são definidos. A Figura 14.6 lista os comandos e o significado de

cada um.

Figura 14.5 O formato de uma mensagem RIP versão IPv4. Após o cabeçalho de 32 bits, a mensagem contém uma sequência de pares, em que cada par especifica um prefixo IPv4, próximo salto, e distância ao destino.

Comando	Significado
1	Requisição de informações de roteamento parciais ou completas
2	Resposta contendo pares rede-distância da tabela de roteamento do emissor
9	Requisição de atualização (usado com circuitos de demanda)
10	Resposta de atualização (usado com circuitos de demanda)
11	Reconhecimento de atualização (usado com circuitos de demanda)

Figura 14.6 Os comandos usados com RIP. Em implementações típicas, somente o comando 2 é usado.

Embora tenhamos descrito RIP como enviando atualizações de roteamento periodicamente, o protocolo inclui comandos que permitem o

envio de consultas. Por exemplo, um host ou roteador pode enviar um comando *requisição* para solicitar informações de roteamento. Os roteadores podem usar o comando *resposta* para responder à solicitação. Na maioria das vezes, um roteador transmite periodicamente mensagens de resposta não solicitada. O campo VERSÃO, na Figura 14.5, contém o número da versão do protocolo (neste caso *2*), e é usado pelo receptor para verificar que ele interpretou corretamente a mensagem.

14.7 Campos de uma mensagem RIP

Como ele foi usado inicialmente com endereços outros que não o IPv4, o RIP usa um campo de FAMÍLIA DE REDE (FAMILY OF NET) para especificar o tipo de endereço que vem a seguir. Valores para o campo foram adotados do 4.3 BDS Unix; endereços IPv4 têm atribuição família *2*. Dois campos em cada entrada especificam um prefixo IPv4: ENDEREÇO IP DA REDE e MÁSCARA DE SUB-REDE PARA NET. Como era de se esperar, a máscara especifica quais bits no endereço correspondem ao prefixo. O campo PRÓXIMO SALTO PARA REDE especifica o endereço de um roteador que é o próximo salto para a rota. O último campo de cada entrada em uma mensagem RIP, DISTÂNCIA PARA A NET, contém um contador de inteiros da distância até a rede especificada. Como discutido anteriormente, o RIP usa rotas 1-origem, o que significa que uma rede diretamente conectada está um salto à frente. Além disso, devido ao RIP interpretar *16* como infinito (ou seja, não existe rota), todas as distâncias estão limitadas ao range *1* a *16*. Curiosamente, tem atribuição 32 bits, mesmo que somente os cinco bits de baixa ordem sejam usados.

O RIP2 adiciona um campo ETIQUETA DE ROTA PARA REDE (ROUTE TAG FOR NET) de 16 bits a cada entrada. Portanto, a etiqueta de rota fornece um meio para propagar informação adicional tal como a origem da rota. Em particular, se o RIP2 aprender uma rota de outro sistema autônomo, ela pode usar a etiqueta de rota para propagar o número do sistema autônomo.

Adicionalmente ao endereço IPv4 unicast, o RIP usa a convenção que o endereço zero (ou seja, *0.0.0.0*) denota uma *rota-padrão*. O RIP adiciona uma distância métrica a cada rota que anuncia, inclusive à rota-padrão. Portanto, é possível organizar para que dois roteadores anunciem uma rota-padrão (ou seja, uma rota para o resto da internet) a diferentes métricas, fazendo uma delas um caminho primário e a outra um backup.

Para prevenir o RIP de aumentar a carga da CPU de hosts desnecessários, os desenvolvedores permitem que o RIP2 multicast atualizações em vez de fazer o broadcast delas. Além disso, o RIP2 é atribuído com um endereço multicast fixo, 224.0.0.9, que significa que máquinas usando RIP2 não precisam executar IGMP,* Finalmente, o multicast RIP2 está restrito a uma única rede.

Note que uma mensagem RIP não contém um campo de tamanho explícito ou um contador de entradas explícito. Em vez disso, O RIP assume que o mecanismo de entrega subjacente vai dizer ao receptor o tamanho da mensagem. O RIP opera na porta UDP *520*. Embora uma requisição RIP possa se originar em outra porta UDP, a porta de destino para requisições é sempre a *520*, como é a porta de origem da qual se originam as mensagens RIP broadcast.

14.8 RIP para IPv6 (RIPng)

Pode parecer que o campo FAMÍLIA DE NET no projeto original permite o uso de protocolos arbitrários. Entretanto, ao contrário de simplesmente usar o projeto existente, uma nova versão do RIP foi criada para o IPv6. Chamada *RIPng*,** o novo protocolo tem um formato de mensagem inteiramente novo e também opera em uma porta UDP diferente do que o RIP (porta *521* em oposição à porta *520*). A Figura 14.7 ilustra o formato.

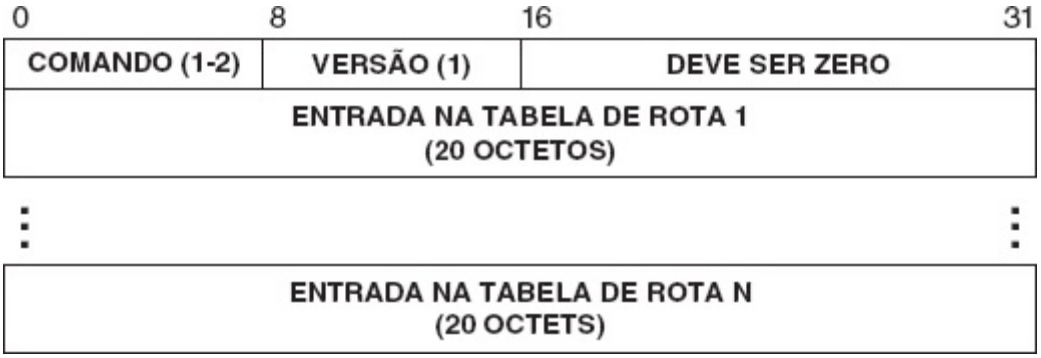


Figura 14.7 O formato geral de uma mensagem RIPng usada para levar informação de roteamento IPv6.

Como a RIP2, uma mensagem de RIPng não inclui um campo de tamanho, nem inclui uma contagem de itens em seguida; um receptor calcula o número de entradas de tabela de rota a partir do tamanho do pacote (o qual é obtido a partir de UDP). Como a figura indica, cada

ENTRADA NA TABELA DE ROTA ocupa 20 octetos (ou seja, a figura não acompanha a escala). A Figura 14.8 ilustra o formato de uma entrada na tabela de rota individual.

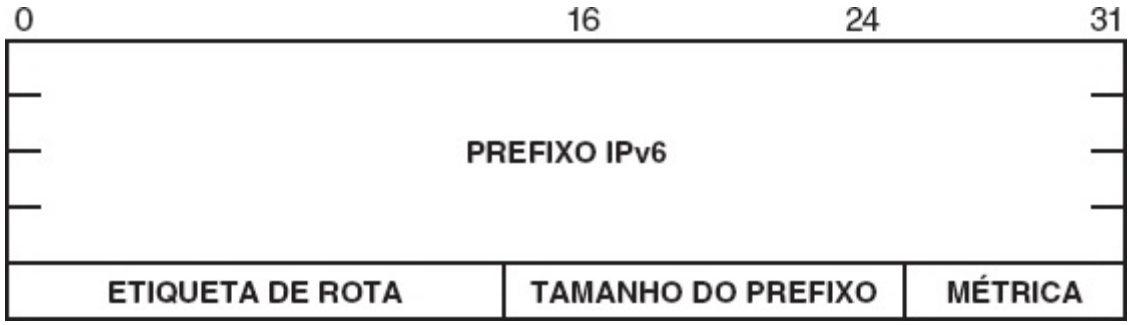


Figura 14.8 O formato de cada ENTRADA DE TABELA DE ROTA em uma mensagem RIPng.

Leitores observadores podem ter notado que o RIPng não inclui um campo que armazena o próximo salto para uma rota. Os desenvolvedores estavam cientes de que, incluir um próximo hop em cada entrada da tabela de rota faria o tamanho da mensagem extremamente grande. Portanto, eles escolheram uma alternativa: uma entrada de tabela de rotas com um campo de métrica de 0xFF especifica um próximo salto em vez de um destino. O próximo salto aplica-se a todas as entradas da tabela de rotas que se seguem, até outra entrada de próximo salto ou o fim da mensagem.

À exceção do novo formato de mensagem, o uso de endereços IPv6 e o fornecimento especial para um próximo hop, o RIPng se assemelha ao RIP. As mensagens ainda são enviadas via UDP, e RIP ainda transmite uma atualização de roteamento a cada 30 segundos e usa um tempo limite de 180 segundos antes de considerar uma rota expirada. O RIP também preserva as técnicas de horizonte dividido, poison reverse, e atualizações engatilhadas.

14.9 A desvantagem de usar contador de saltos

Usar o RIP ou o RIPng como um protocolo de gateway interno limita o roteamento de duas maneiras. Mesmo no melhor dos casos, uma contagem de saltos promove somente uma medida grosseira da capacidade ou responsabilidade da rede. Sabemos que usar contagem de saltos nem sempre fornece rotas com menor atraso ou maior capacidade. Além disso, calcular rotas com base em contagem de saltos mínimos tem a grande desvantagem que isso torna o roteamento relativamente estático, pois as rotas não podem responder a mudanças na carga da rede. Portanto, pode

parecer estranho que um protocolo tenha sido desenvolvido para usar uma métrica de contagem de saltos. A próxima seção considera uma métrica alternativa e explica por que a métrica de contagem de salto continua popular a despeito de suas limitações.

14.10 Métrica de atrasos (HELLO)

Embora agora obsoleto, o protocolo HELLO fornece um exemplo de um IGP que já foi empregado na Internet e usa uma métrica de roteamento diferente da contagem de saltos. Cada mensagem HELLO leva informação de estampa de tempo assim como informação de roteamento, que permitem aos roteadores usando HELLO sincronizarem seus clocks. Curiosamente, o HELLO usava os clocks sincronizados para encontrar o atraso no link entre cada par de roteadores de forma que cada roteador pudesse calcular os caminhos com menor atraso para seus destinos.

A ideia básica por trás do HELLO é simples: usar um algoritmo de vetor de distância para propagar informações de roteamento. Em vez de fazer os roteadores relatarem uma contagem de saltos, o HELLO relata uma estimativa do atraso até o destino. Ter clocks sincronizados permite que um roteador estime o atraso colocando uma estampa de tempo em cada pacote: antes de enviar um pacote, o emissor coloca o valor de clock atual no pacote como um estampa de tempo, e o receptor subtrai esse valor do valor de seu clock atual. Ter clocks sincronizados permite que o atraso seja calculado sem se basear em amostras de ida e volta, o que significa que o atraso em cada direção pode ser estimado de forma independente (ou seja, congestionamento em uma direção não vai afetar a estimativa de atraso na outra direção).

O HELLO usa o método de vetor de distância padrão para atualização. Quando uma mensagem chega da máquina X, o receptor examina cada entrada na mensagem e muda o próximo salto para X se a rota através de X tiver um custo menor do que a rota atual (ou seja, se o atraso até X mais o atraso de X até o destino for menor do que o atraso atual até o destino).

14.11 Métrica de atrasos, oscilação e route flapping

Pode parecer que usar atraso como uma métrica de roteamento produziria melhores rotas do que uma contagem de saltos. Na verdade, o HELLO funcionava bem no antigo backbone da Internet. Entretanto, há uma importante razão por que o atraso não é usado como uma métrica nos

protocolos atuais: a instabilidade.

Mesmo que dois caminhos tenham características idênticas, qualquer protocolo que mude de rota rapidamente pode se tornar instável. A instabilidade surge porque o atraso, diferentemente das contagens de saltos, não é fixo. Pequenas variações nas medições do atraso ocorrem por causa da oscilação do clock de hardware, da carga de CPU durante a medição ou dos atrasos de bit causados pela sincronização de link. Portanto, se um protocolo de roteamento reagir rapidamente a ligeiras diferenças no atraso, ele pode produzir um efeito de oscilação de dois estágios em que o tráfego alterna entre os caminhos alternativos. No primeiro estágio, o roteador descobre o atraso no caminho *1* ligeiramente menor e abruptamente muda o tráfego para ele. No próximo ciclo, o roteador descobre que mudar a carga para o caminho *1* aumentou o atraso e que o caminho *2* tem agora um atraso ligeiramente menor. Portanto, o roteador muda o tráfego de volta para o caminho *2* e a situação se repete.

Para ajudar a evitar oscilação, os protocolos que usam atraso implementam várias heurísticas. Primeiro, eles empregam a técnica de *hold down*, discutida anteriormente, para evitar que as rotas mudem rapidamente. Segundo, em vez de medir da forma mais precisa possível e comparar os valores diretamente, os protocolos arredondam as medições para grandes múltiplos ou implementam um *limiar* mínimo ignorando diferenças abaixo desse limiar. Terceiro, em vez de comparar cada medição de atraso individual, eles mantêm uma *média* ativa dos recentes valores ou aplicam alternativamente uma regra *K-de-N* que exige que pelo menos *K* das medições de atraso *N* mais recentes sejam menores que o atraso atual antes que a rota seja alterada.

Mesmo com a heurística, os protocolos que usam atraso podem se tornar instáveis ao compararem atrasos em caminhos que não têm características idênticas. Para entender por que, é necessário saber que o tráfego pode ter um efeito dramático sobre o atraso. Sem tráfego algum, o atraso de rede é simplesmente o tempo necessário para o hardware transferir bits de um ponto para outro. No entanto, conforme aumenta a carga de tráfego imposta sobre a rede, os atrasos começam a surgir porque os roteadores no sistema precisam enfileirar pacotes que estão esperando para serem transmitidos. Se a carga for ainda que ligeiramente maior que 100% da capacidade de rede, a fila se torna descontrolada, significando que o atraso efetivo se torna

infinito. Podemos então fazer o resumo a seguir.

> *O retardo efetivo através de uma rede depende do tráfego. Conforme a carga aumenta para 100% da capacidade da rede, o atraso cresce rapidamente.*

Como os atrasos são extremamente sensíveis a mudanças na carga, os protocolos que usam atraso como métrica podem facilmente cair em um *ciclo de feedback positivo*. O ciclo é engatilhado por uma pequena mudança externa na carga (por exemplo, um computador injetando uma rajada de tráfego adicional). O tráfego elevado aumenta o atraso, o que faz com que o protocolo mude de rota. Entretanto, como uma mudança de rota afeta a carga, ela pode produzir uma mudança ainda maior em atrasos, o que significa que o protocolo recalculará novamente as rotas. Como resultado, os protocolos que usam atraso precisam conter mecanismos para reduzir a oscilação.

Descrevemos a heurística como podendo resolver casos simples de oscilação de rota quando caminhos possuem características taxa de transferência idênticas e a carga não é excessiva. A heurística pode se tornar ineficiente, no entanto, quando os caminhos alternativos possuem características de atraso e taxa de transferência diferentes. Como exemplo, considere o atraso em dois caminhos: um por um satélite e outro por uma linha de circuito digital de baixa capacidade (por exemplo, um circuito fracional T1). No primeiro estágio do protocolo, quando ambas as partes estão ociosas, o circuito digital parecerá ter atraso significativamente menor do que o satélite e será escolhido para o tráfego. Como o circuito digital tem baixa capacidade, ele rapidamente se tornará sobrecarregado e o atraso aumentará abruptamente. No segundo estágio, o atraso na linha serial será muito maior que o do satélite; portanto, o protocolo retirará o tráfego do caminho congestionado. Como o caminho do satélite possui grande capacidade, o tráfego que sobrecarregou a linha serial não impõe uma carga significativa sobre o satélite, significando que o atraso no caminho do satélite não muda com o tráfego. No próximo ciclo, o atraso do circuito digital, não carregado, novamente parecerá ser muito menor do que o atraso no caminho do satélite. O protocolo inverterá o roteamento, e o ciclo continuará. Essas oscilações, na verdade, ocorrem na prática. Como mostra o exemplo, elas são difíceis de manejar porque o tráfego que tem pouco efeito sobre uma rede pode sobrecarregar outro. O ponto importante é

descrito a seguir.

> *Embora a intuição sugira que o roteamento use caminhos com menores atrasos, fazer isso torna o roteamento sujeito a oscilações conhecidas como route flapping.*

14.12 O protocolo Open SPF (OSPF)

No Capítulo 12, dissemos que uma técnica de roteamento de estado de link, que usa um algoritmo gráfico SPF para calcular os caminhos mais curtos, escala melhor do que um algoritmo de vetor de distância. Para encorajar a adoção da tecnologia de estado de link, um grupo de trabalho do IETF projetou um protocolo de gateway interno que usa o algoritmo de estado de link. Chamado *Open SPF* (*OSPF*), o protocolo possui várias metas ambiciosas.

- *Padrão aberto*. Como o nome implica, a especificação está disponível na literatura publicada. Torná-lo um padrão aberto que qualquer um pode implementar sem pagar taxas de licença encorajou muitos fornecedores a aceitar o OSPF.
- *Tipo de roteamento de serviço*. Os gerentes podem instalar múltiplas rotas para um determinado destino, uma para cada prioridade ou tipo de serviço. Um roteador executando OSPF pode usar ambos os endereços de destino e tipo de serviço quando estiverem escolhendo uma rota.
- *Balanceamento de carga*. Se um gerente especifica múltiplas rotas, a um mesmo custo, para um determinado destino, o OSPF distribui o tráfego igualmente por todas as rotas.
- *Subdivisão hierárquica em áreas*. Para lidar com grandes intranets e limitar a sobrecarga de roteamento, o OSPF permite que um site particione suas redes e roteadores em subconjuntos chamados áreas. Cada área é autossuficiente; a topologia de uma área permanece oculta das outras áreas. Portanto, múltiplos grupos dentro de um dado site podem cooperar no uso do OSPF para roteamento, mesmo que cada grupo retenha a habilidade de mudar sua topologia interna de rede independentemente.

- *Suporte para autenticação*. O OSPF permite que todas as trocas entre roteadores sejam *autenticadas*. Ele suporta uma variedade de esquemas de autenticação e permite que uma área escolha um esquema diferente do de outra área.
- *Granularidade arbitrária*. O OSPF inclui suporte para host específico, sub-rede específica, rede específica e roteadores-padrão.
- *Suporte para redes multiacesso*. Para acomodar redes tais como Ethernet, o OSPF estendeu o algoritmo SPF. Normalmente, o SPF requer que cada par de roteadores façam o broadcast de mensagens sobre o link entre eles. Se K roteadores se conectam a uma Ethernet, eles vão fazer o broadcast de K^2 mensagens de status. Em vez de um gráfico que usa conexões ponto a ponto, o OSPF reduz os broadcasts por permitir uma topologia gráfica mais complexa em que cada nó representa tanto um roteador como uma rede. Um *gateway designado* (ou seja, um *roteador designado*) envia mensagens de estado de link em nome de todos os roteadores conectados à rede.
- *Entrega multicast*. Para reduzir a carga em sistemas não participantes, o OSPF usa as capacidades do hardware multicast, onde eles existem, para a entrega de mensagens de status de link. O OSPF envia mensagens via IP multicast, e permite que o mecanismo IP multicast mapeie o multicast dentro da rede subjacente; dois endereços IPv4 multicast são pré-atribuídos ao OSPF 224.0.0.5 para todos os roteadores e 224.0.0.6 para todos os nós.
- *Topologia virtual*. Um gerente pode criar uma topologia. Por exemplo, ele pode configurar um link virtual entre dois roteadores no gráfico de roteamento se a conexão física entre os dois exigir comunicação através de múltiplas redes de trânsito.
- *Importação de rota*. O OSPF pode importar e disseminar informação de roteamento que aprendeu de sites externos (ou seja, de roteadores que não usam o OSPF). As mensagens OSPF distinguem entre informação adquirida de fontes externas e informações adquiridas de roteadores internos ao site.
- *Uso direto do IP*. Diferentemente do RIP e do RIPng, as mensagens OSPF são encapsuladas diretamente em datagramas IP. O valor *89* é

usado no campo PROTO (IPv4) ou no campo PRÓXIMO SALTO (IPv6) no cabeçalho para identificar o datagrama que está levando o OSPF.

14.13 Formatos de mensagem (IPv4) OSPFv2

Atualmente, a versão-padrão do OSPF é a versão 2. Essa versão foi criada para o IPv4 e não consegue lidar com o IPv6. Ao contrário do RIP, em que o IETF escolheu criar um protocolo totalmente novo para o IPv6, um grupo de trabalho IETF propôs que as mudanças na OSPFv2 para IPv6 apenas fossem incorporadas em uma nova versão do OSPF, a versão 3. Vamos primeiro examinar os formatos de mensagem da versão 2 utilizados com IPv4, e então considerar os formatos de mensagens da versão 3 usados com IPv6. Para distinguir entre os dois, vamos escrever *OSPFv2* e *OSPFv3*.

Cada tipo de mensagem OSPFv2 começa com um cabeçalho fixo de 24 octetos. A Figura 14.19 ilustra o formato.

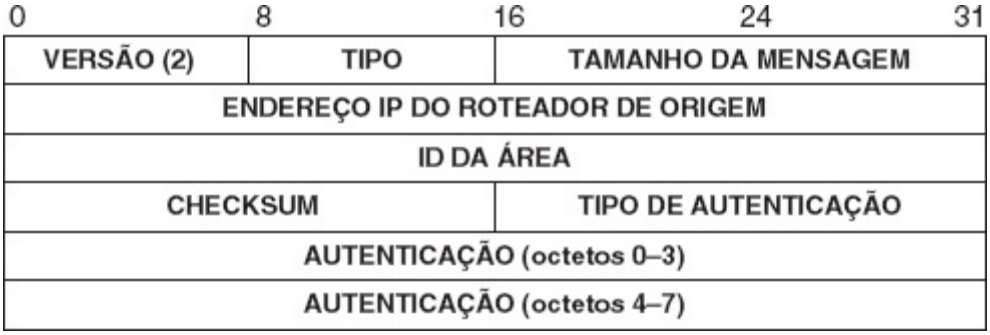


Figura 14.9 O cabeçalho fixo OSPFv2 de 24-octetos que aparece em cada mensagem.

O campo VERSÃO especifica a versão do protocolo como *2*. O campo TIPO identifica o tipo de mensagem como um dos seguintes tipos (que são explicados nas próximas seções):

Tipo	Significado
1	HELLO (usado para testar alcançabilidade)
2	Descrição de banco de dados (topologia)
3	Requisição de estado de link
4	Atualização de estado de link

O campo denominado ENDEREÇO IP DO ROTEADOR DE ORIGEM fornece o endereço do emissor, e o campo denominado ID DE ÁREA fornece o número identificador de 32 bits da área. Por convenção, Área 0 é a área do backbone. Como cada mensagem pode incluir autenticação, o campo TIPO AUTENTICAÇÃO especifica qual esquema de autenticação é usado (por exemplo, *0* significa nenhuma autenticação e *1* significa que uma única senha é usada).

14.13.1 Formato de mensagem HELLO OSPFv2

O OSPFv2 envia mensagens HELLO periodicamente em cada link para estabelecer e testar a alcançabilidade do vizinho. A Figura 14.10 mostra o formato da mensagem. O campo MÁSCARA DE REDE contém uma máscara de endereço para a rede pela qual a mensagem foi enviada. O campo INTERVALO INATIVO DE ROTEADOR fornece o tempo, em segundos, após o qual se um vizinho não responder é considerado inativo. O campo INTERVALO HELLO é o período normal, em segundos, entre mensagens HELLO. O campo PRIO GWAY é a prioridade em inteiros desse roteador e é usado para selecionar um roteador designado de backup. Os campos ROTEADOR DESIGNADO e ROTEADOR DESIGNADO DE BACKUP contêm endereços IP que fornecem a visão do emissor do roteador designado e do roteador de backup designado para a rede para a qual a mensagem é enviada. Finalmente, os campos designados ENDEREÇO IP VIZINHO fornecem os endereços IP de todos os vizinhos dos quais o emissor recebeu mensagens HELLO recentemente.

Figura 14.10 Formato de mensagem HELLO OSPFv2. Um par de roteadores vizinhos troca mensagens periodicamente para testar a alcançabilidade.

14.13.2 Formato de mensagem descrição de banco de dados OSPFv2

Os roteadores trocam mensagens de *descrição de banco de dados* OSPFv2 para inicializar seus bancos de dados de topologia de rede. Na troca, um roteador serve como mestre, enquanto outro serve como escravo. O escravo reconhece cada mensagem de descrição de banco de dados com uma resposta. A Figura 14.11 mostra o formato.

Como o banco de dados de topologia pode ser grande, ele pode ser dividido em várias mensagens usando os bits *I* e *M*. O bit *I* é definido em 1 na mensagem inicial; o bit *M* é definido em 1 se houverem mensagens adicionais em seguida. O bit *S* indica se uma mensagem foi enviada por um mestre (*1*) ou por um escravo (*0*). O campo NÚMERO DE SEQUÊNCIA DE BANCO DE DADOS numera as mensagens sequencialmente para que o receptor possa saber se alguma delas está faltando. A mensagem inicial contém um inteiro aleatório *R*; mensagens subsequentes contêm inteiros sequenciais começando em R.

O campo MTU DE INTERFACE fornece o tamanho do maior datagrama de IP que pode ser transmitido pela interface sem fragmentação. Os campos

de IDADE LS até TAMANHO LS descrevem um link na topologia de rede; eles são repetidos para cada link.

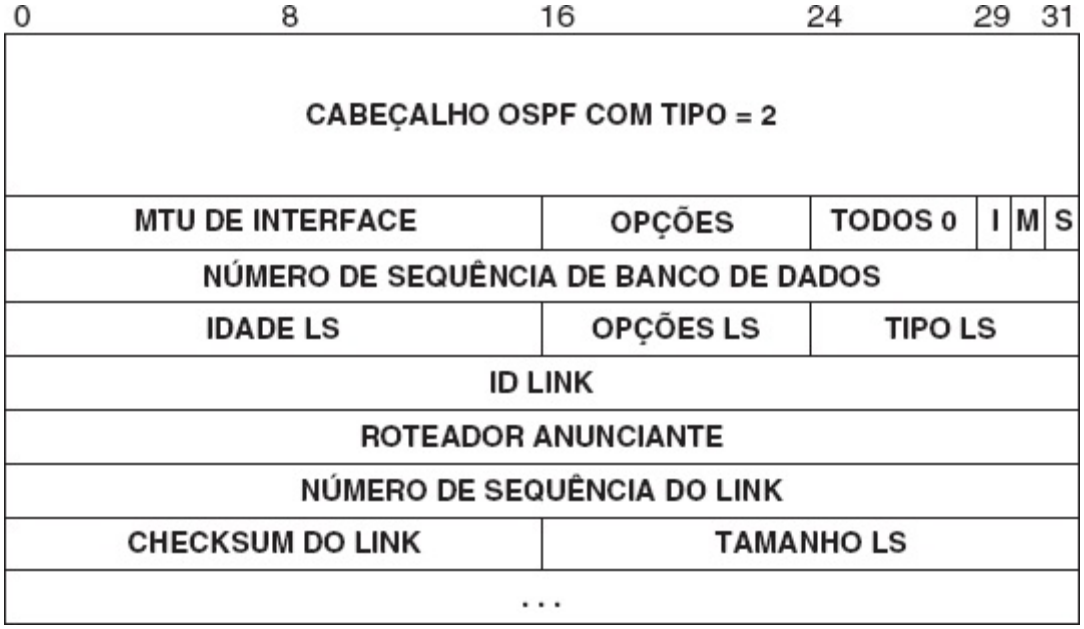


Figura 14.11 Formato de mensagem *descrição de banco de dados* OSPFv2. Os campos começando em IDADE LS são repetidos para cada link sendo especificado.

O campo TIPO LS descreve o tipo de um link. Os valores possíveis são dados de acordo com a seguinte tabela:

Tipo LS	Significado
1	Link de roteador
2	Link de rede
3	Link de resumo (Rede IP)
4	Link de resumo (link para o roteador de borda)
5	Link externo (link para outro site)

O campo ID LINK fornece uma identificação para o link (que pode ser o endereço IP de um roteador ou uma rede, dependendo do tipo de link).

O campo IDADE LS ajuda a ordenar as mensagens – ele fornece o tempo, em segundos, desde que o link foi estabelecido. O campo ROTEADOR ANUNCIANTE especifica o endereço do roteador que está anunciando esse link, e NÚMERO DE SEQUÊNCIA DE LINK contém um inteiro gerado por esse roteador para garantir que as mensagens não serão perdidas ou

recebidas fora de ordem. O campo CHECKSUM DO LINK fornece mais garantia de que as informações de link não foram corrompidas.

14.13.3 Formato de mensagem de requisição de status de link OSPFV2

Após trocar mensagens de descrição de banco de dados com um vizinho, um roteador tem uma descrição inicial da rede. Entretanto, um roteador pode descobrir que partes de seu banco de dados estão desatualizadas. Para requisitar que o vizinho forneça informações atualizadas, o roteador envia uma mensagem *requisição de estado de link* (*Link Status Request*). A mensagem lista links específicos para os quais são necessárias informações, como mostrado na Figura 14.12. O vizinho responde com as informações mais atuais que ele possui sobre os links da mensagem de requisição. Os três campos mostrados na figura são repetidos para cada link sobre o qual o status é solicitado. Mais de uma mensagem de requisição pode ser necessária se a lista de requisições for longa.

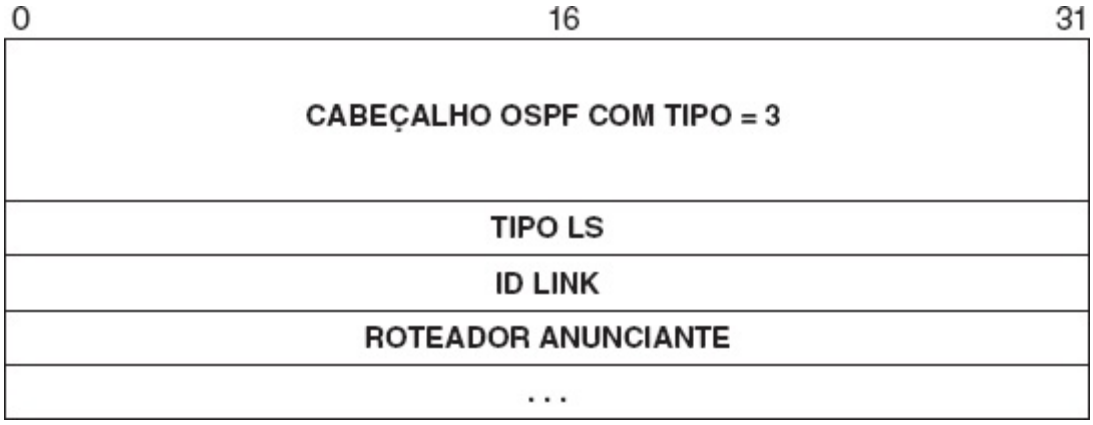


Figura 14.12 Formato de mensagem Link State Request OSPFv2. Um roteador envia esta mensagem a um vizinho para requisitar informações atuais sobre um conjunto de links específicos.

14.13.4 Formato de mensagem atualização de status de link OSPFv2

Como o OSPF usa um algoritmo de estado de link, roteadores devem difundir periodicamente mensagens por broadcast que especificam o status dos links diretamente conectados. Para tanto, os roteadores usam um tipo 4 de mensagem OSPFv2 que é denominado *atualização de status de link*

(*link-status update*). Cada mensagem de atualização consiste de um contador de anúncios seguido por uma lista de anúncios. A Figura 14.3 mostra o formato das mensagens de atualização de status de link.

Na figura, cada *anúncio de status de link* (*link-status advertisement – LSA*) tem um formato que especifica informação sobre a rede que está sendo anunciada. A Figura 14.14 mostra o formato de um anúncio de estado de link. Os valores usados em cada campo são os mesmos que na mensagem de descrição do banco de dados.

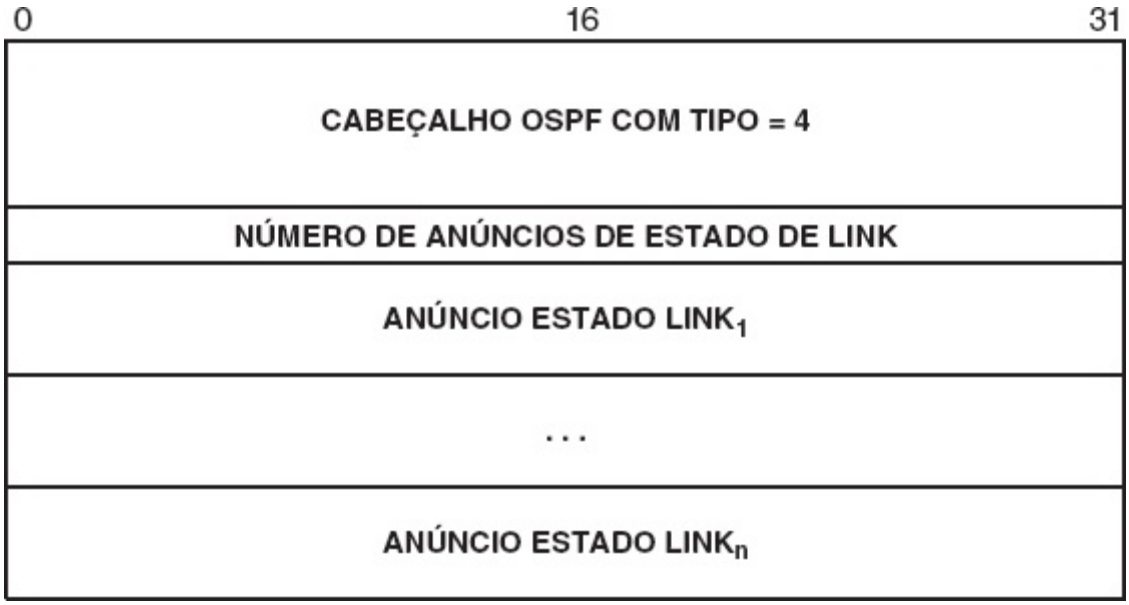


Figura 14.13 Formato de mensagem – Atualização de Status de Link OSPFv2. Um roteador envia essa mensagem por broadcast para difundir informações, a todos os outros roteadores, sobre seus links diretamente conectados.

Seguindo o cabeçalho de estado de link vem um dos quatro formatos possíveis para descrever os links de um roteador para uma determinada área, os links de um roteador com uma rede específica, os links de um roteador com as redes físicas que constituem uma única rede IP com sub-rede (veja o Capítulo 5) ou os links de um roteador com redes em outros sites. Em todos os casos, o campo TIPO LS no cabeçalho de estado de link especifica qual dos formatos foi usado. Portanto, um roteador que recebe uma mensagem de Atualização de Status de Link sabe exatamente quais dos destinos descritos estão dentro do site e quais são externos.

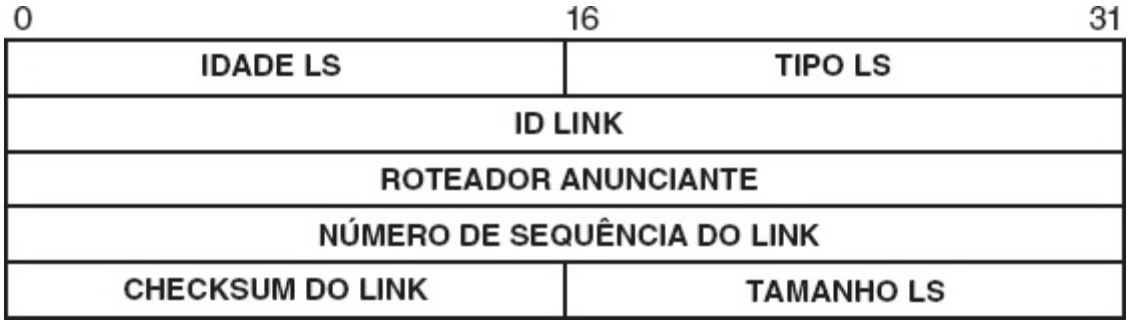


Figura 14.14 Formato do *anúncio de estado de link* usado em uma mensagem de status de link.

14.14 Mudanças no OSPFv3 para suportar o IPv6

Embora os conceitos básicos de OSPF permaneçam os mesmos na versão 3, muitos detalhes mudaram. O protocolo ainda usa a abordagem* link-state. Todo endereçamento foi removido do protocolo básico, tornando-se independente dele, exceto para os endereços IP em anúncios status de link. Em particular, o OSPFv2 usava um endereço IP de 32 bits para identificar um roteador; o OSPFv3 usa um *roteador ID* de 32-bit. Da mesma forma, os identificadores de área permanecem em 32 bits, mas não estão relacionados a endereços IPv4 (embora o ponto decimal seja usado para expressá-los). O OSPFv3 permite que *instâncias* independentes de OSPF executem um conjunto de roteadores e redes ao mesmo tempo. Cada instância tem um único ID, e pacotes levam a instância ID. Por exemplo, seria possível ter uma instância IPv6 propagando informações de roteamento, enquanto outra instância propaga informações de roteamento MPLS. Finalmente, o OSPFv3 remove toda a autenticação de mensagens individuais e ainda conta com o cabeçalho de autenticação IPv6.

A mudança mais significativa entre o OSPFv2 e o OSPFv3 surge a partir dos formatos de mensagens, que mudam todas. Há duas motivações. Em primeiro lugar, as mensagens devem ser alteradas para acomodar os endereços IPv6. Segundo, porque os endereços IPv6 são muito maiores, os desenvolvedores decidiram que simplesmente substituir cada ocorrência de um endereço IPv4 com um endereço IPv6 faria mensagens muito grande. Portanto, sempre que possível, o OSPFv3 minimiza o número de endereços IPv6 transportados em uma mensagem e substitutos identificadores de 32 bits para qualquer identificador que não precisa ser um endereço IPv6.

14.14.1 Formatos de mensagem OSPFv3

Cada mensagem OSPFv3 começa com um cabeçalho de 16 octetos. A Figura 14.15 ilustra o formato.

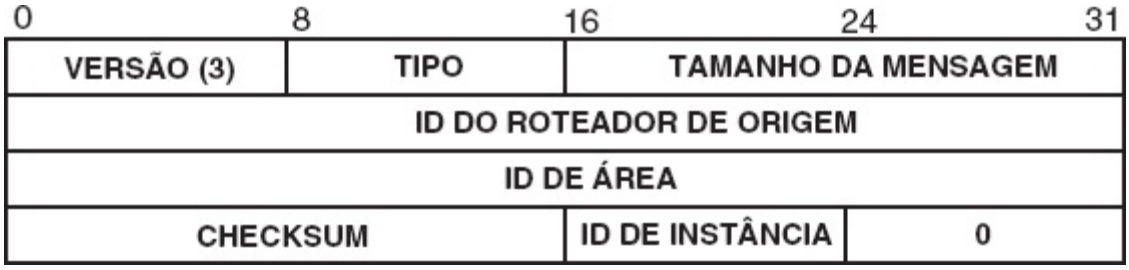


Figura 14.15 O cabeçalho fixo OSPFv3 de16-octetos que aparece em cada mensagem.

Note que o número da versão ocupa o primeiro octeto, exatamente como no OSPFv2. Portanto, mensagens OSPFv3 podem ser enviadas usando o mesmo valor PRÓXIMO CABEÇALHO OSPFv2 sem ambiguidade. Note também que o cabeçalho fixo é menor que o cabeçalho OSPFv2, pois a informação de autenticação foi removida.

14.14.2 Formato de mensagem HELLO OSPFv3

A mensagem *hello* OSPFv3 ajuda a ilustrar a mudança básica de endereçamento IPv4 para identificadores de 32 bits. O objetivo é manter o tamanho do pacote pequeno, separando o protocolo do IPv4. Como a Figura 14.16 ilustra, os leitores devem comparar o formato da versão 3 para o formato da versão 2 mostrado na página 208.

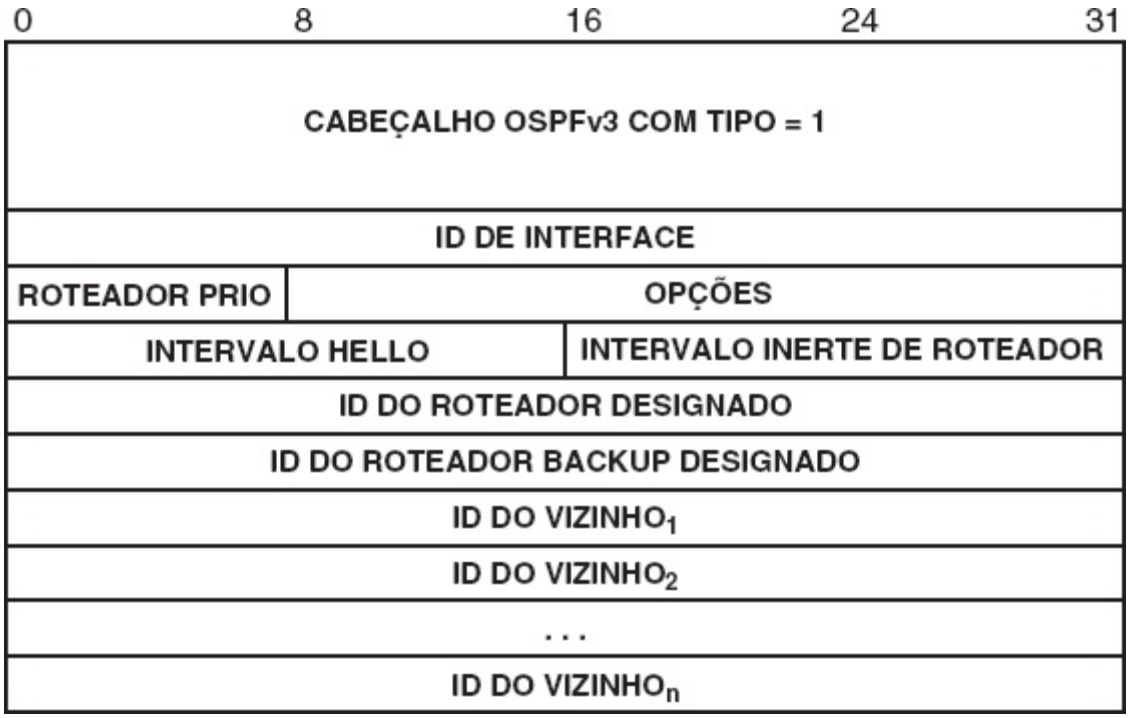


Figura 14.16 O formato de mensagem HELLO OSPFv3. Todos os endereços IPv4 foram substituídos por identificadores de 32 bits.

14.14.3 Outras características e mensagens OSPFv3

O OSPFv3 combina e generaliza muitos dos recursos e características que foram definidos para o OSPFv2. Consequentemente, o OSPFv3 define vários tipos de *anúncio de status de link* (*link-status advertisement – LSA*). Por exemplo, o OSPFv3 suporta LSAs roteador LSA, link LSA, prefixo interárea LSA, roteador interárea LSA, AS externo LSA, prefixo intra-área LSA, e *Not To Stubby Area* (*NSSA*) LSA. Cada mensagem de status de link começa com um cabeçalho que é o mesmo do OSPFv2 ilustrado na Figura 14.14, e utiliza o tipo de campo para identificar os conteúdos restantes.

O ponto de oferecer vários tipos de LSA é apoiar grandes sistemas autônomos que têm uma topologia complexa e regras complexas para as áreas. Em particular, provedores Tier-1 usam OSPF como um IGP através de uma intranet, que inclui um backbone, muitas redes regionais, e muitas redes conectadas.

14.15 Protocolo de propagação de rota IS-IS

Por volta da mesma época, o IETF definiu o OSPF e a Digital Equipment

Corporation desenvolveu um protocolo de propagação de rotas internas chamado IS-IS.* O *IS-IS* fazia parte do conjunto de protocolos da *Fase V DECnet* da Digital, e mais tarde foi normatizado pela ISO em 1992 para uso nos agora extintos *protocolos OSI*. O nome se expandiu para *Intermediate System– Sistema Intermediário –*, e é equivalente à nossa definição de um Protocolo de Gateway Interno.

O IS-IS e o OSPF são conceitualmente muito próximos; diferindo apenas em detalhes. Ambos usam o algoritmo link-state, ambos exigem cada roteador participando de propagar status de link, mensagens para roteadores diretamente conectados e ambos usam mensagens de status do link de entrada para construir um banco de dados de topologia. Ambos os protocolos permitem que mensagens de status possam ser multicast se a rede subjacente suportar. Além disso, ambos os protocolos usam o algoritmo Caminho Mais Curto Primeiro (Shortest Path First) para calcular caminhos mais curtos.

Ao contrário do OSPF, o IS-IS não foi originalmente concebido para lidar com IP. Portanto, foi mais tarde estendido, e a versão estendida é conhecida como *IS-IS* integrado (*Integrated*) ou *Dual IS-IS*. Como ele foi estendido para lidar com IP, tem a vantagem de não ser *integrado* com o IPv4. Assim, ao contrário do OSPF, que exigiu uma nova versão para lidar com IPv6, o Dual IS-IS acomoda o IPv6 como mais uma família de endereços. Difere também porque *não* usa IP para a comunicação. Em vez disso, seus pacotes são encapsulados em frames de rede e enviados diretamente através da rede subjacente.

Como o OSPF, o IS-IS permite aos gerentes subdividir roteadores em áreas. No entanto, a definição de uma área difere da OSPF. Em particular, o IS-IS não requer que um ISP defina a área de 0 para ser uma rede backbone através da qual flui todo o tráfego. Em vez disso, IS-IS define um roteador como *Nível 1* (intra-área), *Nível 2* (interárea) ou *Nível 1-2* (ambos tanto intra-área e interárea). Um roteador Nível 1 só se comunica com outros roteadores Nível 1 na mesma área. Um roteador Nível 2 só se comunica com roteadores Nível 2 em outras áreas. Um roteador Nível 1-2 conecta-se aos outros dois conjuntos. Assim, ao contrário do OSPF, que impõe uma topologia em forma de estrela, o IS-IS permite que o centro seja um conjunto de redes nível 2.

Os defensores do OSPF salientam que OSPF foi estendido para lidar com muitos casos especiais que surgem em um grande ISP. Por exemplo, o

OSPF tem mecanismos para lidar com as redes stub, redes not-so-stub e comunicação com outros protocolos IETF. Os partidários do IS-IS salientam que ele é menos "tagarela" (ou seja, envia menos mensagens por unidade de tempo), e pode lidar com áreas maiores (ou seja, áreas com mais roteadores). Assim, o IS-IS é considerado uma alternativa adequada ao OSPF para casos especiais.

14.16 Confiança e desvio de rota

Já observamos que um único roteador pode usar um Protocolo de Gateway Interno para reunir informações de roteamento dentro de seu sistema autônomo e um Protocolo de Gateway Externo para anunciar rotas para outros sistemas autônomos. Em princípio, deveria ser fácil construir uma única peça de software que combinasse informações dos dois protocolos, o que tornaria possível reunir rotas e anunciá-los, sem intervenção humana. Na prática, obstáculos técnicos e políticos tornam isso complexo.

Tecnicamente, os protocolos IGP, como RIP e HELLO, são protocolos de roteamento. Um roteador usa tais protocolos para atualizar a sua tabela de encaminhamento com base em informações que ele adquire a partir de outros roteadores dentro de seu sistema autônomo. Assim, softwares RIP ou OSPF mudam a tabela de encaminhamento local quando novas atualizações de roteamento chegam carregando novas informações. O IGP confia nos roteadores dentro do mesmo sistema autônomo para passar dados corretos.

Em contraste, os protocolos exteriores, como BGP não confiam em roteadores arbitrários e não revelam todas as informações da tabela de encaminhamento local. Em vez disso, os protocolos externos mantêm um banco de dados de alcançabilidade de rede e aplicam restrições políticas ao enviar ou receber informações. Ignorar essas restrições políticas pode afetar o roteamento em um sentido mais amplo – algumas partes da Internet podem se tornar inalcançáveis. Por exemplo, se acontecer de um roteador em um sistema autônomo que esteja executando um IGP propagar uma rota de baixo custo para a rede da Universidade Purdue, quando ele não tem essa rota, outros roteadores que recebem o anúncio aceitam e instalam a rota. Consequentemente, os roteadores dentro do SA onde o erro ocorreu encaminham o tráfego para Purdue incorretamente; Purdue pode tornar-se inacessível a partir de redes dentro do SA. O problema se torna mais grave se protocolos de gateway externo propagarem informações incorretas – se um SA reivindica incorretamente ter rota para um determinado destino, este pode se tornar inacessível em toda a Internet . Dizemos que o endereço de destino foi *sequestrado* (*hijacked*).

14.17 Gated: um daemon gateway de roteamento

Um mecanismo foi criado para fornecer uma interface entre um grande conjunto de protocolos de roteamento, como o RIP, RIPng, BGP, HELLO, e OSPF. O mecanismo também inclui informações de roteamento aprendidas via ICMP e ICMPv6. Conhecido como *gated*,* o mecanismo compreende vários protocolos (ambos os protocolos de gateway interior e exterior, incluindo BGP), e assegura que as restrições políticas são honradas. Por exemplo, *gated* pode aceitar mensagens RIP e modificar a tabela de encaminhamento do computador local. Também pode anunciar rotas de dentro de seu sistema autônomo utilizando BGP. Seguir as regras *gated* permite que um administrador de sistema especifique exatamente quais redes *gated* podem e não podem fazer propaganda e como relatar distâncias para essas redes. Assim, embora não seja um IGP, que desempenha um papel importante no roteamento porque demonstra que é possível construir um mecanismo automatizado ligando um IGP com BGP sem sacrificar a proteção.

O *gated* tem uma história interessante. Ele foi criado originalmente por Mark Fedor em Cornell, e foi adotado por MERIT para uso com o backbone NSFNET. Pesquisadores acadêmicos contribuíram com novas ideias, um consórcio industrial foi formado e, finalmente, a MERIT vendeu o *gated* para a Nexthop.

14.18 Métricas artificiais e métricas de transformação

No capítulo anterior, foi dito que os ISPs muitas vezes escolhem rotas por razões econômicas, em vez de técnicas. Para isso, os gerentes de rede configuram protocolos de roteamento manualmente e atribuem pesos ou distâncias artificiais, o que o software de protocolo usa no lugar de pesos ou distâncias reais. Considere uma rede usando RIP. Se um gerente quiser direcionar o tráfego ao longo de um caminho que tem mais saltos do que o caminho ideal, ele pode configurar um roteador para especificar que o caminho ideal seja o de vários saltos a mais. Por exemplo, um roteador pode ser configurado para fazer anúncio de uma rede diretamente conectada como a uma distância 5. Da mesma forma, ao usar OSPF, a cada link deve ser atribuído um peso. Em vez de basear o peso sobre a capacidade da rede subjacente, um gestor pode escolher pesos artificiais que fazem o software de protocolo preferir um caminho no lugar outro. Nos maiores ISP, a atribuição de pesos artificiais é importante porque tem uma

relação direta e significativa para as receitas. Portanto, grandes provedores costumam contratar pessoas talentosas cujo trabalho é analisar todo o roteamento e escolher os pesos que irão otimizar a receita.

Softwares como *gated* ajudam os gerentes de rede a controlarem o roteamento, oferecendo transformações métricas. Um gerente pode colocar tal software entre dois grupos de roteadores que usam cada um IGP e configurá-lo para transformar métricas de forma que o roteamento prossiga como desejar. Por exemplo, pode haver uma via de baixo custo utilizada dentro de um grupo que é reservada para uso interno. Para evitar que pessoas de fora usem a rota reservada, o software na borda entre os dois grupos infla artificialmente o custo da rota antes de anunciá-lo externamente. Assim, usuários externos acham que o caminho é caro e escolhem uma alternativa. O ponto principal é descrito a seguir.

Embora tenhamos descrito protocolos de roteamento buscando caminhos mais curtos, o software de protocolo geralmente inclui opções de configuração que permitem que um gerente de rede possa substituir os custos reais por valores artificiais que farão com que o tráfego siga as rotas que o gerente prefere.

É claro que um gerente poderia alcançar o mesmo resultado configurando manualmente as tabelas de encaminhamento em todos os roteadores. Usar métricas artificiais tem uma vantagem significativa: se a rede falhar, o software irá selecionar automaticamente uma rota alternativa. Portanto, os gestores se concentram em configurar métricas, em vez de configurar as tabelas de encaminhamento.

14.19 Roteamento com informações parciais

Começamos nossa discussão da arquitetura de roteador de internet e roteamento examinando o conceito de informações parciais. Os hosts podem rotear apenas com informações parciais porque eles se baseiam em roteadores. Agora deve estar claro que nem todos os roteadores possuem informações completas. A maioria dos sistemas autônomos possui um único roteador que conecta o sistema autônomo a outros sistemas autônomos. Por exemplo, se o site se conecta à Internet global, pelo menos um roteador precisa ter uma conexão que vai do site até um ISP. Os roteadores dentro do sistema autônomo conhecem os destinos dentro desse sistema autônomo,

mas eles usam uma rota-padrão para enviar qualquer outro tráfego para o ISP.

Como fazer roteamento com informações parciais se torna óbvio se examinarmos as tabelas de roteamento de um roteador. Os roteadores no centro da Internet possuem um conjunto completo das rotas para todos os destinos possíveis; esses roteadores não usam roteamento-padrão. Os roteadores além desses ISPs no centro da Internet normalmente não possuem um conjunto completo de rotas; eles se baseiam em uma rota-padrão para lidar com os endereços de rede que não entendem.

Usar rotas-padrão para a maioria dos roteadores traz duas consequências. Primeiro, isso significa que os erros de roteamento locais podem passar despercebidos. Por exemplo, se uma máquina em um sistema autônomo rotear incorretamente um pacote para um sistema autônomo externo em vez de para um roteador local, o sistema externo poderá roteá-lo de volta (talvez para um ponto de entrada diferente). Assim, a conectividade pode parecer estar preservada mesmo se o roteamento estiver incorreto. O problema pode não parecer grave para pequenos sistemas autônomos que têm redes de área local de alta velocidade. Mas, em uma rede remota, rotas incorretas podem ser desastrosas, pois o caminho que os pacotes seguem podem envolver múltiplos ISPs, o que causa grandes atrasos, e os ISPs ao longo do caminho podem estar carregados de trânsito, o que resulta em perdas de receita desnecessárias. Segundo, pelo lado positivo, usar rotas-padrão sempre que possível significa que as mensagens de atualização de roteamento trocadas pela maioria dos roteadores serão muito menores do que seriam se informações completas fossem incluídas.

14.20 Resumo

O proprietário de um sistema autônomo é livre para escolher os protocolos que passam informação de roteamento entre os roteadores locais dentro do sistema autônomo. A manutenção manual das informações de roteamento é suficiente apenas para internets pequenas, que mudam lentamente e possuem interconexão mínima; a maioria requer procedimentos automatizados que descobrem e atualizam rotas automaticamente. Usamos o termo Protocolo de Gatewat Interno (IGP) para nos referirmos ao protocolo que é usado para trocar informação de roteamento dentro de um Sistema Autônomo.

Um IGP implementa a ambos o algoritmo de vetor de distância ou o algoritmo de estado de link, que é conhecido pelo nome Menor Caminho Primeiro (Shortest Path First – SPF). Examinamos três IGPs específicos: RIP, HELLO e OSPF. O RIP é um protocolo de vetor de distância que usa as

técnicas de horizonte dividido, hold-down, e poison reverse para ajudar a eliminar loops de encaminhamento e o problema de contagem ao infinito. Embora obsoleto, o HELLO é interessante porque ilustra um protocolo de vetor de distância que utiliza atraso em vez das contagens de saltos como métrica de distância.

Discutimos as desvantagens de usar atraso como uma métrica de roteamento e enfatizamos que, embora a heurística possa prevenir instabilidades quando os caminhos apresentam características de transferência de dados semelhantes, as instabilidades de longo prazo surgem quando caminhos têm características diferentes. O OSPF implementa o algoritmo do estado de link e vem em duas versões: OSPFv2 para IPv4 e OSPFv3 para IPv6. O IS-IS é uma alternativa que lida, melhor do que o OSPF, com alguns casos especiais.

Embora os protocolos de roteamento sejam descritos como computação de caminhos mais curtos, o software de protocolo inclui opções de configuração que permitem aos gerentes inflarem artificialmente os custos. Ao configurar os custos com cuidado, um gestor pode direcionar o tráfego ao longo de caminhos que implementam a política corporativa ou gerar mais receita, enquanto continua a ter a capacidade de rota ao longo de caminhos alternativos automaticamente quando o equipamento de rede falhar.

EXERCÍCIOS

14.1 Quais famílias de rede o RIP aceita? Por quê?

14.2 Considere um grande sistema autônomo usando um IGP que baseia as rotas em atraso. Que dificuldade esse sistema autônomo terá se um subgrupo decidir usar o RIP em seus roteadores? Explique.

14.3 Dentro de uma mensagem RIP, cada endereço IP é alinhado em um limite de 32 bits. Esses endereços serão alinhados em um limite de 32 bits se o datagrama IP transportando a mensagem iniciar em um limite de 32 bits? Sim ou não?Por quê?

14.4 Um sistema autônomo pode ser tão pequeno quanto uma única rede local ou tão grande quanto múltiplas redes de longo alcance. Por que a variação no tamanho torna difícil definir um único IGP que funcione bem em todas as situações?

14.5 Caracterize as circunstâncias sob as quais a técnica de horizonte dividido evitará a convergência lenta.

14.6 Considere uma internet composta de muitas redes locais executando

RIP como um IGP. Encontre um exemplo que mostra como um loop de encaminhamento pode ocorrer mesmo se o código usar "hold-down" após receber informações de que uma rede está inalcançável.

14.7 Um host deve executar o RIP em modo ativo? Sim ou não? Por quê?

14.8 Sob que circunstâncias uma métrica de contagem de saltos produz melhores rotas do que uma métrica que usa atraso?

14.9 Você pode imaginar uma situação em que um sistema autônomo escolhe *não* anunciar todas as suas redes? (Dica: pense em uma universidade.)

14.10 Em termos gerais, poderíamos dizer que um IGP distribui a tabela de roteamento local, enquanto o BGP distribui uma tabela das redes e roteadores usados para alcançá-las (ou seja, um roteador pode enviar um anúncio BGP que não corresponda exatamente a itens em sua tabela de roteamento). Quais são as vantagens de cada método?

14.11 Considere uma função usada para converter entre as métricas de atraso e contagem de saltos. Você pode encontrar propriedades dessas funções que sejam suficientes para evitar loops de encaminhamento? As propriedades são necessárias também?

14.12 Existem circunstâncias sob as quais um protocolo SPF pode formar loops de endereço? (Dica: pense na entrega de melhor esforço).

14.13 Construa um programa aplicativo que envie uma requisição para um roteador executando RIP e exiba as rotas retornadas.

14.14 Leia cuidadosamente a especificação RIP. Podem rotas relatadas em uma resposta a uma consulta diferir das rotas relatadas por uma mensagem de atualização de roteamento? Se sim, como?

14.15 Leia a especificação OSPFv2 cuidadosamente. Como um gerente pode usar as facilidades do link virtual do OSPF?

14.16 O OSPF permite que gerentes atribuam muitos de seus próprios identificadores, possivelmente levando à duplicação de valores em múltiplos sites. Que identificador(es) pode(m) precisar mudar se dois sites executando o OSPF decidirem se fundir?

14.17 Você pode usar as mensagens de redirecionamento ICMP para passar informações de roteamento entre roteadores *internos*? Sim ou não? Por quê?

14.18 Leia as especificações para o OSPFv3. O que é uma *stub area* e o que é uma *not so stubby area* (NSSA)? Por que as duas são importantes?

14.19 O que faz o timeout para que o padrão OSPFv3 recomende um intervalo HELLO?

14.20 Escreva um programa que aceite como entrada uma descrição da internet da sua organização, use SNMP para obter tabelas de encaminhamento de todos os roteadores e relate quaisquer inconsistências.

14.21 Se sua organização executa softwares tais como o *gated* ou o *Zebra*, que gerenciam múltiplos protocolos de roteamento TCP/IP, obtenha uma cópia dos arquivos de configuração e explique o significado de cada item.

---

* O nome vem da convenção UNIX de anexar "d" aos nomes dos processos daemon; é pronunciado "route-d".

** Outros protocolos de roteamento definem uma conexão direta para ter zero salto; dizemos que RIP usa 1 contador de salto de origem.

* Observe que a contagem de saltos usada em RIP mede o vão da rede – a maior distância entre dois roteadores – e não o número total de redes ou de roteadores. A maioria das internets corporativas possui um pequeno vão que é muito menor do que 16.

* Para ajudar a evitar uma avalanche, o RIP exige que cada roteador espere um pequeno período de tempo aleatório antes de enviar uma atualização engatilhada.

* O Capítulo 15 descreve o Protocolo de Gerenciamento do Grupo Internet.

** O sufixo *ng* significa "nova geração"; o IPv6 foi inicialmente denominado *IPng*.

* Infelizmente, a terminologia tornou-se um tanto ambígua, porque o IPv6 usa o termo *link* no lugar de sub-rede IP (para permitir que vários prefixos IPv6 sejam atribuídos a uma determinada rede). Na maioria dos casos, o conceito IPv6 e conceito OSPFv3 se alinham, mas a distinção pode ser importante em casos especiais.

* O nome é pronunciado "I-S-I-S".

* O nome é uma abreviatura para *gateway daemon*, e é pronunciado "gate d."

CAPÍTULO 15

Multicasting de Internet

CONTEÚDOS DO CAPÍTULO

15.1 Introdução

Os capítulos anteriores definem os mecanismos que o IP usa para encaminhar e entregar datagramas unicast. Este capítulo explora outra característica do IP: a entrega multiponto de datagramas. Começamos com uma breve revisão do suporte de hardware subjacente. As seções posteriores descrevem o endereçamento IP para distribuição multiponto e os protocolos que os roteadores usam para propagar as informações de roteamento necessárias.

15.2 Broadcast de hardware

Muitas tecnologias de hardware contêm mecanismos para enviar pacotes a vários destinos simultaneamente (ou quase simultaneamente). No Capítulo 2, examinamos várias tecnologias e discutimos a forma mais comum de entrega multiponto: o *broadcast de hardware*. Entrega em broadcast significa que a rede entrega uma cópia de um pacote para cada destino. Os detalhes do broadcast de hardware variam. Em algumas tecnologias, o hardware envia uma única cópia de um pacote e organiza para que cada

computador conectado receba uma cópia. Em outras redes, o equipamento de rede implementa o broadcast encaminhando uma cópia independente de um pacote broadcast para cada computador individual.

A maioria das tecnologias de hardware fornece um endereço de destino especial, reservado, chamado de *endereço broadcast*. Para especificar um entrega broadcast, tudo que um emissor precisa fazer é criar um frame no qual o campo endereço de destino contenha o endereço broadcast. Por exemplo, a Ethernet usa o endereço de hardware tudo um como endereço broadcast; cada computador conectado a uma rede Ethernet aceita frames enviados para o endereço broadcast assim como pacotes enviados para o endereço MAC unicast do computador.

A principal desvantagem do broadcast de hardware surge de sua demanda por recursos – além de usar largura de banda de rede, cada broadcast consome recursos computacionais dos computadores conectados à rede. Em princípio, seria possível projetar software de internet que usasse broadcast para todos os datagramas enviados através de uma rede. Cada computador receberia uma cópia de cada datagrama, e o software IP no computador poderia examinar o endereço de destino e descartar datagramas endereçados a outras máquinas. Na prática, todavia, tal esquema é um contrassenso, pois cada computador vai gastar ciclos de CPU para descartar a maior parte dos datagramas que chegaram. Então, os desenvolvedores de TCP/IP desenvolveram mecanismos de vinculação de endereços que permitem que os datagramas sejam entregues via unicast.

15.3 Multicast de hardware

Algumas tecnologias de hardware aceitam uma segunda forma de entrega multiponto chamada multicast de hardware. Diferentemente do broadcast de hardware, o multicast de hardware permite que cada computador escolha se deseja participar do recebimento de um determinado multicast de hardware. Em geral, uma tecnologia de hardware reserva um grande conjunto de endereços para uso com multicast. Quando um grupo de aplicações deseja usar multicast de se comunicar, elas escolhem um determinado *endereço multicast* para usar para a comunicação. A aplicação em execução no computador deve solicitar ao sistema operacional que configura a placa de interface de rede para reconhecer o endereço multicast que foi selecionado. Após seu hardware ter sido configurado, o computador vai receber uma cópia de qualquer pacote enviado para o endereço multicast.

Nós usamos o termo *grupo multicast* para denotar o conjunto de computadores que está escutando um endereço multicast particular. Se aplicativos em seis computadores estão ouvindo um endereço multicast particular, o grupo multicast é dito ter seis *membros*. Em muitas tecnologias de hardware, um grupo multicast é definido apenas pelo conjunto de ouvintes – um computador arbitrário pode enviar um pacote para um determinado endereço multicast (ou seja, o remetente não precisa ser um membro do grupo multicast).

Em um nível conceitual, o endereçamento multicast pode ser visto como uma generalização de todas as outras formas. Por exemplo, podemos pensar em um *endereço multicast* convencional como um endereço multicast que exatamente um computador está escutando. De igual modo, podemos pensar em endereço broadcast como um endereço multicast que todos os computadores em uma determinada rede estão ouvindo. Outros endereços multicast podem corresponder a subconjuntos arbitrários de computadores na rede, possivelmente um conjunto vazio.

Apesar de sua aparente generalidade, o endereçamento multicast não vai substituir formas convencionais de endereçamento porque existe uma diferença fundamental no modo como os mecanismos de hardware subjacente implementam o encaminhamento e a distribuição de pacotes. Em vez de um único computador ou todos os computadores, um endereço multicast identifica um subconjunto arbitrário de computadores, e membros do grupo podem mudar a qualquer momento. Portanto, o hardware não pode determinar exatamente onde um determinado computador se conecta com a rede, e deve fazer flood de pacotes para todos os computadores e deixar que eles escolham se aceitam o pacote. O flooding é dispendioso, pois evita que os pacotes sejam transferidos em paralelo. Portanto, podemos ter a conclusão a seguir.

> *Embora seja interessante pensar no endereçamento multicast como uma generalização que inclui endereços unicast e broadcast, os mecanismos subjacentes de encaminhamento e distribuição podem tornar o multicast menos eficiente.*

15.4 Multicast Ethernet

A Ethernet fornece um exemplo de multicasting de hardware, e é

especialmente pertinente a multicast IP porque a Ethernet é amplamente difundida na Internet global. Reserva-se a para multicast a metade dos endereços Ethernet – o bit de ordem inferior do octeto de ordem superior distingue os endereços de unicast convencionais (*0*) dos endereços multicast (*1*). Na notação hexadecimal hifenizada,* o bit de multicast é dado por:

$$\mathbf{01\text{-}00\text{-}00\text{-}00\text{-}00\text{-}00_{16}}$$

Quando um quadro de interface Ethernet é inicializado, ele começa aceitando pacotes destinados a ambos, o endereço hardware de unicast ou endereço broadcast Ethernet. Entretanto, o software de driver de dispositivo pode reconfigurá-lo para permitir que também reconheça um ou mais endereços multicast. Por exemplo, suponha que o driver configura o endereço multicast Ethernet:

$$\mathbf{01\text{-}5E\text{-}00\text{-}00\text{-}00\text{-}01_{16}}$$

Após a configuração, o hardware de interface aceitará qualquer pacote enviado ao endereço de unicast MAC do computador, o endereço MAC de broadcast ou o exemplo de endereço MAC multicast (o hardware continuará a ignorar pacotes enviados a outros endereços multicast). As próximas seções explicam a semântica multicast do IP e como o IP utiliza o hardware de multicast básico.

15.5 A Construção conceitual dos blocos multicast da Internet

Três blocos de construção conceitual, descritos a seguir, são exigidos para um sistema multicast internet de propósito geral.

- Um esquema de endereçamento multicast.
- Um mecanismo eficaz de notificação e entrega.
- Um dispositivo de encaminhamento inter-redes eficiente.

Muitos detalhes e restrições apresentam desafios para um projeto geral. Por exemplo, além de fornecer endereços suficientes para muitos grupos, o *esquema de endereçamento* multicast deve acomodar dois objetivos conflitantes: permitir autonomia local na atribuição de endereços enquanto define endereços que têm significado global. Da mesma forma, os hosts

precisam de um *mecanismo de notificação* para informar aos roteadores sobre os grupos multicast em que estão participando, e roteadores precisam de um *mecanismo de entrega* para transferir pacotes multicast para hosts. Novamente, há duas possibilidades: o multicast internet deve fazer uso efetivo de multicast de hardware quando ele estiver disponível, mas também deve permitir a entrega através de redes que não têm suporte de hardware para multicast. Finalmente, uma *facilidade de encaminhamento* multicast apresenta o maior desafio de projeto dos três: a meta é um esquema que seja eficiente e dinâmico – ele deve transmitir pacotes multicast por caminhos mais curtos, não deve enviar uma cópia de um datagrama ao longo de um caminho e, se o caminho não leva a um membro do grupo, e deve permitir que a qualquer momento hosts se juntem aos grupos ou saiam deles.

Veremos que o mecanismo multicast de IP inclui todos os três blocos de construção. Ele define endereçamento multicast para IPv4 e IPv6, fornece um mecanismo que permite aos hosts entrar e sair de grupos de multicast IP, especifica como datagramas multicast são transferidos através de redes de hardware individual e fornece um conjunto de roteadores de protocolo que podem ser usados para trocar informações de roteamento de multicast e construir tabelas de encaminhamento para grupos multicast. A seção seguinte lista as propriedades do sistema multicast IP, e o restante do capítulo considera cada aspecto mais detalhadamente, começando com endereçamento.

15.6 O esquema do multicast IP

O multicasting IP é a abstração de internet do multicasting de hardware. Ele segue o modelo de permitir a transmissão a uma sub-rede de computadores host, mas generaliza o conceito de permitir que a sub-rede abranja redes físicas arbitrárias em toda uma internet. A ideia é que, sempre que possível, uma única cópia de um datagrama multicast que seja transmitido até um roteador deve encaminhar o datagrama ao longo de vários caminhos. Nesse ponto, uma cópia do datagrama é enviada para cada caminho. Assim, o objetivo é evitar a duplicação desnecessária.

Na terminologia do IP, um subconjunto de computadores escutando um determinado endereço multicast IP é conhecido como um *grupo multicast* IP. O multicasting IP está disponível tanto para IPv4 como para IPv6. A definição é inacreditavelmente ambiciosa, e possui as características gerais descritas a seguir.

- *Um endereço multicast IP por grupo*. A cada grupo multicast IP é atribuído um único endereço multicast IP. Alguns endereços multicast IP são permanentemente atribuídos pela autoridade da Internet e correspondem a grupos que sempre existem, mesmo se não tiverem membros atuais. Outros endereços são temporários e estão disponíveis para uso privado.
- *Número de grupos*. O IPv4 fornece endereços para até 2^{28} grupos multicast simultâneos. O IPv6 fornece muito mais. De modo prático, o limite em endereços não representa uma restrição no multicast IP. Ao contrário, restrições práticas no número de grupos multicast simultâneos surgem de restrições no tamanho da tabela de encaminhamento e do tráfego de rede necessário para propagar rotas quando os membros de grupos mudam.
- *Associação de grupo dinâmico*. Um host arbitrário pode se juntar ou deixar um grupo multicast IP a qualquer tempo. Além disso, um host pode ser um membro de um número arbitrário de grupos multicast simultaneamente.
- *Uso de hardware*. Se o hardware de rede subjacente aceitar multicasting, o IP utiliza o multicast de hardware para entregar um datagrama multicast IP na rede. Se o hardware não aceitar multicast, o IP usa broadcast ou unicast para entregar datagramas multicast IP.
- *Encaminhamento de internet*. Como os membros de um grupo multicast IP podem se conectar a várias redes físicas através de uma internet, *roteadores multicast* especiais são necessários para encaminhar multicast IP; na maior parte dos casos, em vez de usar roteadores separados, normalmente é acrescentada capacidade multicast a roteadores convencionais.
- *Semântica de distribuição*. O multicast IP utiliza a mesma semântica de distribuição de melhor esforço que outra distribuição de datagrama IP, significando que os datagramas multicast podem ser perdidos, retardados, duplicados ou entregues fora de ordem.
- *Associação e transmissão*. Um host arbitrário pode enviar datagramas a qualquer grupo multicast; a associação de grupo só é usada para determinar se o host recebe datagramas enviados ao grupo.

15.7 Endereços multicast IPv4 e IPv6

Dissemos que os endereços IP multicast dividem-se em dois tipos: os que são atribuídos de forma permanente e aqueles que estão disponíveis para uso temporário. Endereços permanentes são chamados de *bem conhecidos*; eles são definidos para os principais serviços na Internet global, bem como para a manutenção de infraestrutura (por exemplo, vimos que RIP e RIPng usam um endereço de multicast bem conhecido). Outros endereços multicast correspondem *a grupos multicast transitórios*, que são criados quando necessário e descartados assim que a contagem de membros do grupo chega a zero.

Como o multicasting de hardware, o multicasting IP usa o endereço de destino do datagrama para especificar que um datagrama particular deve ser entregue via multicast. O IPv4 reserva endereços de classe *D* para multicast: os primeiros *4* bits contêm *1110* e identificam o endereço como um endereço de multicast. No IPv6, um endereço de multicast tem os primeiros *8* bits definidos como *1*, conforme a Figura 15.1 ilustra.

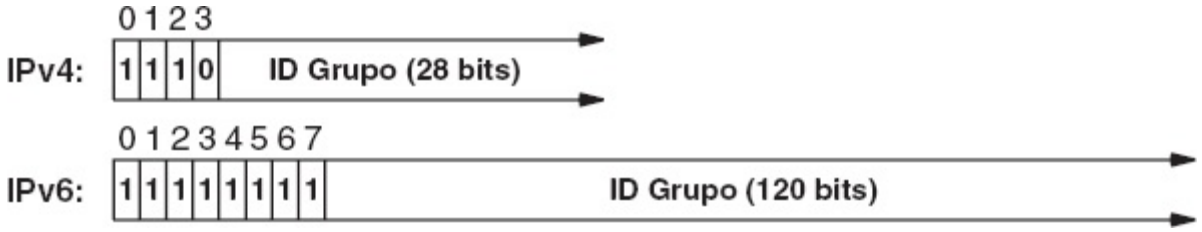


Figura 15.1 O formato dos endereços multicast IPv4 e IPv6. Um prefixo identifica o endereço como multicast.

Em cada caso, o restante de um endereço seguinte ao prefixo consiste de um identificador para um determinado grupo multicast. O IPv4 aloca 28 bits para ID de grupo multicast, o que significa 10^8 grupos são possíveis. O IPv6 aloca 120 bits para ID de grupo, o que dá 10^{36} grupos possíveis! O ID do grupo multicast *não* é particionado em bits que identificam a origem ou o proprietário do grupo.

15.7.1 IPv4 Multicast Address Space

Quando expressos em notação decimal hifenizada, os endereços multicast IPv4 variam de

224.0.0.0 até 239.255.255.255.

Muitas partes do espaço de endereço receberam significado especial. Por exemplo, o endereço inferior, 224.0.0.0, é reservado; ele não pode ser atribuído a grupo algum. Endereços até 224.0.0.255 são restritos a uma única rede (ou seja, um roteador é proibido de encaminhar um datagrama enviado a qualquer endereço nessa faixa e um emissor presume-se colocar o TTl em 1) e endereços de 239.0.0.0 até 239.255.255.255 são restritos a uma organização (ou seja, roteadores não devem encaminhá-los através de links externos). A Figura 15.2 mostra como o espaço do endereço multicast do IPv4 é dividido.

Faixa de endereço	Significado
224.0.0.0	**Endereço Base (Reservado)**
224.0.0.1 – 224.0.0.255	**Escopo restrito para uma rede**
224.0.1.0 – 238.255.255.255	**Escopo é global através da Internet**
239.0.0.0 – 239.255.255.255	**Escopo restrito para uma organiza**

Figura 15.2 A divisão do espaço de endereço multicast IPv4 de acordo com o escopo.

A Figura 15.3 lista alguns exemplos de atribuições de endereço multicast IPv4 específicos. Muitos outros endereços foram atribuídos, e alguns fornecedores escolheram endereços para usar em seus sistemas.

Endereço	Propósito atribuído
224.0.0.1	**Todos os sistemas nesta sub-rede**
224.0.0.2	**Todos os roteadores nesta sub-rede**
224.0.0.5	**Todos os roteadores OSPFIGP**
224.0.0.6	**Roteadores designados OSPFIGP**
224.0.0.9	**Roteadores RIP2**
224.0.0.12	**Servidor DHCP /agente de retransmissão**
224.0.0.22	**IGMP**

Figura 15.3 Exemplos de atribuições de endereço multicast IPv4. Todos os exemplos têm um escopo restrito a uma rede.

Na Figura 15.3, o endereço 224.0.0.l é permanentemente atribuído ao

grupo todos os sistemas, e o endereço 224.0.0.2 é permanentemente atribuído ao *grupo todos os roteadores*. O grupo *todos os sistemas* inclui todos os hosts e roteadores em uma rede que estão participando no multicast IP, enquanto o grupo *todos os roteadores* inclui apenas os roteadores que estão participando. Ambos os grupos são usados para protocolos de controle e precisam estar na mesma rede local do emissor; não existem endereços multicast IP que se refiram a todos os sistemas na Internet ou a todos os roteadores na Internet.

15.7.2 Espaço de endereço multicast IPv6

Como o IPv4, o IPv6 especifica o escopo associado com endereços multicast. Lembre que o primeiro octeto de um endereço multicast IPv6 contém tudo um. O IPv6 usa o segundo octeto do endereço para especificar o escopo. A Figura 15.4 lista as atribuições.

Segundo octeto	Significado
0x?0	**Reservado**
0x?1	**O escopo é restrito a um computador(loopback)**
0x?2	**O escopo é restrito a uma rede local**
0x?3	**O escopo é equivalente ao escopo local IPv4**
0x?4	**O escopo é administrativamente configurado**
0x?5	**O escopo é restrito a um único site**
0x?8	**O escopo é restrito a uma única organização**
0x?E	**O escopo o églobal através da Internet**

Figura 15.4 O uso do segundo octeto em um endereço para especificar o escopo de um endereço multicast IPv6.

Na figura, constantes começando com *0x* são hexadecimais. O ponto de interrogação indica um nibble arbitrário. Assim, 0x?1 se refere a 0x01, 0x11, 0x21… 0xF1.

Usar um octeto para especificar o escopo permite que um serviço seja acessado com uma variedade de escopos. Por exemplo, o *Protocolo de Tempo da Rede (Network Time Protocol – NTP)* foi atribuído ao grupo

multicast ID 0x101. O escopo da atribuição é irrestrito, significando que um emissor pode escolher o escopo de um multicast. Por exemplo, é possível enviar um datagrama multicast para todos os servidores NTP em um único link (endereço FF02::101) ou todos os servidores NTP em uma organização (endereço FF08::101). Somente o segundo octeto do endereço é diferente.

Alguns serviços são atribuídos a escopo específico ou a um conjunto de escopos específicos, porque o IETF não pode prever razão para enviar um multicast para o grupo globalmente. Por exemplo, o grupo multicast *Todos os Nós* é limitado – não se pode especificar um datagrama multicast para todos os nós através da Internet. A maioria dos protocolos de roteamento é tão limitada a um único link porque a comunicação pretendida é entre roteadores na mesma rede subjacente. A Figura 15.5 lista alguns exemplos de endereços multicast IPv6 permanentemente atribuídos.

Endereço	Propósito atribuído
FF02::1	**Todos os nós no segmento de rede local**
FF02::2	**Todos os roteadores no segmento de rede**
FF02::5	**Todos os roteadores SPF OSPFv3**
FF02::6	**Todos os roteadores DR OSPFv3**
FF02::9	**Roteadores RIP**
FF02::a	**Roteadores EIGRP**
FF02::d	**Roteadores PIM**
FF02::1:2	**Servidores DHCP e agentes de redistribuição na rede local**
FF05::1:3	**Servidores DHCP no site da rede local**
FF0x::FB	**DNS Multicast**
FF0x::101	**Protocolo de tempo da rede**
FF0x::108	**Serviço de Informação da rede**
FF0x::114	**Disponível para experimentos**

Figura 15.5 Exemplos de algumas atribuições de endereço multicast IPv6 permanentes usando conotação colon hex e abreviando zeros com dois-pontos. Alguns outros endereços têm significados específicos.

Assim como para o IPv4, fornecedores escolheram certos endereços

multicast IPv6 para usar com seus produtos. Embora nem todas as escolhas sejam oficialmente registradas com a autoridade da Internet que controla endereçamento, elas são geralmente respeitadas.

15.8 Semântica de endereço multicast

As regras que o IP segue quando encaminha um datagrama multicast diferem drasticamente daquelas usadas para datagramas de encaminhamento unicast. Por exemplo, um endereço multicast só pode ser usado como endereço de destino. Então, se um roteador encontra um endereço multicast no campo de endereço de origem de um datagrama ou em uma opção (por exemplo, rota de origem), o roteador descarta o datagrama. Além disso, nenhuma mensagem de erro ICMP pode ser gerada sobre datagramas multicast. A restrição se aplica ao eco ICMP (ou seja, requisição de ping) assim como a erros convencionais tais como destino inalcançável. Por isso, um ping enviado a um endereço multicast ficará sem resposta.

A regra proibindo erros ICMP é de certa maneira surpreendente, pois os roteadores IP respeitam o campo limite de salto no cabeçalho de um datagrama multicast. Normalmente, cada roteador decrementa a contagem e descarta o datagrama se esta atingir zero. A única distinção é que um roteador não envia uma mensagem ICMP para um datagrama multicast. Vamos ver que alguns protocolos multicast usam o limite de salto como um meio de limitar a propagação de datagrama.

15.9 Mapeando multicast IP para multicast Ethernet

Embora o padrão de multicast IP não cubra todos os tipos de hardware de rede, ele especifica como mapear um endereço multicast IP para um endereço multicast Ethernet. Para o IPv4, o mapeamento é eficiente e fácil de entender: IANA possui o endereço Ethernet prefixo *0x01005E.** Define-se um mapeamento da forma a seguir.

> *Para mapear um endereço multicast IPv4 para o endereço multicast Ethernet correspondente, coloque os 23 bits de ordem inferior do endereço multicast IPv4 nos 23 bits de ordem inferior do endereço multicast Ethernet especial 01-00- 5E-00-00-00*$_{16}$.

Por exemplo, o endereço multicast IPv4 224.0.0.2 se torna o endereço multicast Ethernet $01\text{-}00\text{-}5E\text{-}00\text{-}00\text{-}02_{16}$.

O IPv6 não usa o mesmo mapeamento que o IPv4. Na verdade, as duas versões nem compartilham o mesmo prefixo MAC. Em vez disso, o IPv6 usa o prefixo Ethernet 0x3333 e seleciona 32 bits do ID do grupo multicast do IP.

Para mapear um endereço multicast IPv6 para o endereço multicast Ethernet correspondente, coloque os 23 bits de ordem inferior do endereço multicast IPv6 nos 23 bits de ordem inferior do endereço multicast Ethernet especial $33\text{-}33\text{-}00\text{-}00\text{-}00\text{-}00_{16}$.

Por exemplo, o endereço multicast FF02:09:09:1949::DC:1 vai mapear para o endereço MAC Ethernet 33-33-00-DC-00-01.

Curiosamente, nem o mapeamento IPv4 nem o IPv6 são únicos. Como os endereços multicast IPv4 têm 28 bits significativos que identificam o grupo multicast, mais de um grupo multicast pode mapear dentro do mesmo endereço multicast de Ethernet ao mesmo tempo. De forma similar, alguns ID de grupo multicast IPv6 mapeiam o mesmo multicast Ethernet. Os desenvolvedores escolhem o esquema como um compromisso. De um lado, usar 23 (IPv4) ou 32 (IPv6) dos bits do ID de grupo para um endereço de hardware significa que a maior parte dos endereços multicast estão incluídos. O conjunto de endereços é grande o bastante para que as chances de dois grupos escolherem endereços com os bits de baixa ordem idênticos sejam pequenas. Por outro lado, organizar para o IP usar uma parte fixa do espaço do endereço multicast Ethernet torna a depuração muito fácil e elimina interferência entre protocolos Internet e outros protocolos que compartilham uma Ethernet. A consequência do projeto é que alguns datagramas multicast podem ser recebidos em um host que não é o host destinado a eles. Então, o software IP deve verificar os endereços cuidadosamente em todos os datagramas de entrada e descartar qualquer datagrama multicast indesejado.

15.10 Hosts e entrega multicast

Dissemos que o multicasting IP pode ser usado em uma única rede física ou através de uma internet. No primeiro caso, um host pode enviar diretamente a um host de destino simplesmente colocando o datagrama em

um frame e usando um endereço multicast de hardware que o receptor esteja escutando. No último caso, são necessários *roteadores multicast* especiais para encaminhar datagramas multicast através de múltiplas redes, para todos os hosts participantes em um grupo multicast. Então, se um host tem um datagrama com escopo diferente da rede local, deve enviá-lo para um roteador multicast. Surpreendentemente, um host não precisa instalar uma rota em um roteador multicast, nem o software IP usa uma rota-padrão para alcançar um roteador multicast. Em vez disso, a técnica que um host usa para encaminhar um datagrama multicast para um roteador é diferente do encaminhamento de datagramas para unicast e broadcast – o host meramente usa a capacidade de multicast do hardware de rede local para transmitir o datagrama. Roteadores multicast escutam *todas* as transmissões multicast IP; se um roteador multicast estiver presente na rede, ele receberá o datagrama e o encaminhará para outra rede se necessário.* Assim, a principal diferença entre multicast local e não local está nos roteadores multicast, não nos hosts.

15.11 Escopo multicast

O termo *escopo multicast* é usado para dois conceitos. Usamos esse termo para tornar claro o conjunto de hosts que estão escutando um dado grupo multicast, ou para especificar uma propriedade de um endereço multicast. No caso de especificar como os hosts estão localizados, usamos o termo para explicar se os membros atuais do grupo estão em uma rede, em múltiplas redes em um site, em múltiplas redes em múltiplos sites dentro de uma organização, em múltiplas redes dentro de um limite administrativamente definido ou redes arbitrárias na Internet global. No segundo caso, sabemos que as normas especificam quão longe um datagrama enviado para um endereço específico vai se propagar (ou seja, um conjunto de redes pelas quais um datagrama enviado ao endereço vai ser encaminhado). Informalmente, às vezes usamos o termo *faixa* em vez de escopo.

O IP usa duas técnicas para controlar o escopo multicast. A primeira se baseia no campo *limite de salto* do datagrama para controlar sua faixa. Definindo o limite de salto em um valor pequeno, um host pode limitar a distância a que o datagrama será encaminhado. Por exemplo, o padrão especifica que as mensagens de controle, que são usadas para comunicação entre um host e um roteador na mesma rede, precisam ter um limite de salto de 1. Como consequência, um roteador nunca encaminha qualquer

datagrama transportando informações de controle porque o limite de salto chegou a zero. Da mesma forma, se duas aplicações em execução em um único host quiserem usar multicast IP para comunicação interprocessos (por exemplo, para testar software), elas podem escolher um valor de TTL de 0 a fim de evitar que o datagrama deixe o host. É possível usar valores sucessivamente maiores do campo TTL para estender mais a noção de escopo. Por exemplo, alguns fornecedores de roteador sugerem configurar os roteadores em um lado para restringir datagramas multicast de deixar o site a menos que o datagrama tenha um TTL maior que 15. O ponto é: o limite de salto em um cabeçalho de datagrama fornece controle de grão grosso sobre o escopo do datagrama.

A segunda técnica, conhecida como *escopo administrativo*, consiste em escolher endereços multicast que têm escopo limitado. De acordo com o padrão, os roteadores na Internet são proibidos de encaminhar qualquer datagrama que tem um endereço escolhido do espaço restrito. Portanto, para evitar que a comunicação multicast entre membros de grupo atinja acidentalmente o exterior, uma organização pode atribuir ao grupo um endereço que tenha escopo local (ou seja, restrito ao site ou restrito a uma organização).

15.12 Participação do host em multicasting IP

Um host pode participar no multicast IP em um dos três níveis, como mostra a Figura 15.6.

Nível	Significado
0	O host não pode enviar nem receber multicast IP
1	O host pode enviar, mas não receber multicast IP
2	O host pode enviar e receber multicast IP

Figura 15.6 Os três níveis da participação de host no multicast IP.

Estender o software IP para permitir que um host envie multicast IP não é difícil; fazer com que o software de host receba datagramas multicast IP é mais complexo. Para enviar um datagrama multicast, uma aplicação precisa ser capaz de fornecer um endereço multicast como destino. Para receber multicast, uma aplicação precisa ser capaz de declarar que deseja se unir ou deixar um grupo multicast específico, e o software do protocolo precisa encaminhar uma cópia de um datagrama de entrada para cada aplicação

que se uniu ao grupo. Além disso, um datagrama não chega a um host automaticamente: as seções posteriores explicam que, quando se junta a um grupo multicast, o host precisa executar um protocolo especial para informar um roteador multicast local do seu estado de associação de grupo. Uma grande parte da complexidade vem de uma decisão de projeto do multicast IP.

Os hosts se unem a grupos multicast IP específicos em redes específicas.

Em outras palavras, um host com múltiplas conexões de rede pode se unir a um determinado grupo multicast em uma rede, e não em outra. Para entender a razão para manter a associação de grupo relacionada com as redes, lembre-se de que é possível usar multicast IP entre conjuntos locais de máquinas. O host pode querer usar uma aplicação multicast para interagir com máquinas em uma rede, mas não com máquinas em outra.

Como a associação de grupo está relacionada com redes específicas, o software precisa manter listas separadas de endereços multicast para cada rede à qual o host se conecta. Além disso, um programa aplicativo também precisa especificar uma rede em particular quando pede para se unir ou deixar um grupo multicast. É claro que a maior parte dos aplicativos não sabe (ou não se importa) com as redes às quais um host se conecta, o que significa que eles não sabem qual rede especificar quando precisam se juntar a um grupo multicast.

15.13 Protocolo de gerenciamento de um grupo Internet IPv4 (IGMP)

Dissemos que, para enviar ou receber multicast em IPv4 através de uma única rede local, um host só precisa ter um software que o permita usar a rede subjacente para transmitir e receber datagramas multicast IP. Entretanto, para participar de um multicast que abranja múltiplas redes, o host precisa informar pelo menos um roteador multicast local. O roteador local contata outros roteadores multicast, passando as informações de associação e estabelecendo rotas. Veremos, mais tarde, que o conceito é semelhante à propagação de rota convencional entre roteadores na Internet.

Roteadores multicast não atuam enquanto pelo menos um host em uma dada rede se junte ao grupo multicast. Quando ele decide se juntar a um grupo multicast, um host informa a um roteador multicast local. Um host

IPv4 usa o *Protocolo de Gerenciamento de Grupo* (*Internet Group Management Protocol – IGMP*). Como a versão atual é a 3, o protocolo descrito aqui é oficialmente conhecido como *IGMPv3*.

O IGMP é um padrão para IPv4 exigido em todas as máquinas que recebem multicast IPv4 (ou seja, todos os hosts e roteadores que participam no nível 2). O IGMP usa datagramas IP para carregar mensagens. Além disso, consideramos o IGMP como um serviço integrado com IP, análogo ao ICMP. Por isso, não devemos pensar em IGMP como um protocolo usado por aplicativos arbitrários.

> *Embora ele use datagramas IP para carregar mensagens, pensamos no IGMP como parte integrante do IPv4, não como um protocolo independente.*

Conceitualmente, o IGMP possui duas fases. Fase 1: quando se une a um novo grupo multicast, um host envia uma mensagem IGMP para o endereço multicast do grupo declarando sua associação. Os roteadores multicast locais recebem a mensagem e estabelecem o roteamento necessário propagando as informações de associação de grupo para outros roteadores multicast através da Internet. Fase 2: como a associação é dinâmica, os roteadores multicast locais pesquisam periodicamente os hosts na rede local para determinar se algum host ainda permanece membro do grupo. Se qualquer host responder por um determinado grupo, o roteador mantém o grupo ativo. Se nenhum host relatar associação em um grupo após várias pesquisas, o roteador multicast considera que nenhum dos hosts na rede permanece no grupo e para de anunciar associação de grupo para outros roteadores multicast.

Para complicar ainda mais a associação de grupo, o IGMP permite que uma aplicação em um host instale um *filtro de endereço de origem* (*source address filter*), que especifica se o host deve incluir ou excluir tráfego multicast de um determinado endereço de origem. Assim, é possível se unir a um grupo multicast, mas excluir datagramas enviados ao grupo por uma determinada origem. A presença de filtros é importante porque o IGMP permite que um host passe o conjunto de especificações de filtro para o roteador local juntamente com as informações de associação de grupo. No caso em que duas aplicações discordam (ou seja, uma aplicação exclui uma

determinada origem e outra inclui a origem), o software no host precisa racionalizar as duas especificações e, depois, tratar a decisão sobre que aplicações recebem um determinado datagrama localmente.

15.14 Detalhes do IGMP

O IGMP é cuidadosamente projetado para evitar sobrecarga que possa congestionar as redes. Em especial, como uma determinada rede pode incluir múltiplos roteadores multicast, bem como hosts que participam no multicasting, o IGMP precisa evitar ter participantes gerando tráfego de controle desnecessário. Existem várias maneiras de o IGMP minimizar seu efeito sobre a rede.

- Toda a comunicação entre os hosts e os roteadores multicast utiliza multicast IP. Ou seja, quando envia mensagens, o IGMP sempre usa multicast IPv4. Desse modo, datagramas carregando mensagens IGM são transmitidos usando multicast de hardware, se disponível, significando que um host que não é participante no multicast IP nunca recebe mensagens IGMP.
- Ao pesquisar para determinar a associação de grupo, um roteador multicast envia uma única consulta a fim de requisitar informações sobre todos os grupos em vez de enviar uma mensagem separada para cada um. A frequência de pesquisa-padrão é de 125 segundos, o que significa que o IGMP não gera tanto tráfego.
- Se vários roteadores multicast estiverem conectados à mesma rede, eles escolhem de maneira rápida e eficiente um único roteador para pesquisar a associação de host, o que significa que o tráfego IGMP em uma rede não aumenta à medida que roteadores multicast adicionais são conectados a ela.
- Os hosts não respondem imediatamente a uma consulta IGMP de um roteador. Em vez disso, cada consulta contém um valor, *N*, que especifica um tempo de resposta máximo (o padrão é 10 segundos). Quando uma consulta chega, um host escolhe um atraso aleatório entre *O* e *N* que ele espera antes de enviar uma resposta. Se um determinado host for um membro de vários grupos, ele escolherá um número aleatório diferente para cada um. Assim, a resposta de um host à consulta de um roteador será espaçada em *10* segundos.
- Se um host for membro de múltiplos grupos multicast, o host poderá enviar um relatório para múltiplas associações de grupo em um único

pacote para minimizar tráfego.

Embora essa atenção cuidadosa para detalhe possa parecer desnecessária, a natureza dinâmica do multicast IP significa que a troca de mensagem em uma dada rede depende das aplicações. Então, de forma diferente dos protocolos de roteamento onde o tráfego depende do protocolo, o tráfego IGMP depende do número de grupos multicast a que as aplicações estão ouvindo.

15.15 Transições de estado de associação de grupo IGMP

Em um host, o IGMP precisa se lembrar do estado de cada grupo multicast ao qual o host pertence, juntamente com os filtros de origem associados a cada grupo.* Pensamos em um host como mantendo uma tabela na qual ele registra informações de associação de grupo. Inicialmente, todas as entradas na tabela estão vagas. Sempre que um programa aplicativo no host se une a um novo grupo, o software IGMP aloca uma entrada e preenche informações sobre o grupo, incluindo filtros de endereço que o aplicativo tenha especificado. Quando uma aplicação deixa um grupo, a entrada correspondente é removida da tabela. Ao criar um relatório, o software consulta a tabela, racionaliza todos os filtros para um grupo e monta um relatório único.

As ações tomadas pelo software IGMP em resposta a vários eventos podem ser mais bem explicadas pelo diagrama de transição de estado na Figura 15.7.

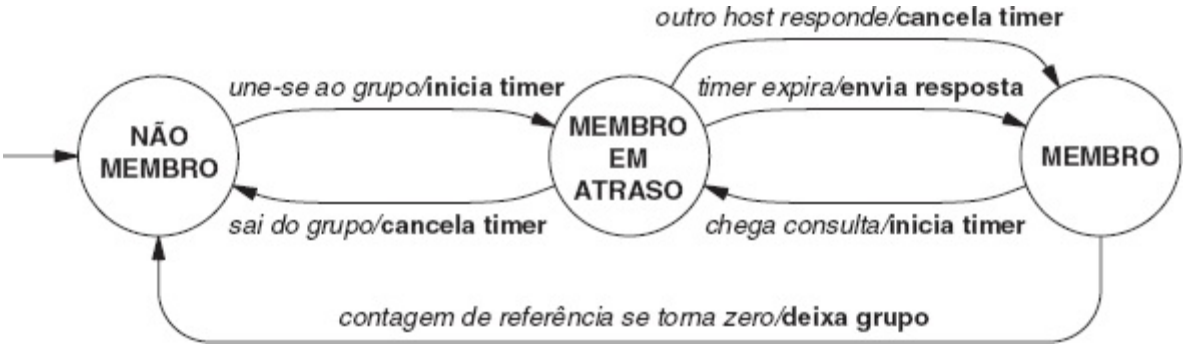


Figura 15.7 Os três estados possíveis de uma entrada na tabela do grupo multicast de um host e as transições entre eles, em que cada transição é rotulada com um evento e uma ação. As transições de estado não mostram mensagens enviadas ao se unirem a um grupo e ao deixá-lo.

Um host mantém uma entrada de tabela independente para cada grupo

do qual ele é atualmente um membro. Como a figura mostra, quando um host se une ao grupo ou quando uma consulta chega de um roteador multicast, o host move a entrada para o estado MEMBRO EM ATRASO e escolhe um atraso aleatório. Se outro host no grupo responde à consulta do roteador antes que o timer expire, o host cancela seu timer e muda para o estado MEMBRO. Se o timer expirar, o host enviará uma mensagem de resposta antes de mudar para o estado MEMBRO. Como um roteador apenas gera uma consulta a cada 125 segundos, espera-se que um host permaneça no estado MEMBRO na maior parte do tempo.

O diagrama na Figura 15.7 omite alguns detalhes. Por exemplo, se uma consulta chega enquanto o host está no estado MEMBRO EM ATRASO, o protocolo exige que o host resete seu timer.

15.16 Formato de mensagem membership query IGMP

O IGMPv3 define dois tipos de mensagem: uma mensagem *consulta de associação* (*membership query*), que um roteador envia para examinar membros do grupo, e uma *mensagem relatório de associação* (*membership report*), que um host gera para relatar os grupos que as aplicações no host estão usando atualmente. A Figura 15.8 ilustra o formato de mensagem consulta de associação (membership query).

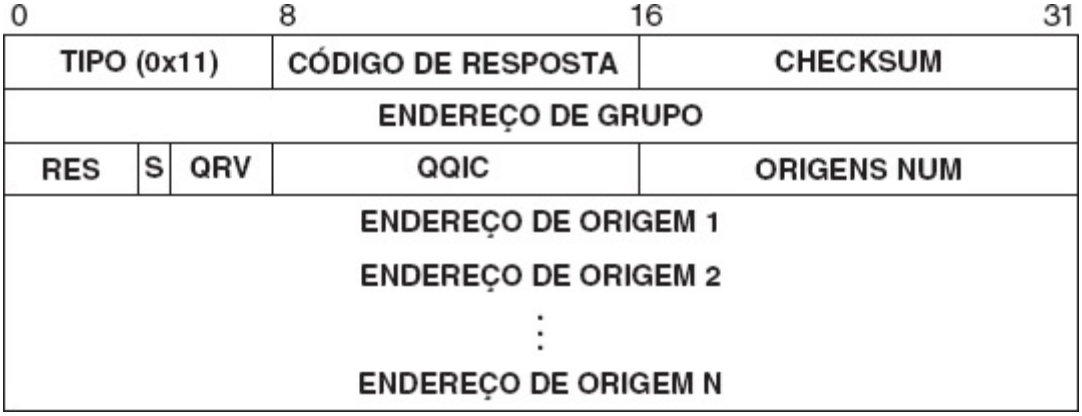


Figura 15.8 Formato de uma mensagem *membership query* IGMP.

Como a figura mostra, uma mensagem membership query começa com um cabeçalho de tamanho fixo de doze octetos. O campo TIPO identifica o tipo de mensagem, com os tipos para várias versões de IGMP listados na Figura 15.9.

Tipo	Versão do Protocolo	Significado
0x11	**3**	**Consulta de associação**
0x22	**3**	**Relatório de associação**
0x12	**1**	**Relatório de associação**
0x16	**2**	**Relatório de associação**
0x17	**2**	**Deixar grupo**

Figura 15.9 Tipos de mensagem IGMP. Para fornecer retrocompatibilidade, a versão 3 do protocolo inclui tipos de mensagem 1 e 2.

Quando um roteador pesquisa por associações de grupo, o campo denominado CÓDIGO RESP especifica um intervalo máximo para o atraso randômico que os membros do grupo calculam. Se o campo começa com um bit 0, o valor é tomado como um inteiro medido em décimos de segundo; se o campo começa com um 1, o valor é um número de ponto flutuante com três bits de expoente e quatro bits de mantissa. Cada host no grupo atrasa um tempo randômico entre 0 e 10. O IGMP permite que roteadores definam um valor máximo em cada mensagem de consulta para dar aos gerentes controle sobre o tráfego IGMP. Se uma rede contém muitos hosts, um valor de atraso maior espalha ainda mais os tempos de resposta e, dessa forma, diminui a probabilidade de ter mais do que um host respondendo à consulta. O campo CHECKSUM contém um checksum para a mensagem (IGMP checksums são computados na mensagem IGMP somente, e usam o mesmo algoritmo IP complemento de um de 16 bits). O campo ENDEREÇO DE GRUPO (GROUP ADDRESS) é usado para especificar um grupo particular ou contém zero para uma pesquisa geral. Ou seja, quando ele envia uma pesquisa a um grupo específico ou uma combinação de grupo específico e origem, um roteador preenche o campo ENDEREÇO DE GRUPO. O campo *S* indica se o roteador deve suprimir a atualização de timer normal que é executada quando chega uma atualização; o bit não se aplica aos hosts. O campo *QVR* controla a robustez por permitir ao IGMP enviar um pacote múltiplas vezes em uma rede com perdas. O valor-padrão é 2; um emissor transmite a mensagem *QRV-1* vezes. O campo *QQIC* usa a mesma representação que o campo CÓDIGO RESP.

A última parte de uma mensagem query IGMP consiste de zero ou mais

origens; o campo ORIGENS NUM especifica o número de entradas que seguem. Cada ENDEREÇO DE ORIGEM consiste de um endereço IP de 32 bits. O número de origem é zero em uma *consulta geral* (ou seja, uma requisição por parte de um roteador por informações sobre todos os grupos multicast em uso na rede) e em uma *consulta específica* do grupo (ou seja, uma requisição por parte de um roteador por informações sobre um grupo multicast especificado). Para uma *consulta específica do grupo e da origem*, a mensagem contém uma lista de uma ou mais origens; um roteador usa essa mensagem para requisitar estado de recepção para uma combinação do grupo multicast e quaisquer das origens especificadas.

15.17 Formato de mensagem membership report IGMP

O segundo tipo de mensagem usado com o IGMPv3 é um *relatório de associação* (*membership report*) que os hosts utilizam para passar estado de participação para o roteador. A Figura 15.10 ilustra o formato. Como a figura mostra, uma mensagem membership report consiste em um cabeçalho de 8 octetos que especifica o tipo de mensagem e uma contagem de inteiro dos registros de grupo, *K*, seguida de *K registros do grupo*. A Figura 15.11 ilustra o formato de cada registro de grupo. O formato é simples. O campo inicial, rotulado como TIPO REGISTRO (REC TYPE), permite que o emissor especifique se a lista de origens no registro corresponde a um *filtro inclusivo* (*inclusive filter*), um *filtro exclusivo* ou uma mudança em um relatório anterior (por exemplo, uma origem adicional a ser incluída ou excluída). O campo rotulado como ENDEREÇO MULTICAST especifica o endereço multicast ao qual o registro de grupo se refere, e o campo rotulado como NUM ORIGENS especifica o número de endereços de origem contidos no registro de grupo.

É importante notar que o IGMP não fornece todas as mensagens ou recursos possíveis. Por exemplo, o IGMP não inclui um mecanismo que permita a um host descobrir o endereço IP de um grupo – o software de aplicação precisa saber o endereço do grupo para que possa usar o IGMP para se unir ao grupo. Portanto, algumas aplicações usam endereços de grupo permanentemente atribuídos, algumas permitem que um gerente configure o endereço quando o software é instalado e outras obtêm o endereço dinamicamente (por exemplo, de um servidor). De modo semelhante, o IGMPv3 não fornece mensagens explícitas que um host pode

emitir para deixar um grupo ou escutar toda a comunicação em um grupo. Em vez disso, para deixar um grupo, um host envia uma mensagem membership report que especifica um filtro inclusivo com uma lista de endereços de origem IP vazia. Para escutar todas as origens, um host envia uma mensagem membership report que especifica um filtro exclusivo com uma lista de endereços de origem IP vazia.

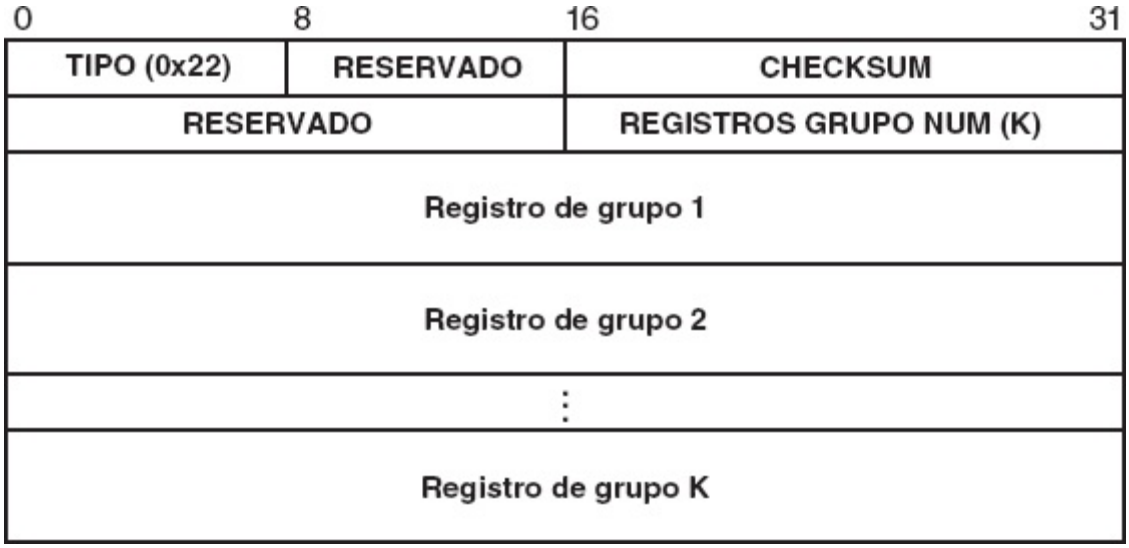


Figura 15.10 O formato de uma mensagem *membership report* IGMPv3.

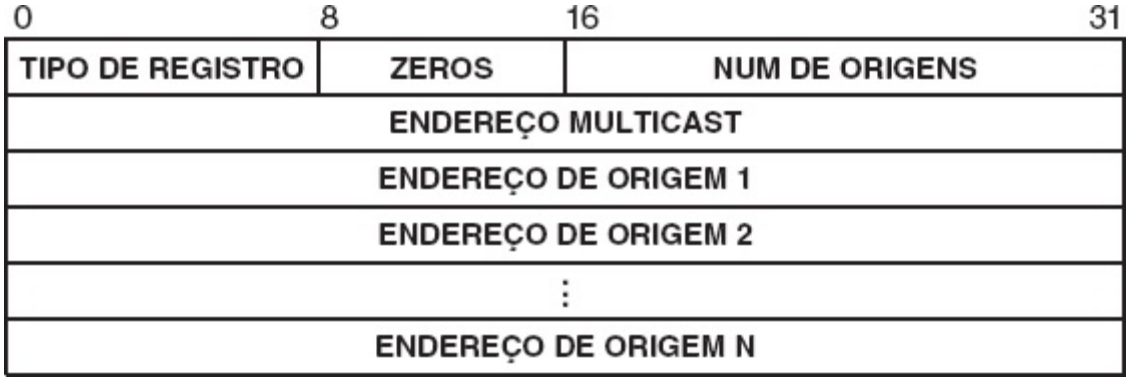


Figura 15.11 O formato de cada *registro de grupo* dentro de uma mensagem *membership report* IGMPv3.

15.18 IPv6 multicast group membership com MLDv2

O IPv6 não usa o IGMP; em vez disso, ele define um *Protocolo Multicast Listener Discovery*. A versão atual é a 2, e o protocolo é abreviado MLDv2. Apesar das mudanças, o IPv4 e o IPv6 usam essencialmente a mesma abordagem. Na realidade, o padrão MLDv2 determina que o MLDv2 é

simplesmente uma translação do IGMP para usar semântica IPv6.

Um host usa MLDv2 para informar roteadores multicast sobre a rede local da(s) associação(ões) de grupo de host. Como no IGMP, uma vez que um host anuncia associações, um roteador na rede usa MLDv2 para pesquisar o host periodicamente para determinar se ele continua membro do(s) grupo(s). Um conjunto de roteadores multicast em uma dada rede coopera para encontrar um *roteador de consulta* que vai enviar consultas periódicas; se o roteador de consulta atual falha, outro roteador multicast na rede toma para si a responsabilidade.

O MLDv2 define três tipos de mensagens de consulta que os roteadores enviam: *consultas gerais, consultas específicas de endereço multicast e consultas específicas de endereços e origens multicast.* Como com o IGMP, um roteador multicast típico envia consultas gerais que pedem que os hosts respondam especificando grupos que eles estão escutando. A Figura 15.12 ilustra o formato de uma mensagem de consulta MLDv2.

Figura 15.12 Formato de uma mensagem consulta MLDv2. Um roteador

multicast envia essas mensagens para determinar se hosts em uma rede estão participando em multicast.

Como mencionado antes, o campo QQIC especifica um intervalo de consulta. Em uma consulta geral, não há endereço multicast específico, então o campo é definido em zero, e por isso não há origens, o campo ORIGENS DE NÚMERO contém zero e a mensagem contém campos sem ENDEREÇO ORIGEM.

Quando uma consulta chega, um host segue as mesmas etapas que no IGMP: o host atrasa um tempo randômico e retorna enviando uma resposta que especifica o endereço multicast a quem ele ainda está escutando. A resposta consiste de uma mensagem *relatório de escuta multicast* (*Multicast Listener Report*). A Figura 15.13 mostra o formato geral.

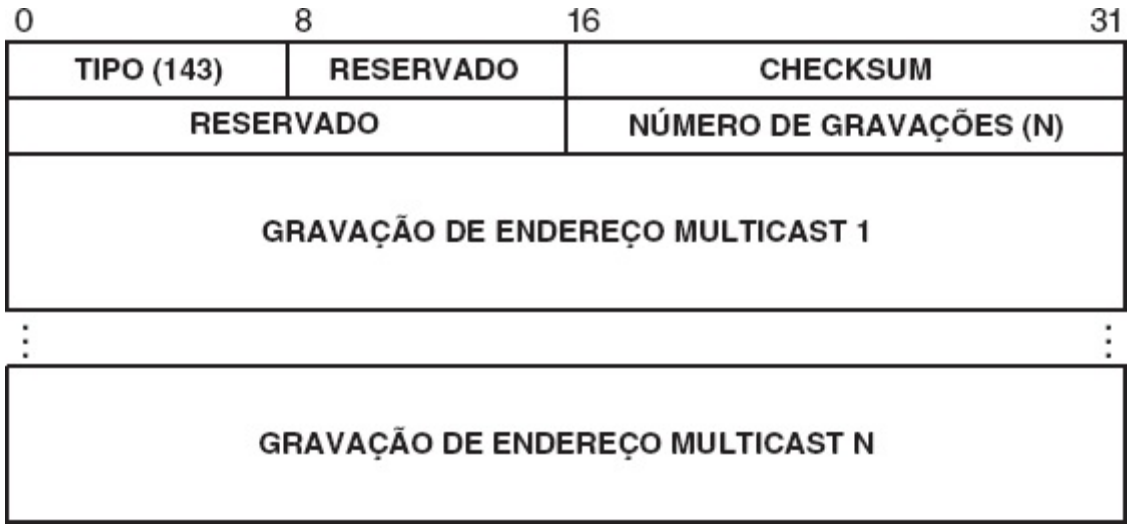


Figura 15.13 Forma geral de uma mensagem relatório de escuta multicast.

Em vez de simplesmente escutar endereços multicast que o host está escutando, o relatório de escuta multicast especifica uma lista de gravações de endereço multicast. Cada gravação na lista especifica o endereço multicast de um grupo seguido por uma série de endereços unicast nos quais o host está escutando. Pode parecer estranho que um host especifique mais de um endereço unicast, pois a maioria dos hosts são singly-homed. Então, vamos esperar um host ter somente um endereço unicast. Entretanto, o IPv6 permite que um host tenha endereços múltiplos em uma única rede. Por isso, o relatório de escuta admite vários endereços por entrada.

15.19 Informações de encaminhamento e roteamento multicast

Embora o esquema de endereçamento multicast IP permita que um host envie e receba multicast local e IGMP ou MLDv2 permitam que um roteador mantenha dados de hosts na rede local que estão escutando a grupos multicast, não especificamos como roteadores multicast trocam informação de associação de grupo ou como os roteadores garantem que uma cópia de cada datagrama multicast alcance todos os membros do grupo.

É interessante que vários protocolos se propuseram a permitir que roteadores trocassem informação de roteamento multicast. Porém, não surgiu um único padrão como o líder. Na verdade, embora muitos esforços tenham sido despendidos, não há acordo sobre um projeto global – os protocolos existentes diferem em seus objetivos e abordagens básicas. Consequentemente, o multicasting não é usado de forma ampla na Internet global.

Por que o roteamento multicast é tão difícil? Por que não estender os esquemas de roteamento convencionais para manipular multicast? A resposta é que os protocolos de roteamento multicast diferem dos protocolos de roteamento convencionais em aspectos fundamentais, pois o encaminhamento multicast difere do encaminhamento convencional. Para verificar algumas das diferenças, considere o encaminhamento multicast através da arquitetura descrita na Figura 15.14

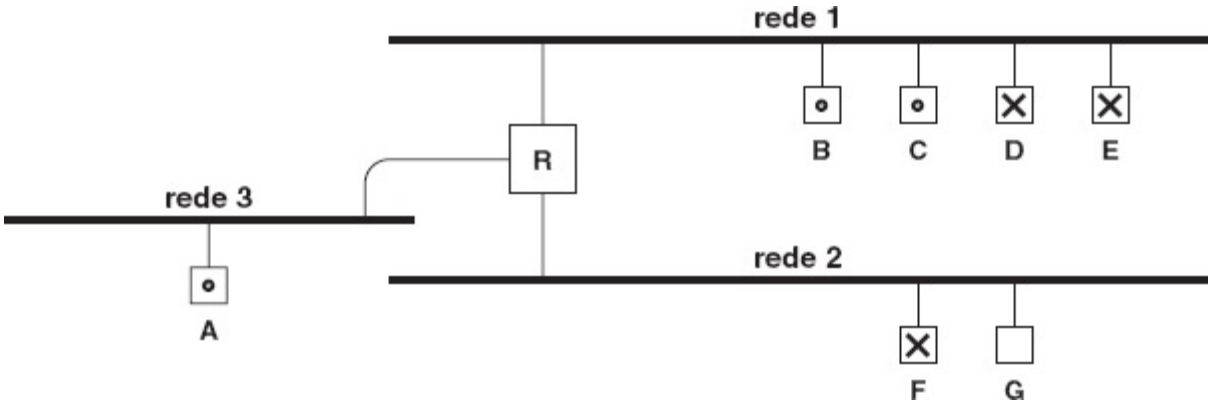


Figura 15.14 Uma internet simples que ilustra o encaminhamento multicast com três redes conectadas por um roteador. Os hosts marcados com um pequeno círculo participam de um grupo multicast, enquanto os marcados com um "X" participam de outro.

15.19.1 Necessidade de encaminhamento dinâmico

Mesmo para a topologia simples mostrada na Figura 15.14, o encaminhamento multicast difere do encaminhamento unicast. Por exemplo, a figura mostra dois grupos multicast: o grupo indicado por um círculo possui os membros *A*, *B* e *C*, e o grupo indicado por um *X* possui os membros *D*, *E* e F. O grupo com círculo não possui membros na rede *2*. Para evitar transmissões desnecessárias, o roteador *R* nunca deve enviar pacotes destinados ao grupo com círculo através da rede *2*. Entretanto, um host pode se unir a qualquer grupo em qualquer momento – se o host for o primeiro em sua rede a se unir ao grupo, o encaminhamento multicast precisa ser alterado para incluir a rede. Portanto, encontramos uma importante diferença entre propagação de rota convencional e propagação de rota multicast.

Diferentemente do encaminhamento unicast em que as rotas mudam apenas quando a topologia muda ou o equipamento falha, as rotas multicast podem mudar simplesmente porque um programa aplicativo se une ou deixa um grupo multicast.

15.19.2 Insuficiência de encaminhamento de destino

O exemplo da Figura 15.14 ilustra outro aspecto do encaminhamento multicast. Se o host *F* e o host *E* enviarem, cada um, um datagrama para o grupo com *X*, o roteador *R* os receberá e os encaminhará. Como os dois datagramas estão direcionados ao mesmo grupo, eles têm o mesmo endereço de destino. Entretanto, as ações de encaminhamento corretas diferem: *R* envia o datagrama de *E* para a rede *2* e envia o datagrama de *F* para a rede *1*. Curiosamente, ao receber um datagrama destinado ao grupo com *X* enviado pelo host *A*, o roteador usa uma terceira ação: ele encaminha duas cópias, uma para a rede *1* e a outra para a rede *2*. Assim, a seguir, observamos a segunda grande diferença entre encaminhamento convencional e encaminhamento multicast.

Diferentemente do encaminhamento unicast, o encaminhamento multicast exige que um roteador examine mais do que o endereço de

destino.

15.19.3 Emissores arbitrários

O último recurso do encaminhamento multicast ilustrado pela Figura 15.14 surge porque o IP permite que um host arbitrário, que não seja necessariamente um membro do grupo, envie um datagrama para o grupo. Na figura, o host *G* pode enviar um datagrama para o grupo com círculo, mesmo que *G* não seja membro de qualquer grupo e não existam membros do grupo com círculo na rede de *G*. Mais importante, enquanto viaja através da rede, o datagrama pode atravessar outras redes que não possuem membros conectados. Portanto, podemos fazer o resumo a seguir.

Um datagrama multicast pode se originar em um computador que não seja parte do grupo multicast e pode ser encaminhado através de redes que não tenham qualquer membro de grupo conectado.

15.20 Paradigmas básicos de encaminhamento multicast

Sabemos, pelo exemplo anterior, que os roteadores multicast precisam usar mais de um endereço de destino quando processam um datagrama. Exatamente que informações um roteador multicast usa ao decidir como encaminhar um datagrama? A resposta reside em entender que, como um destino de multicast representa um conjunto de computadores, um sistema de encaminhamento ótimo alcançará todos os membros do conjunto sem enviar um datagrama através de uma determinada rede duas vezes. Embora um único roteador multicast como o da Figura 15.14 possa simplesmente evitar o envio de um datagrama de volta pela interface em que chega, usar unicamente a interface não evitará que um datagrama seja encaminhado entre um conjunto de roteadores que são arranjados em um ciclo. Para evitar esses loops de encaminhamento, os roteadores multicast se baseiam no endereço de origem do datagrama.

Uma das primeiras ideias que surgiram para o encaminhamento multicast foi uma forma do broadcasting descrito anteriormente. Conhecido como *Reverse Path Forwarding* (*RPF*),* o esquema usa o endereço de origem do datagrama para impedir que este viaje em torno de um loop

repetidamente. Para usar o RPF, um roteador multicast precisa ter uma tabela de roteamento convencional com caminhos mais curtos a todos os destinos. Quando um datagrama chega, o roteador extrai o endereço de origem, consulta-o na tabela de roteamento local e encontra *I*, a interface que leva à origem. Se o datagrama tiver chegado pela interface *I*, o roteador encaminha uma cópia a cada uma das outras interfaces; caso contrário, o roteador descarta a cópia.

Como garante que uma cópia de cada datagrama multicast seja enviada através de cada rede na Internet, o esquema RPF básico assegura que cada host em um grupo multicast receberá uma cópia de cada datagrama enviado ao grupo. Entretanto, o RPF sozinho não é usado para encaminhamento multicast porque desperdiça largura de banda transmitindo datagramas multicast por redes que nem possuem membros de grupo nem levam a membros de grupo.

Para evitar a propagação de datagramas multicast onde não são necessários, foi criada uma forma modificada de RPF. Conhecido como *Encaminhamento por Caminho Reverso Truncado* (*Truncated Reverse Path Forwarding – TRPF –* ou *Truncated Reverse Path Broadcasting – TRPB*), o esquema segue o algoritmo do RPF, mas restringe mais a propagação evitando caminhos que não levam a membros de grupo.

Podemos fazer o resumo a seguir.

> *Ao tomar uma decisão de encaminhamento, um roteador multicast usa os endereços de origem e de destino do datagrama. O mecanismo básico é conhecido como Truncated Reverse Path Forwarding (TRPF).*

15.21 Consequências do TRPF

Embora o TRPF garanta que cada membro de um grupo multicast receba uma cópia de cada datagrama enviado ao grupo, ele apresenta consequências surpreendentes. Primeiro, por se basear no RPF para evitar loops, o TRPF distribui uma cópia extra dos datagramas para algumas redes, exatamente como o RPF convencional. A Figura 15.15 ilustra como as duplicatas surgem.

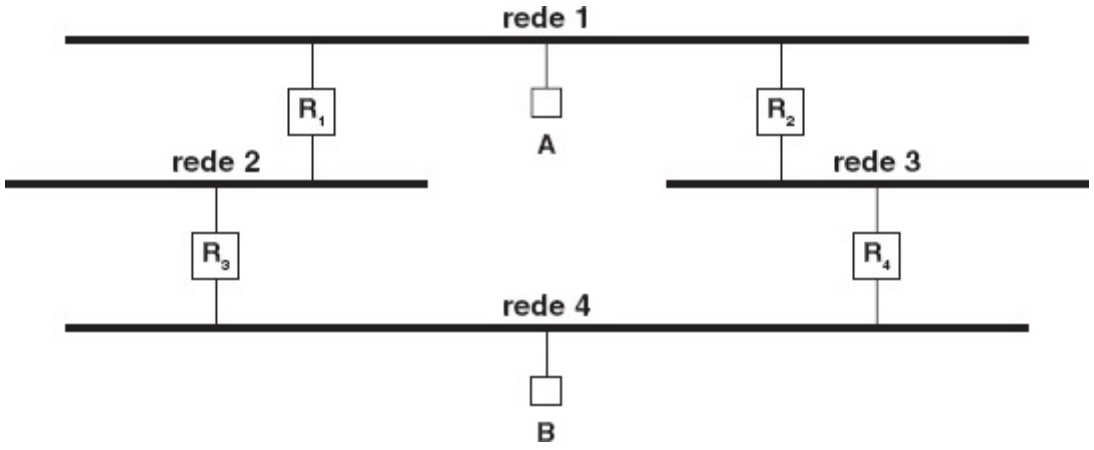


Figura 15.15 Uma topologia que faz um esquema RPF distribuir várias cópias de um datagrama para alguns destinos.

Na figura, quando o host A envia um datagrama, os roteadores R_1 e R_2 recebem uma cópia cada um. Como o datagrama chega através da interface que se encontra ao longo do caminho mais curto para A, R_1 encaminha uma cópia para a rede 2 e R_2 encaminha uma cópia para a rede 3. Ao receber uma cópia de rede 2 (o caminho mais curto para A), R_3 encaminha a cópia de rede 4. Infelizmente, R_4 também envia uma cópia para a rede 4. Assim, embora RPF permita a R_3 e R_4 evitar um loop, descartando a cópia que chega pela rede 4, o host B recebe duas cópias do datagrama.

Uma segunda consequência surpreendente surge porque o TRPF usa endereços de origem e destino quando encaminha datagramas: a distribuição depende da origem de um datagrama. Por exemplo, a Figura 15.16 mostra como os roteadores multicast encaminham datagramas de duas origens diferentes através de uma topologia fixa.

Como mostra a figura, a origem afeta o caminho que um datagrama segue para alcançar uma determinada rede e também os detalhes da entrega. Por exemplo, na parte (a) da Figura 15.16, uma transmissão pelo host X faz com que o TRPF entregue duas cópias do datagrama para a rede 5. Na parte (b), apenas uma cópia de uma transmissão pelo host Z alcança a rede 5, mas duas cópias alcançam as redes 2 e 4.

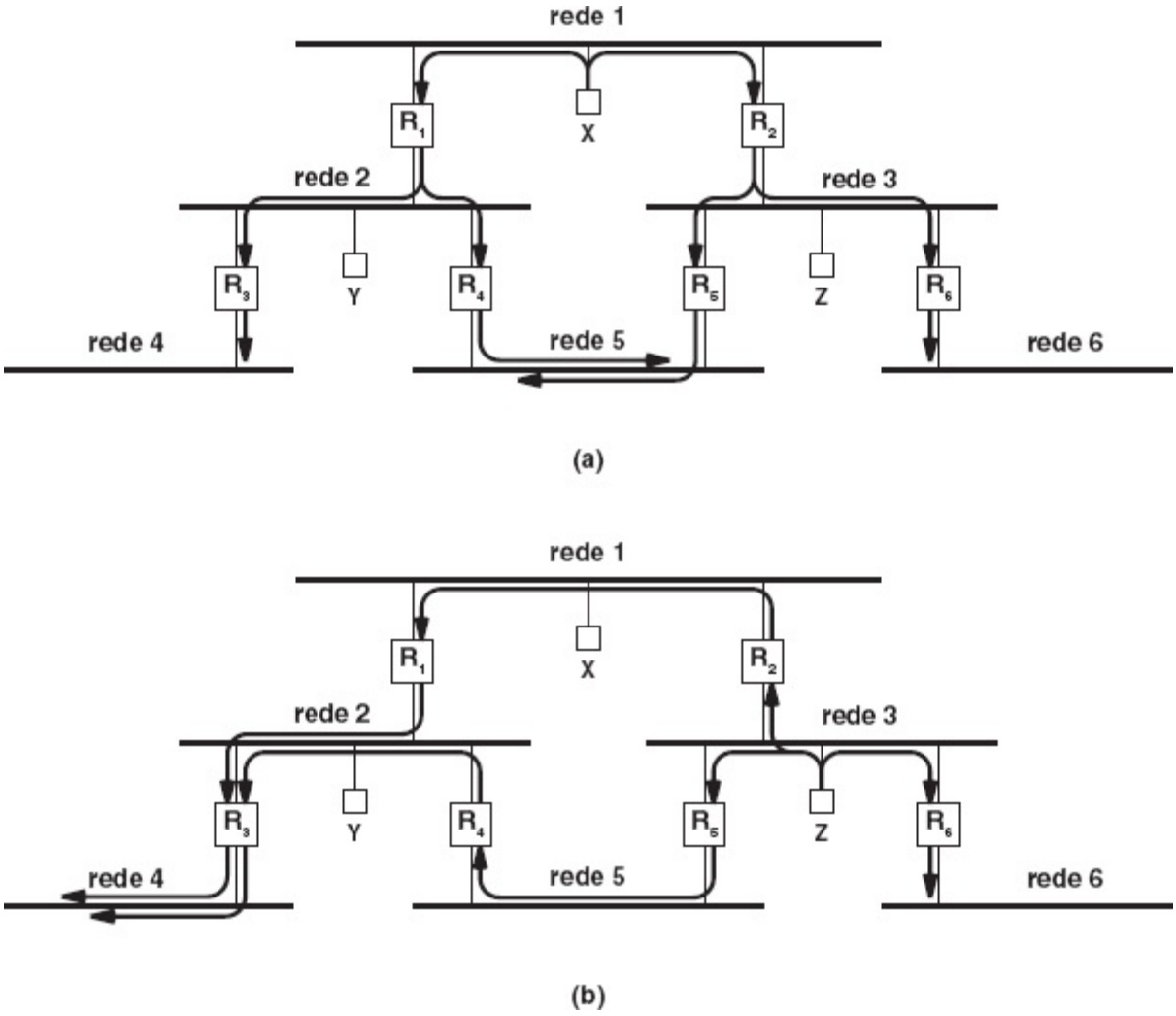


Figura 15.16 Exemplos dos caminhos que um datagrama multicast segue sob o TRPF considerando que a origem é (a) host *X* e (b) host *Z*, e o grupo tem um membro em cada uma das redes. O número de cópias recebidas depende da origem.

15.22 Árvores multicast

Os pesquisadores usam a terminologia da teoria dos grafos para descrever o conjunto dos caminhos de uma determinada origem para todos os membros de um grupo multicast: os caminhos definem uma *árvore* grafo-teórica,* que, algumas vezes, é chamada de *árvore de encaminhamento* ou *árvore de entrega*. Cada roteador multicast corresponde a um *nó* na árvore, e uma rede que conecta dois roteadores corresponde a uma *aresta* na árvore. A origem de um datagrama é a *raiz* ou o *nó de raiz da árvore*. Finalmente, o último roteador ao longo de cada um dos caminhos desde a origem é

chamado de roteador *folha*. A terminologia algumas vezes é aplicada às redes também – os pesquisadores chamam uma rede contendo um roteador folha de *rede folha* (*leaf network*).

Como um exemplo da terminologia, considere a parte (a) da Figura 15.16, a qual mostra uma árvore com raiz X, e folhas R_3, R_4, R_5, e R_6. Tecnicamente, a parte (b) não mostra uma árvore porque o roteador R_3 está em dois caminhos. Informalmente, os pesquisadores em geral ignoram os detalhes e se referem a esses grafos como árvores.

A terminologia dos grafos permite que expressemos a seguir um importante princípio.

> *Uma árvore de encaminhamento multicast é definida como um conjunto de caminhos através de roteadores multicast, a partir de uma origem para todos os membros do grupo multicast. Para um dado grupo multicast, cada origem de datagramas possível pode determinar uma árvore de encaminhamento diferente.*

Uma das consequências imediatas do princípio diz respeito ao tamanho das tabelas usadas para encaminhar multicast. Diferentemente das tabelas de encaminhamento unicast, cada entrada em uma tabela multicast é identificada por um par:

(grupo multicast, origem).

Conceitualmente, *origem* identifica um único host que pode enviar datagramas para o grupo (ou seja, qualquer host na internet). Na prática, não é uma boa ideia manter uma entrada separada para cada host porque as árvores de encaminhamento definidas por todos os hosts em uma única rede são idênticas. Assim, para economizar espaço, os protocolos de encaminhamento usam um prefixo de rede como uma *origem*. Ou seja, cada roteador define uma entrada de encaminhamento que é usada para todos os hosts na mesma rede IP.

Agregar entradas por prefixo de rede e não por endereço de host reduz drasticamente o tamanho da tabela. No entanto, as tabelas de encaminhamento multicast podem se tornar muito maiores do que as

tabelas de encaminhamento convencionais. Diferentemente de uma tabela convencional unicast em que o tamanho é proporcional ao número de redes na internet subjacente, uma tabela de multicast possui tamanho proporcional ao produto do número de redes na internet pelo número de grupos multicast.

15.23 A essência da propagação de rota multicast

O leitor atento pode ter percebido uma inconsistência entre os recursos do multicasting IP e do TRPF. Dissemos que o TRPF é usado no lugar do RPF convencional para evitar tráfego desnecessário: o TRPF não encaminha um datagrama para uma rede a menos que essa rede leve a pelo menos um membro do grupo. Consequentemente, um roteador multicast precisa ter conhecimento da associação de grupo. Também dissemos que o IP permite que qualquer host se una a um grupo multicast ou o deixe a qualquer hora, o que resulta em rápidas mudanças de associação.

Mais importante, a associação não segue o escopo local – um host que se une pode estar longe de algum roteador que está encaminhando datagramas para o grupo. Assim, as informações de associação de grupo precisam ser propagadas através da internet subjacente.

O problema da associação de grupo dinâmico é fundamental para o roteamento multicast; todos os esquemas de roteamento multicast fornecem um mecanismo para propagar as informações de associação, assim como uma maneira de usar as informações ao encaminhar datagramas. Em geral, como a associação pode mudar rapidamente, as informações disponíveis em um determinado roteador são imperfeitas, de modo que as atualizações de rota podem defasar mudanças. Portanto, um projeto de multicast representa um equilíbrio entre a sobrecarga do tráfego de roteamento extra e a transmissão de dados ineficiente. Por um lado, se as informações de associação de grupo não forem propagadas rapidamente, os roteadores multicast não tomarão decisões ideais (ou seja, eles encaminharão datagramas através de algumas redes desnecessariamente ou deixarão de enviar datagramas para todos os membros). Por outro lado, um esquema de roteamento multicast que comunica cada mudança de associação a cada roteador está condenado porque o tráfego resultante pode sobrecarregar uma internet. Cada projeto escolhe um compromisso entre os dois extremos.

15.24 Reverse Path Multicasting

Uma das primeiras formas de roteamento multicast foi derivada do TRPF.

Conhecido como *Reverse Path Multicasting* (*RPM*), o esquema estende o TRPF para torná-lo mais dinâmico. Três suposições baseiam o projeto. Primeiro, é mais importante garantir que um datagrama multicast alcance cada membro do grupo para o qual ele é enviado do que eliminar transmissão desnecessária. Segundo, os roteadores multicast contêm uma tabela de encaminhamento convencional que possui informações corretas. Terceiro, o roteamento multicast deve melhorar a eficiência quando possível (ou seja, eliminar transmissão desnecessária).

O RPM usa um processo de duas etapas. Quando começa, o RPM usa o esquema de broadcast para enviar uma cópia de cada datagrama através de todas as redes na internet. Fazer isso garante que todos os membros de grupo recebam uma cópia. Simultaneamente, o RPM continua fazendo os roteadores multicast informarem um ao outro sobre caminhos que não levam a membros do grupo. Uma vez que ele aprende que nenhum membro de grupo se encontra ao longo de um determinado caminho, um roteador para de encaminhar ao longo desse caminho.

Como os roteadores aprendem sobre a localização dos membros de grupo? Como na maioria dos esquemas de roteamento multicast, o RPM propaga informações de associação de baixo para cima. As informações começam com hosts que escolhem se unir ou deixar grupos. Os hosts comunicam informações de associação com seu roteador local usando protocolos IGMP ou MLDv2. O protocolo local só informa a um roteador multicast sobre membros locais em cada uma de suas redes diretamente conectadas; o roteador não vai aprender sobre membros em grupos distantes. Como consequência, um roteador conectado por uma rede folha ao resto da Internet pode decidir que datagrama multicast encaminhar pela rede folha. Se uma rede folha não tiver qualquer membro para o grupo *G*, o roteador folha vai informar ao próximo roteador do caminho de volta à origem e vai começar a encaminhar datagramas que cheguem destinados para o grupo *G*. Por outro lado, se todos os hosts além de um dado roteador deixam o grupo *G*, o roteador informa ao próximo roteador ao longo do caminho para a origem para interromper o envio de datagramas destinados a *G*.

Usando a terminologia grafo-teórica, dizemos que, quando um roteador aprende que um grupo não possui membros ao longo de um caminho e para de encaminhar, ele *podou* (removeu) esse caminho da árvore de encaminhamento. Na verdade, o RPM é chamado de estratégia de

broadcast-and-prune porque um roteador difunde (usando RPF) até que receba informação que permita a ele podar(prune) um caminho. Os pesquisadores também usam outro termo para o algoritmo RPM: dizem que o sistema é *orientado por dados* porque um roteador não envia informações de associação de grupo para qualquer outro roteador até que cheguem datagramas para esse grupo.

No modelo baseado em dados, um roteador multicast também precisa lidar com o caso em que um host decide se unir a um determinado grupo após o roteador ter podado o caminho para esse grupo: quando um host informa a um roteador multicast local, M_1 que o host quer voltar a se unir a um grupo particular, M_1 consulta seu registro do grupo e obtém o endereço do roteador multicast, M_2, para o qual anteriormente enviou uma requisição de prune. O roteador M_1 envia uma nova mensagem que desfaz o efeito da poda anterior e faz com que os datagramas fluam novamente. Essas mensagens são conhecidas como *requisições de enxerto*, e dizemos que o algoritmo enxertou a ramificação aparada anteriormente de volta na árvore.

15.25 Exemplo de protocolos de roteamento

O IETF investigou muitos protocolos multicast, incluindo *Distance Vector Multicast Routing Protocol (DVMRP)*, *Core Based Trees (CBT)* e *Protocol Independent Multicast (PIM)*. Embora os protocolos tenham sido implementados e os fornecedores tenham oferecido algum suporte, nenhum deles se tornou amplamente usado. A próxima seção fornece uma breve descrição de cada protocolo.

15.25.1 Protocolo de roteamento multicast vetor de distância e tunelamento

Um antigo protocolo, conhecido como *Distance Vector Multicast Routing Protocol (DVMRP)*, permite que roteadores multicast transmitam informações de associação de grupo e de roteamento entre eles próprios. O DVMRP se assemelha ao protocolo RIP descrito no Capítulo 14, mas foi estendido para multicast. Basicamente, o protocolo passa informações sobre

associação de grupo multicast atuais e o custo para transferir datagramas entre roteadores. Para cada par (grupo, origem) possível, os roteadores impõem uma árvore de encaminhamento no topo das interconexões físicas. Quando um roteador recebe um datagrama destinado a um grupo multicast IP, ele envia uma cópia do datagrama através dos links de rede que correspondem a ramificações na árvore de encaminhamento.* O DVMPR é implementado por um programa UNIX chamado *mrouted*, que usa um *kernel multicast* especial.

O *mrouted* usa o tunelamento multicast para permitir que sites encaminhem multicast através da Internet. Em cada site, um gerente configura um *túnel mrouted* para outros sites. O túnel usa encapsulamento IP em IP para enviar multicast. Ou seja, quando recebe um datagrama multicast gerado por um host local, o *mrouted* encapsula o datagrama em um datagrama unicast convencional e encaminha uma cópia para o *mrouted* em cada um dos outros sites. Quando recebe um datagrama unicast através de um de seus túneis, o *mrouted* extrai o datagrama multicast e depois encaminha de acordo com sua tabela de roteamento multicast.

15.25.2 Core Based Trees (CBT)

O protocolo de roteamento multicast *Core Based Trees* (*CBT*) usa outra abordagem para construir um sistema de encaminhamento multicast. O CBT evita o broadcasting e permite que todas as origens compartilhem a mesma árvore de encaminhamento sempre que possível. Para evitar broadcasting, o CBT não encaminha multicasts ao longo de um caminho até que um ou mais hosts no caminho se unam ao grupo multicast. Portanto, o CBT inverte o método inundar e aparar (flood-and-prune) usado pelo DVMRP – em vez de encaminhar datagramas até que informações negativas tenham sido propagadas, o CBT não encaminha ao longo de um caminho até que informações positivas tenham sido recebidas. Dizemos que, em vez de usar o modelo baseado em dados, o CBT usa um modelo *baseado em demanda.*

O modelo baseado em demanda no CBT significa que, quando um host usa o IGMP para se unir a um grupo específico, o roteador local precisa então informar outros roteadores antes que datagramas sejam encaminhados. Que roteador ou roteadores devem ser informados? A

pergunta é crítica em todos os esquemas de roteamento multicast baseados em demanda. Lembre-se de que, em um esquema baseado em dados, um roteador usa a chegada do tráfego de dados a fim de saber para onde enviar mensagens de roteamento (ele propaga mensagens de roteamento de volta pelas redes das quais o tráfego chega). Entretanto, em um esquema baseado em demanda, nenhum tráfego chegará para um grupo até que informações de associação tenham sido propagadas.

O CBT utiliza uma combinação de algoritmos estáticos e dinâmicos para construir uma árvore de encaminhamento multicast. Para tornar o esquema escalável, o CBT divide a internet subjacente em *regiões*, e o tamanho de uma região é determinado pelos administradores da rede. Dentro de cada região, um dos roteadores é designado como um *roteador central* e outros roteadores na região precisam ser configurados para conhecerem o roteador central da sua região ou para usar um *mecanismo de descoberta* dinâmico para descobrirem o roteador central quando inicializam.

O conhecimento de um roteador central é importante porque permite que roteadores multicast em uma região formem uma *árvore compartilhada* para a região. Assim que um host se une a um grupo multicast, um roteador central, L, recebe a requisição de host. O roteador L gera uma *requisição de união* CBT que ele envia para o roteador central usando encaminhamento unicast convencional. Cada roteador intermediário ao longo do caminho até o roteador central examina a requisição. Quando a requisição alcança um roteador R que já é parte da árvore compartilhada CBT, R retorna um reconhecimento, passa as informações de associação de grupo para seu pai e inicia o tráfego de encaminhamento para o novo grupo. Enquanto o reconhecimento é transferido de volta para o roteador folha, os roteadores intermediários examinam a mensagem e configuram sua tabela de encaminhamento multicast para encaminhar datagramas ao grupo. Portanto, o roteador L é ligado na árvore de encaminhamento no roteador R.

Podemos fazer o resumo a seguir.

> *Como o CBT usa um modelo baseado em demanda, ele divide uma internet em regiões e designa um* roteador central *para cada região; outros roteadores na região constroem dinamicamente uma árvore de*

encaminhamento enviando requisições de união para o central.

15.25.3 Protocol Independent Multicast (PIM)

Na realidade, o PIM consiste em dois protocolos independentes que compartilham pouco mais do que o nome e os formatos de cabeçalho de mensagem básicos: *Protocol Independent Multicast – Dense Mode* (*PIM-DM*) – e *Protocol Independent Multicast – Sparse Mode* (*PIM-SM*). A diferença surge porque nem um único protocolo funciona bem em todas as situações. Em especial, o modo denso do PIM é projetado para um ambiente de LAN em que todas ou quase todas as redes possuem hosts escutando cada grupo multicast; enquanto o modo esparso do PIM é projetado para acomodar um ambiente remoto em que os membros de um determinado grupo multicast ocupam um pequeno subconjunto de todas as redes possíveis.

O termo *independência de protocolo* surge porque o PIM considera uma tabela de encaminhamento unicast tradicional que contém um caminho mais curto até cada destino. Como o PIM não especifica como essa tabela deve ser construída, um protocolo de roteamento arbitrário pode ser usado. Assim, o PIM opera *independentemente* do protocolo(s) de atualização de roteamento unicast que um roteador emprega.

Para acomodar muitos escutadores, o PIM-DM usa um método broadcast-and-prune em que datagramas são encaminhados para todos os roteadores usando RPF até que um roteador envie uma requisição de *prune* explícita. Por outro lado, o modo esparso do PIM pode ser visto como uma extensão dos conceitos básicos do CBT. O modo esparso designa um roteador, chamado *Rendezvous Point* (RP), o equivalente funcional de um roteador central CBT.

15.26 Multicast seguro e implosões de ACK

O termo *multicast seguro* se refere a qualquer sistema que usa entrega multicast, mas também garante que todos os membros de grupo recebam os dados em ordem, sem qualquer perda, duplicação ou dano. Teoricamente, o multicast seguro combina a vantagem de um esquema de encaminhamento que é mais eficiente do que o broadcast com a vantagem de fazer todos os dados chegarem intactos. Portanto, o multicast seguro possui uma grande

vantagem e aplicabilidade potenciais (por exemplo, uma bolsa de valores poderia usar o multicast seguro para entregar preços de ações a muitos destinos).

Na prática, o multicast seguro não é tão geral ou simples quanto parece. Primeiro, se um grupo multicast possui vários emissores, a noção de entregar datagramas em sequência se torna sem sentido. Segundo, vimos que esquemas de encaminhamento multicast amplamente usados como o RPF podem produzir duplicação mesmo em internets pequenas. Terceiro, além da garantia de que todos os dados fatalmente chegarão, aplicações como áudio ou vídeo esperam que sistemas seguros evitem o atraso e o jitter. Quarto, como a confiabilidade requer reconhecimentos e um grupo multicast pode ter um número arbitrário de membros, os protocolos seguros tradicionais requerem que um emissor manipule um número arbitrário de reconhecimentos. Infelizmente, nenhum computador possui poder de processamento suficiente para fazer isso. Nós nos referimos a esse problema como uma *implosão* de ACK, e esta se tornou o foco de muita pesquisa.

Para resolver o problema da implosão de ACK, os protocolos de multicast seguro utilizam um método hierárquico no qual o multicasting é restrito a uma única origem.* Antes de os dados serem enviados, uma árvore de encaminhamento é estabelecida da origem a todos os membros e grupo, e *pontos de reconhecimento* (*acknowledgement point*) precisam ser identificados.

Um ponto de reconhecimento, que também é chamado de *agregador de reconhecimento* (*acknowledgement aggregator*) ou *roteador designado* (*DR*), consiste em um roteador na árvore de encaminhamento que concorda em colocar cópias dos dados no cache e processar reconhecimentos dos roteadores ou hosts mais abaixo na árvore. Se uma retransmissão é necessária, o ponto de reconhecimento obtém uma cópia do seu cache.

A maioria dos esquemas de multicast seguro usa reconhecimentos negativos ao invés de positivos – um host de recebimento não responde a menos que um datagrama seja perdido. Para permitir que um host detecte perda, cada datagrama precisa ser atribuído a um número de sequência único. Quando aparece uma falha na sequência de números, o host detecta perda e envia um NACK para requisitar a retransmissão do datagrama perdido. O NACK é propagado ao longo da árvore de encaminhamento em direção à origem até que alcance um ponto de reconhecimento. Este processa o NACK e retransmite uma cópia do datagrama perdido ao longo

da árvore de encaminhamento.

Como um ponto de reconhecimento garante que tem uma cópia de todos os datagramas na sequência? Ele usa o mesmo esquema de um host. Quando um datagrama chega, o ponto de reconhecimento verifica o número de sequência, coloca uma cópia em sua memória e, então, continua a propagar o datagrama pela árvore de encaminhamento. Se descobrir que um datagrama está faltando, o ponto de reconhecimento envia um NACK para cima na árvore em direção à origem. O NACK alcança outro ponto de reconhecimento que tenha uma cópia do datagrama (caso em que o ponto de reconhecimento transmite uma segunda cópia) ou o NAC alcança a origem (que retransmite o datagrama faltante).

A escolha de uma topologia de ramificação e pontos de reconhecimento é vital para o sucesso de um esquema de multicast seguro. Sem pontos de reconhecimento suficientes, um datagrama faltante pode causar uma implosão de ACK. Em especial, se um determinado roteador tiver muitos descendentes, um datagrama perdido pode fazer com que o roteador seja preenchido com requisições de retransmissão. Infelizmente, a automatização da escolha de pontos de reconhecimento não tem demonstrado ser simples. Consequentemente, muitos protocolos de multicast seguro exigem configuração manual. Portanto, o multicast seguro é mais apropriado para: serviços que tendem a persistir por longos períodos de tempo, topologias que não mudam rapidamente e situações nas quais roteadores intermediários concordam em servir como pontos de reconhecimento.

Existe um método alternativo para a segurança? Alguns pesquisadores experimentaram com protocolos que incorporavam informação redundante para reduzir ou eliminar retransmissões. Um esquema envia datagramas redundantes. Em vez de enviar uma simples cópia de cada datagrama, a origem envia N cópias (normalmente 2 ou 3) de datagramas redundantes que funcionam especialmente bem quando roteadores implementam uma estratégia *Random Early Discard (RED)*, pois a probabilidade de mais de uma cópia sendo descartada é extremamente pequena.

Outro método para a redundância envolve *encaminhar códigos de correção de erro*. Análogo aos códigos de correção de erro usados com os CDs de áudio, o esquema exige que um emissor incorpore informações de correção de erro em cada datagrama em um fluxo de dados. Se um datagrama for perdido, o código de correção de erro conterá informações redundantes suficientes para permitir que um receptor reconstrua o

datagrama ausente sem requisitar uma retransmissão.

15.27 Resumo

O multicasting IP é uma abstração do multicasting de hardware. Ele permite a distribuição de um datagrama para um subconjunto arbitrário de computadores. Tanto o IPv4 quanto o IPv6 definem um conjunto de endereços multicast. O IPv6 usa o segundo octeto para representar o escopo, permitindo ser independente do serviço. O multicast IP usa multicast de hardware, se disponível.

Os grupos multicast são dinâmicos: um host pode se unir ou deixar um grupo a qualquer hora. Para multicast local, os hosts precisam apenas da capacidade de enviar e receber datagramas multicast. Para multicasting IP que se espalha por múltiplas redes, os roteadores multicast devem propagar informações de associação de grupo e organizar o roteamento de modo que cada membro de um grupo multicast receba uma cópia de cada datagrama enviado para o grupo. O esquema multicasting IP é complicado pela regra de que um host arbitrário pode enviar para um grupo multicast, mesmo que o host não seja um membro.

Os hosts comunicam sua associação de grupo para roteadores multicast usando IGMP (IPv4) ou MLDv2 (IPv6). Os protocolos são intimamente ligados e foram projetados para serem eficientes em evitar o uso excessivo de recursos de rede.

Diversos protocolos foram projetados para propagar informações de roteamento multicast através de uma internet. Os dois métodos básicos são o método baseado em dados e o método baseado em demanda. Em qualquer caso, a quantidade de informação em uma tabela de roteamento multicast é muito maior do que em uma tabela de roteamento unicast, pois o multicasting exige entradas para cada par (grupo, origem).

Nem todos os roteadores na Internet global propagam rotas multicast ou encaminham tráfego multicast. Pode ser usado túnel para conectar *ilhas* multicast que estão separadas por partes de uma internet que não suporta roteamento multicast. Quando se usa um túnel, um programa encapsula um datagrama multicast em um datagrama unicast convencional. O receptor extrai e lida com o datagrama multicast.

O multicast seguro se refere a um esquema que utiliza encaminhamento multicast, mas fornece semântica de entrega segura. Para evitar o problema da implosão de ACK, os esquemas de multicast seguro usam uma hierarquia de pontos de reconhecimento ou enviam informações redundantes.

EXERCÍCIOS

15.1 O padrão IPv4 sugere usar 23 bits de um endereço multicast IP para formar um endereço multicast de hardware. Nesse esquema, quantos endereços multicast IP mapeiam em um único endereço multicast de hardware?

15.2 Responda à questão anterior para o IPv6.

15.3 Discuta o fato de que o ID de grupo multicast não precisa de tantos bits quantos tem alocados. Suponha um ID de grupo usa 23 bits e analise as desvantagens. (Dica: quais são os limites práticos sobre o número de grupos aos quais um host pode pertencer e o número de hosts em uma única rede?)

15.4 O software IP sempre precisa verificar os endereços de destino nos datagramas multicast que chegam e descartar datagramas se o host não estiver no grupo multicast especificado. Explique como o host pode receber um multicast destinado a um grupo do qual esse host não é membro.

15.5 Os roteadores multicast precisam saber se um grupo possui membros em uma determinada rede porque o roteador precisa saber a quais árvores multicast se unir. Existe alguma vantagem no fato de os roteadores saberem o conjunto de hosts exato em uma rede que pertence a um determinado grupo multicast? Explique.

15.6 Encontre três aplicações em seu ambiente que podem se beneficiar do multicast IP.

15.7 O padrão diz que o software IP precisa distribuir uma cópia de qualquer datagrama multicast *que sai* para os programas aplicativos no mesmo host que pertence ao grupo multicast especificado. Esse projeto torna a programação mais fácil ou mais difícil? Explique.

15.8 Quando o hardware subjacente não aceita multicast, o multicast IP usa broadcast de hardware para distribuição. Como isso poderia causar problemas? Existe alguma vantagem em usar multicast IP nessas redes?

15.9 O DVMRP foi derivado do RIP. Leia a RFC 1075 sobre o DVMRP e compare os dois protocolos. Quanto o DVMRP é mais complexo que o RIP?

15.10 A versão 3 do IGMP e o MLDv2 incluem ambos uma medida de robustez que se destina a acomodar perda de pacote permitindo transmissão de múltiplas cópias de uma mensagem. Como o protocolo chega a uma estimativa do valor da robustez necessária em

uma determinada rede?

15.11 Explique por que um host multi-homed pode precisar se unir a um grupo multicast em uma rede, mas não em outra. (Dica: considere uma teleconferência de áudio.)

15.12 Suponha que um programador de aplicação faz uma escolha para simplificar a programação: quando se unir a um grupo de multicast, basta se juntar em cada rede à qual o host conecta. Mostrar que a adesão em todas as interfaces locais pode levar a um tráfego arbitrário em redes remotas.

15.13 Estime o tamanho da tabela de encaminhamento multicast necessária para manipular multicast do áudio de 100 estações de rádio, se cada estação tem um total de dez milhões de ouvintes em locais aleatórios em todo o mundo.

15.14 Considere-se um provedor de TV a cabo usando a tecnologia IP. Suponha que cada set-top box usa IP e elabora um esquema em que o provedor de cabo transmite todos os canais para um ponto de distribuição de bairro e, em seguida, usa multicast para entregar para as residências.

15.15 Argumente que apenas dois tipos de multicast são práticos na Internet: serviços comerciais configurados estaticamente que transmitem multicast para um grande número de assinantes e serviços configurados de forma dinâmica, que incluem poucos participantes (por exemplo, membros da família em três famílias que participam de um único telefonema IP).

15.16 Considere o multicast seguro obtido através de transmissão redundante. Se um determinado link tem uma alta probabilidade de dano, é melhor enviar cópias redundantes de um datagrama ou enviar uma cópia que usa códigos de correção de erro? Explique.

15.17 O modelo de roteamento multicast baseado em dados funciona melhor em redes locais que possuem baixo atraso e capacidade extra, enquanto o modelo baseado em demanda funciona melhor em um ambiente remoto que tem alto atraso e capacidade limitada. Faz sentido criar um único protocolo que combine os dois esquemas? Sim ou não? Por quê? (Dica: investigue o MOSPF.)

15.18 Leia a especificação do protocolo PIM-SM para descobrir como o protocolo define a noção de *esparso*. Encontre um exemplo de uma internet em que a população dos membros de grupo é esparsa, mas para a qual o DVMRP é um protocolo de roteamento de multicast

melhor.

15.19 Invente uma medida quantitativa que possa ser usada para decidir quando o PIM-SM deve mudar de uma árvore compartilhada para uma árvore de caminho mais curto.

---

* A notação hexadecimal hifenizada representa cada octeto como dois dígitos hexadecimais com octetos separados por traço; o subscrito *16* pode ser omitido somente quando o contexto não apresente ambiguidade.

* O IEEE atribui um prefixo *Identificador Organizacional Único* (*Organizational Unique Identifier – OUI*) para cada organização que cria endereços Ethernet.

* Na prática, os sites que usam multicast IP normalmente configuram roteadores convencionais para lidar com multicast para reencaminhamento, bem como com o encaminhamento unicast.

* O grupo todos os sistemas, 224.0.0.1, é uma exceção – um host nunca relata associações nesse grupo.

* O esquema Reverse Path Forwarding (RPF) algumas vezes é chamado de *Reverse Path Broadcasting* (*RPB*).

* Um grafo é uma árvore se ele não contiver quaisquer ciclos (isto é, um roteador não aparece em mais de um caminho).

* O DVMRP mudou substancialmente da versão 2 para a 3, quando incorporou o algoritmo RPM descrito anteriormente.

* Observe que uma única origem não limita a funcionalidade porque a origem pode concordar em encaminhar qualquer mensagem que receba através de unicast. Assim, um host arbitrário pode enviar um pacote para a origem, que então envia o pacote através de multicast para o grupo.

CAPÍTULO 16

Comutação de rótulo, fluxos e MPLS

CONTEÚDOS DO CAPÍTULO

16.1 Introdução

Os capítulos anteriores descreveram o endereçamento IP e como hosts e roteadores usam uma tabela de roteamento e o algoritmo de combinação de prefixo mais longo para pesquisar um próximo salto e encaminhar datagramas. Este capítulo considera uma abordagem alternativa que evita o overhead da combinação de prefixo mais longo. O capítulo apresenta a ideia básica da comutação de rótulo, explica a tecnologia e descreve seu uso para engenharia de tráfego.

O próximo capítulo continua a discussão considerando rede definida por software, e explica como a ideia básica de rótulo de fluxo pode ser usada em uma rede desse tipo.

16.2 Tecnologia de comutação

Na década de 1980, quando a Internet crescia em popularidade, os pesquisadores começaram a explorar maneiras de aumentar o desempenho dos sistemas de processamento de pacotes. Uma ideia que surgiu foi, por fim, adotada pelos fornecedores comerciais: substituir a abordagem de comutação de pacotes sem conexão do IP, que exige pesquisa em uma tabela de prefixo mais longo, por uma abordagem de conexão orientada, que acomoda um algoritmo de pesquisa mais rápido. O conceito geral, conhecido como *comutação de rótulo* (*label switching*), levou os

fornecedores a criarem uma tecnologia de rede conhecida como *Asynchronous Transfer Mode* (*ATM*). Na década de 1990, o ATM teve popularidade de curta duração, mas a moda passou.

Havia várias razões para a eventual morte do ATM. O principal motivo foi econômico: comutadores Ethernet e roteadores IP eram muito menos caros do que os comutadores ATM e, uma vez que o IETF criou as tecnologias de Comutação de Rótulos utilizando roteadores IP convencionais, os departamentos de TI não encontraram razões convincentes para o ATM. Uma vez que entendemos os conceitos gerais, vemos como é possível implementar a comutação com um roteador convencional que não depende de hardware de conexão orientada, que é caro.

Na base da técnica de comutação reside a ideia principal sobre pesquisa: se existem N itens em uma tabela de encaminhamento, um computador exige uma média de aproximadamente $log_2 N$ etapas para executar uma combinação de prefixo mais longo. Os proponentes da comutação de rótulo mostraram que um hardware pode executar um índice matriz em uma etapa. Além disso, indexar pode se traduzir diretamente em um hardware combinatório, enquanto a pesquisa normalmente envolve múltiplas referências de memória.

As tecnologias de comutação exploram a indexação para conseguir uma alta velocidade de encaminhamento. Para isso, cada pacote transporta um valor inteiro pequeno, conhecido como *rótulo* (*label*). Quando um pacote chega a um comutador, este extrai o rótulo e usa o valor como um índice para a tabela que especifica a ação apropriada. Cada comutador tem um conjunto de interfaces de saída, e a ação normalmente consiste em enviar um pacote por uma dessas interfaces. A Figura 16.1 ilustra o conceito.

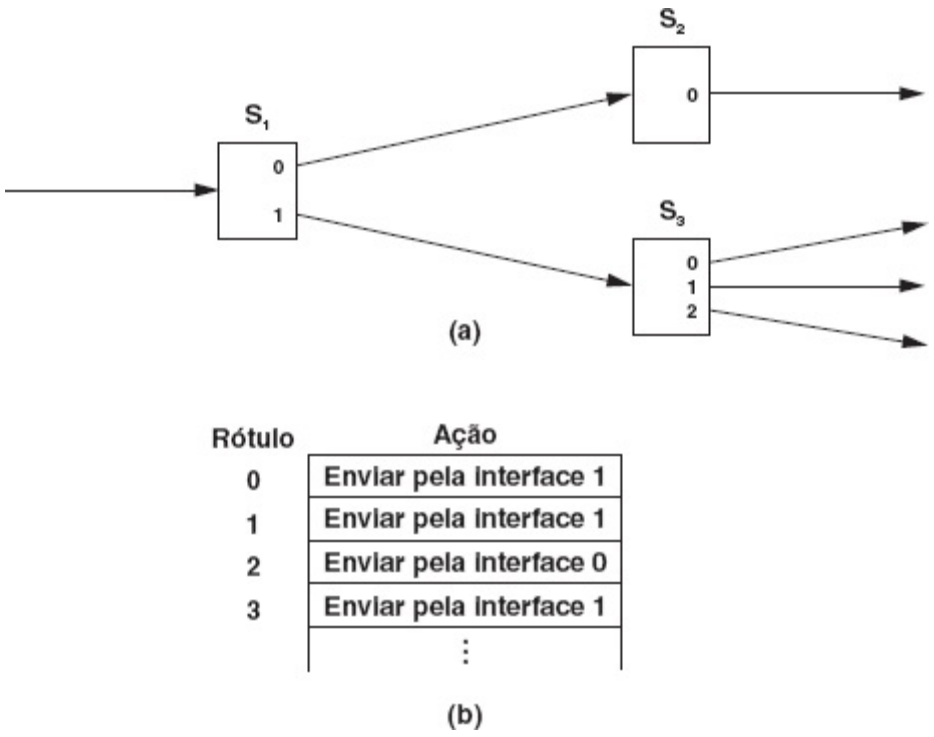


Figura 16.1 Ilustração da comutação básica com (a) uma rede de três comutadores interconectados e (b) uma tabela do comutador S_1.

Na figura, as entradas da tabela especificam como o comutador S_1 encaminha pacotes com rótulos no intervalo de 0 a 3. De acordo com a tabela, um pacote com rótulo *2* será encaminhado pela interface *0*, que leva ao comutador S_2; um pacote com um rótulo *0*, *1* ou *3* será encaminhado pela interface *1*, que leva ao comutador S_3.

16.3 Fluxos e configurações de fluxo

A principal desvantagem do esquema de comutação básico descrito anteriormente surge porque cada rótulo consiste em um valor inteiro pequeno. Como o esquema pode ser estendido para uma rede que possui muitos destinos? A chave para a resposta da questão envolve repensar o que assumimos sobre internets e encaminhamento conforme descrito a seguir.

- Em vez de focar o destino, focar *fluxos* de pacote.
- Em vez de assumir que as tabelas de encaminhamento permanecem estáticas, imaginar um sistema que pode configurar ou mudar rapidamente as tabelas de encaminhamento.

Definimos um *fluxo de pacote* como sendo uma sequência de pacotes enviados de uma dada origem para um dado destino. Por exemplo, todos os pacotes em uma conexão TCP constituem um único fluxo. Um fluxo também pode corresponder a uma chamada telefônica VoIP ou a uma interação de longa duração, tal como todos os pacotes enviados de um roteador em um escritório da costa Leste de uma companhia para um roteador em um escritório da costa Oeste dessa companhia. Podemos a seguir resumir a ideia-chave.

Tecnologias de comutação usam a abstração de fluxo e criam encaminhamento por fluxos em vez de encaminhamento por destinos.

Em termos de fluxos de pacotes, vamos supor que exista um mecanismo para criar entradas em comutadores quando um fluxo está configurado e remova as entradas quando um fluxo termina. Ao contrário de uma internet convencional, em que as tabelas de encaminhamento permanecem estáticas, espera-se que as tabelas em comutadores mudem com frequência. Claro, o sistema para configurar os fluxos precisa de mecanismos que compreendam os destinos e como alcançá-los. Assim, a configuração do fluxo pode precisar entender o encaminhamento convencional que utiliza destinos. Por enquanto, vamos nos concentrar no funcionamento dos comutadores e adiar a discussão sobre a configuração do fluxo até o final do capítulo.

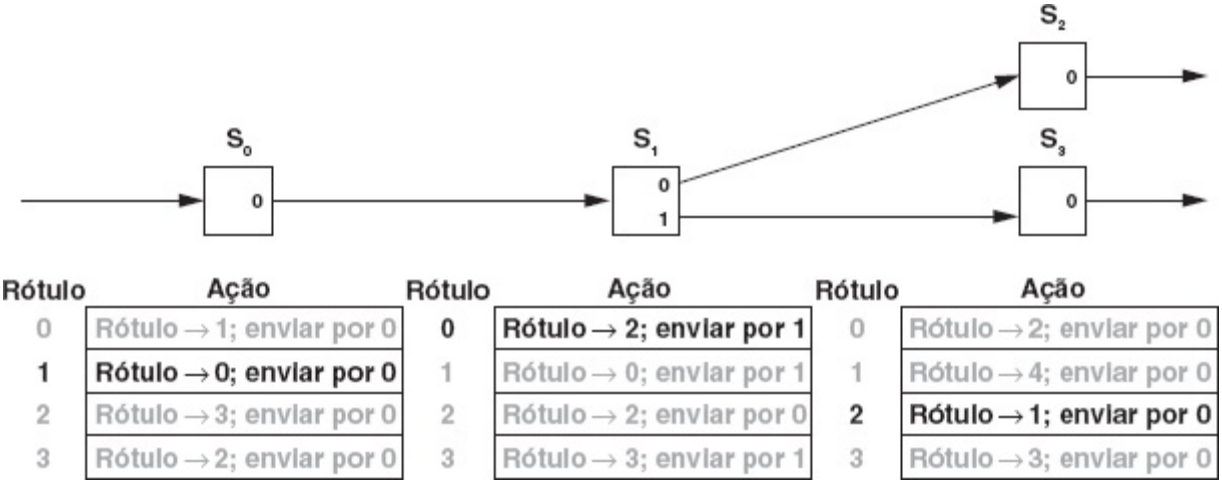
16.4 Redes grandes, troca de rótulo e caminhos

Na descrição simplista anterior, a cada fluxo deve ser atribuído um rótulo único. Um pacote carrega o rótulo atribuído ao seu fluxo e, em cada comutador, o mecanismo de encaminhamento usa o rótulo para selecionar uma interface de saída. Infelizmente, o esquema simplista não escala para grandes redes. Antes de uma etiqueta única poder ser atribuída a um fluxo, seria necessário que o software verificasse que nenhum outro fluxo em qualquer parte da internet está utilizando o rótulo. Por isso, antes que um fluxo possa ser estabelecido, o sistema de configuração teria de se comunicar com cada comutador na internet.

Os desenvolvedores encontraram uma forma engenhosa para resolver o problema de escala, preservando a velocidade de comutação. A solução permite que o rótulo seja escolhido de forma independente para cada comutador ao longo do caminho. Isto é, em vez de um rótulo único para um dado fluxo, um novo rótulo pode ser escolhido em cada comutador ao

longo de um caminho. Apenas um mecanismo adicional é necessário para tornar o trabalho de seleção de rótulo independente: um comutador deve ser capaz de reescrever o rótulo nos pacotes. O sistema é conhecido *como troca de rótulo, comutação de rótulo, ou reescrever rótulo* (*label swapping*, *label switching*, ou *label rewriting*)

Para entender a troca de rótulo, temos de nos concentrar no *caminho* que um dado fluxo seguirá através da rede. Quando a troca de rótulo é utilizada, o rótulo em um pacote pode mudar conforme o pacote passa de comutador em comutador. Ou seja, a ação que um comutador executa pode incluir reescrever o rótulo. A Figura 16.2 ilustra um caminho através de três comutadores.



Rótulo	Ação
0	Rótulo → 1; enviar por 0
1	**Rótulo → 0; enviar por 0**
2	Rótulo → 3; enviar por 0
3	Rótulo → 2; enviar por 0

Rótulo	Ação
0	**Rótulo → 2; enviar por 1**
1	Rótulo → 0; enviar por 1
2	Rótulo → 2; enviar por 0
3	Rótulo → 3; enviar por 1

Rótulo	Ação
0	Rótulo → 2; enviar por 0
1	Rótulo → 4; enviar por 0
2	**Rótulo → 1; enviar por 0**
3	Rótulo → 3; enviar por 0

Figura 16.2 Ilustração da troca de rótulo, em que cada comutador pode reescrever o rótulo em um pacote.

Na figura, um pacote que entra em S_0 com rótulo *1* tem o rótulo reescrito antes de ser encaminhado. Assim, quando o comutador S_1 receber o pacote, o rótulo será *0*. De modo semelhante, S_1 substitui o rótulo *1* por *2* antes de enviar o pacote para S_3; S_3 troca o rótulo por *1*. A troca de rótulo facilita a configuração de uma rede comutada, pois permite que um gerente defina um caminho pela rede sem forçar o mesmo rótulo a ser usado em cada ponto ao longo do caminho. Na verdade, um rótulo só precisa ser válido por um salto – os dois comutadores que compartilham a conexão física precisam concordar sobre o rótulo a ser atribuído a cada fluxo que atravessa a conexão.

O ponto importante é descrito a seguir.

A comutação usa uma abordagem de conexão orientada. Para evitar a necessidade de um acordo global sobre o uso de rótulos, a tecnologia permite a um gerenciador definir um caminho de comutadores sem exigir que o mesmo rótulo seja usado ao longo de todo o caminho.

16.5 Usando comutação com IP

Surge uma questão: podemos criar uma tecnologia que tenha as vantagens de troca de rótulo e as do encaminhamento baseado em destino de IP? A resposta é sim. Embora os paradigmas de conexão orientada usados com a comutação pareçam conflitar com o paradigma de IP sem conexão, os dois foram combinados. Existem três vantagens, que são apresentadas a seguir.

- encaminhamento rápido;
- informação de rota agregada;
- capacidade de gerenciar fluxos agregados.

Encaminhamento rápido. A questão é fácil de entender: a comutação permite que os roteadores realizem encaminhamento mais rápido, pois o roteador pode usar a indexação no lugar da pesquisa na tabela de roteamento. Naturalmente, implementações de encaminhamento IP convencional são extremamente rápidas (ou seja, podem encaminhar várias entradas cada uma operando a 10 Gbps sem perder qualquer pacote). Então a escolha entre comutação ou encaminhamento depende em grande parte do custo.

Informação de rota agregada. ISPs Tier-1 grandes no centro da Internet utilizam a comutação como um modo de evitar ter tabelas completas de roteamento em cada um de seus roteadores. Assim que um pacote alcança o ISP, um roteador de borda verifica o endereço de destino e escolhe um de vários caminhos. Por exemplo, um caminho pode levar a um peer ISP, um segundo caminho pode levar a outro peer ISP, e um terceiro pode levar a um grande cliente. O pacote é atribuído a um rótulo, e roteadores no backbone do ISP usam o rótulo para encaminhar o pacote. Nesses casos, os rótulos são grão grosso (coarse grain) – o rótulo só especifica o próximo ISP ao qual o pacote deve ser enviado, e não o destino final. Desse modo, todo o tráfego que vai para um determinado ISP terá o mesmo rótulo. Em outras

palavras, todo pacote viajando para o mesmo próximo salto é agregado em um único fluxo.

Capacidade de gerenciar fluxos agregados. ISPs normalmente escrevem *acordos em nível de serviço* (*Service Level Agreements – SLAs*) considerando o tráfego que pode ser enviado por um ponto de saída (peering point). Normalmente, esses SLAs se referem ao tráfego agregado (por exemplo, todo o tráfego encaminhado entre os dois ISPs ou todo o tráfego VoIP). Ter um rótulo atribuído a cada agregado facilita a implementação de mecanismos que medem ou obrigam o SLA.

16.6 Tecnologias de comutação IP e MPLS

Até aqui, descrevemos a comutação de rótulo como uma tecnologia de rede de conexão orientada de uso geral. Vamos considerar agora como a comutação de rótulo tem sido combinada com a tecnologia Internet. A Ipsilon Corporation foi uma das primeiras empresas a criar produtos que combinavam IP e comutadores de hardware. Na verdade, a Ipsilon usou comutadores de hardware ATM modificados e nomeou sua tecnologia como *comutação IP*. Desde então, outras companhias produziram uma série de projetos e nomes, incluindo *comutação tag, comutação de camada 3*, e *comutação de rótulo*. Várias ideias foram agrupadas em um padrão endossado pelo IETF, conhecido como *comutação de rótulo multiprotocolo* (*Multi-Protocol Label Switching – MPLS*). Como o termo *multi-protocol* implica, o MPLS é projetado para levar cargas arbitrárias. Na prática, o MPLS é usado quase que exclusivamente para transporte IP.

Como funciona o MPLS? A ideia geral é simples: um grande ISP (ou mesmo uma corporação que tenha uma grande internet) usa MPLS no centro de sua rede, por vezes chamada *centro* MPLS. Roteadores próximos à borda da rede ISP (ou seja, roteadores que se conectam a usuários) usam encaminhamento convencional; só os roteadores no centro entendem o MPLS e usam comutação. Na maioria dos casos, o MPLS não é usado para estabelecer caminhos para fluxos individuais. Ao contrário, o ISP configura caminhos MPLS semipermanentes através do centro que está no lugar. Por exemplo, em cada ponto de entrada para o centro, o ISP configura um caminho para cada um dos pontos de saída.

Roteadores perto da borda de uma rede ISP examinam cada datagrama e escolhem se usam um dos caminhos MPLS ou se lidam com o datagrama com o encaminhamento convencional. Por exemplo, se um cliente em uma cidade envia um datagrama para outro cliente na mesma cidade, o roteador de borda pode entregar o datagrama sem atravessar o centro MPLS. No entanto, se o datagrama deve viajar para um local remoto ou se ele requer um tratamento especial, um roteador de borda pode escolher um dos caminhos MPLS e enviar o datagrama ao longo do caminho. Quando alcança o fim do caminho MPLS, o datagrama chega ao outro roteador de borda, que irá usar o encaminhamento convencional para entrega.

Como mencionado anteriormente, uma das principais motivações para o uso do MPLS é a capacidade de agregar fluxos enquanto eles cruzam o centro do ISP. No entanto, o MPLS permite que os provedores ofereçam serviços especiais para clientes individuais também. Por exemplo, considere uma grande empresa com escritórios em Nova York e São Francisco. Suponha que a empresa quer uma conexão segura entre os seus dois sites com garantias de desempenho. Uma maneira de conseguir uma conexão consiste na locação de um circuito digital. Outra alternativa envolve MPLS: um ISP pode estabelecer um caminho MPLS entre os dois escritórios, e pode configurar os roteadores ao longo do caminho para fornecer uma garantia de desempenho.

Quando dois grandes provedores de internet se conectam a um ponto de troca de tráfego, há duas opções: cada um pode manter um centro MPLS separado ou eles podem cooperar para interligar seus centros MPLS. A desvantagem da utilização de centros separados surge quando um datagrama deve atravessar ambos os núcleos. O datagrama viaja através do núcleo do primeiro ISP até um roteador de borda que remove o rótulo. O datagrama passa então a um roteador no segundo ISP, que atribui um novo rótulo e encaminha o datagrama através do segundo centro MPLS. Se os dois ISP concordarem em interligar os seus centros MPLS, um rótulo pode ser atribuído uma só vez, e o datagrama é comutado ao longo de um caminho MPLS até atingir o roteador de borda no segundo ISP, que remove o rótulo e entrega o datagrama. A desvantagem de tal interconexão surge a partir da necessidade de coordenar – ambos os ISPs devem concordar sobre as atribuições de rótulos utilizados através da conexão.

16.7 Rótulos e atribuição de rótulos

Dissemos que um roteador de borda examina cada datagrama e escolhe se o envia ao longo de um caminho MPLS. Antes que um roteador de borda possa enviar um datagrama através de um centro MPLS, ao datagrama deve

ser atribuído um rótulo, considerado como inicial para um caminho, já que vai ser trocado pelo MPLS. Pode-se pensar o mapeamento de um datagrama para um rótulo como uma função matemática:

rótulo = f (datagrama)

na qual *rótulo* é o rótulo inicial para um dos caminhos MPLS que foi configurado e *f* é uma função que executa o mapeamento.

Na prática, a função *f* não examina normalmente o datagrama inteiro. Na maioria das vezes, *f* somente olha para determinados campos do cabeçalho. O próximo capítulo descreve classificação de pacote em detalhe.

16.8 Uso hierárquico do MPLS e uma pilha de rótulos

Consideremos o uso do MPLS em uma organização onde os roteadores estão organizados em uma hierarquia de dois níveis: uma região externa que usa o encaminhamento IP convencional e uma região interna que usa MPLS. Conforme veremos, o protocolo possibilita níveis de hierarquia adicionais. Por exemplo, suponha que uma empresa tenha três *campi*, com vários prédios em cada *campus*. A corporação pode usar o encaminhamento convencional dentro de um prédio, um nível de MPLS para interconectar os prédios dentro de um *campus* e um segundo nível de MPLS para interconectar os sites. Dois níveis de hierarquia permitem que a corporação escolha políticas para tráfego entre os sites separadamente das políticas usadas entre os prédios (por exemplo, o caminho pelo qual o tráfego atravessa entre os prédios é determinado pelo tipo de tráfego, mas todo o tráfego entre um par de sites segue o mesmo caminho).

Para oferecer uma hierarquia multinível, o MPLS incorpora uma *pilha de rótulos* (*label stack*). Ou seja, em vez de anexar um único rótulo a um pacote, o MPLS permite que vários rótulos sejam anexados. A qualquer momento, somente o rótulo superior é usado; uma vez que o pacote termine sua travessia de um nível, o rótulo superior será removido e o processamento continuará com o próximo rótulo.

No caso da nossa corporação de exemplo, é possível organizar duas áreas de MPLS – uma para o tráfego que percorre os prédios e uma para o tráfego que percorre os dois sites. Assim, quando um datagrama atravessar dois prédios em determinado site, terá um rótulo anexado (o rótulo será

removido quando o datagrama alcançar o prédio correto). Se um datagrama tiver de atravessar entre dois sites e depois até o prédio correto no site de destino, terá dois rótulos anexados. O rótulo superior será usado para mover o datagrama entre os sites e será removido quando este atingir o site correto. Quando o rótulo superior for removido, o segundo rótulo será usado para encaminhar o datagrama ao prédio correto.

16.9 Encapsulamento MPLS

Curiosamente, o MPLS não exige que uma rede subjacente use de uma tecnologia de rede de conexão orientada ou aceite comutação de rótulo. Entretanto, as redes convencionais não oferecem um meio de passar um rótulo junto com um pacote, e um cabeçalho de datagrama IPv4 não provê espaço para armazenar um rótulo. Assim, surge a pergunta: como um rótulo MPLS pode acompanhar um datagrama através de uma rede convencional? A resposta está na técnica de encapsulamento: pensamos no MPLS como um formato de pacote que pode encapsular uma carga arbitrária. O uso principal é o encapsulamento de datagramas IPv4.* A Figura 16.3 ilustra o encapsulamento conceitual.

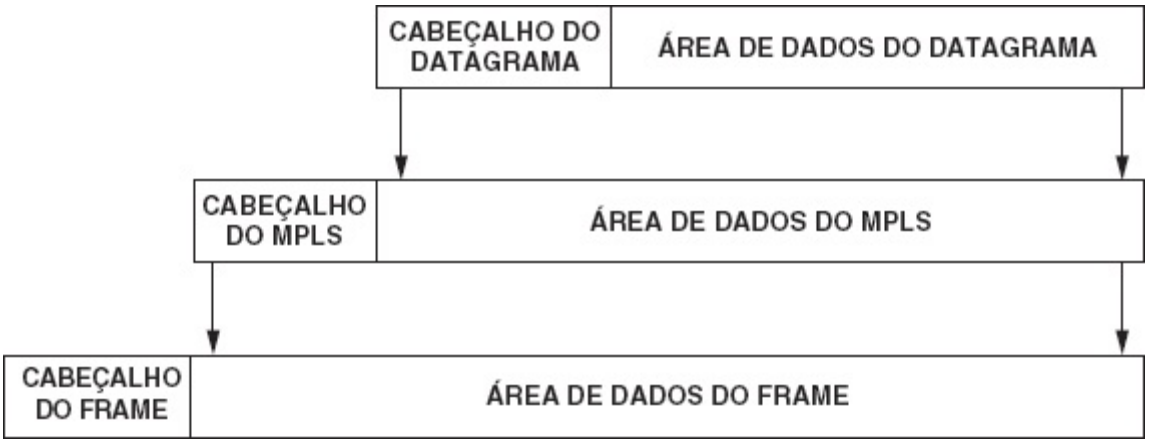


Figura 16.3 O encapsulamento usado com MPLS para enviar um datagrama IPv4 por uma rede convencional, tal como uma Ethernet. Um cabeçalho MPLS tem tamanho variável, e depende do número de entradas em uma pilha de rótulos.

Como exemplo, considere o uso de MPLS para enviar um datagrama através de uma rede Ethernet. Ao utilizar MPLS, o campo Tipo de Ethernet é definido para 0x8847 para transmissão unicast.** Assim, não há ambiguidade sobre o conteúdo de um frame – um receptor pode usar o tipo do frame para determinar se ele carrega MPLS ou um datagrama

convencional.

Um cabeçalho MPLS é de tamanho variável e é constituído por uma ou mais entradas, cada uma é de 32 bits de extensão e especifica um rótulo mais a informação usada para controlar o processamento deste. A Figura 16.4 ilustra o formato de uma entrada no cabeçalho.

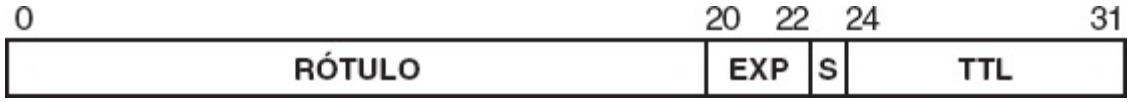


Figura 16.4 Ilustração dos campos em uma entrada de cabeçalho MPLS. Um cabeçalho MPLS consiste em uma ou mais dessas entradas.

Como a Figura 16.4 indica, não existe um campo para especificar o tamanho global de um cabeçalho MPLS, nem o cabeçalho contém um campo para especificar o tipo de carga útil. Para entender por que, lembre-se de que o MPLS é uma tecnologia de conexão orientada. Antes que uma estrutura MPLS possa ser enviada através de um único link, um caminho inteiro deve ser configurado. O roteador de comutação de rótulo ao longo do caminho deve ser configurado para saber exatamente como processar um pacote com um determinado rótulo. Portanto, quando o caminho MPLS é configurado, os dois lados chegam a um acordo sobre o tamanho da pilha de rótulo MPLS e os conteúdos da área de carga útil.

Uma entrada no cabeçalho MPLS começa com um campo de RÓTULO que o receptor usa para processar o pacote. Se o receptor for um salto intermediário ao longo de um caminho MPLS, ele realizará uma comutação de rótulo e continua. Se encontrar-se no limite entre dois níveis de uma hierarquia MPLS, o receptor irá remover o primeiro rótulo da pilha e usar o seguinte. Quando o pacote atingir o salto final ao longo de um caminho MPLS, o receptor irá remover o cabeçalho MPLS final e utilizar tabelas de encaminhamento IP convencional para lidar com o datagrama encapsulado.

O campo rotulado EXP em uma entrada de cabeçalho MPLS é reservado para uso experimental. O bit *S* é definido como 1 para indicar o fim da pilha (isto é, a última entrada no cabeçalho de um MPLS); em outras entradas, o bit *S* é 0. Finalmente, o campo TTL (*Time To Live*) é análogo ao campo TTL em um cabeçalho de datagrama IPv4: cada comutador ao longo do caminho que usa o rótulo para encaminhar o pacote diminui o valor TTL. Se o TTL chega a zero, o MPLS descarta o pacote. Assim, o MPLS impede que um pacote fique em ciclo para sempre, mesmo que um gerente confunda comutadores e acidentalmente crie um loop de encaminhamento.

16.10 Semântica de rótulo

Note-se que um rótulo MPLS é de 20 bits de largura. Em teoria, uma configuração MPLS pode usar todos os 20 bits do rótulo para acomodar até 2^{20} fluxos simultâneos (ou seja, 1.048.576 fluxos). Na prática, no entanto, as instalações MPLS raramente utilizam um grande número de fluxos, porque geralmente é necessário um gerente para autorizar e configurar cada percurso comutado.

A descrição de comutação anterior explica que um rótulo é usado como um índice em uma matriz. De fato, algumas implementações de comutação, especialmente implementações de hardware, usam um rótulo como um índice. No entanto, o MPLS não exige que cada rótulo corresponda a um índice de array. Em vez disso, as implementações MPLS permitem a um rótulo ser um inteiro arbitrário de 20 bits. Quando recebe um pacote de entrada, um roteador MPLS extrai o rótulo e executa uma pesquisa. Tipicamente, o mecanismo de pesquisa utiliza um *algoritmo hashing* (*hashing* algorithm), o que significa que a pesquisa é aproximadamente tão rápida como um índice de matriz.*

Permitir valores arbitrários em rótulos torna possível para os gerentes escolher os rótulos que fazem monitoramento e depuração mais fácil. Por exemplo, se houver três caminhos principais, através de uma rede corporativa, cada caminho pode ser atribuído a um prefixo 0, 1, ou 2, e o prefixo pode ser usado em cada salto ao longo do caminho. Se ocorrer um problema e um gerente de rede capturar os pacotes, ter bits no rótulo que identificam um caminho torna mais fácil para associar o pacote com um caminho.

16.11 Roteador por comutação de rótulo

Um roteador que implementa o MPLS é conhecido como um *Roteador por Comutação de Rótulo* (*Label Switching Router – LSR*). Normalmente, um LSR consiste em um roteador convencional que foi aumentado com software (e se possível hardware) para lidar com MPLS. Conceitualmente, o processamento MPLS e o processamento de datagrama convencional são completamente separados. Quando um pacote chega, o LSR usa o tipo do frame para decidir como processar o pacote, como a Figura 16.5 ilustra.

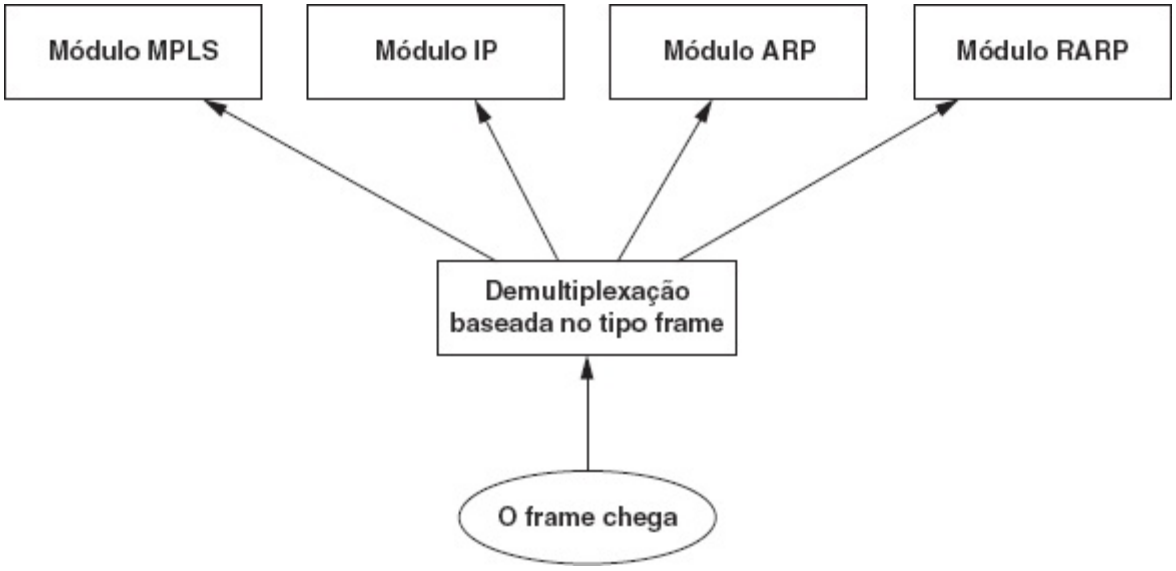


Figura 16.5 Demultiplexação do frame em um LSR que lida com ambos o MPLS e o encaminhamento IP convencional.

Na prática, ter ambos os MPLS e capacidade de IP em uma única LSR permite que o roteador sirva como interface entre a internet não MPLS e um centro MPLS. Ou seja, um LSR pode aceitar um datagrama de uma rede convencional, classificá-lo para atribuir um rótulo inicial MPLS e encaminhá-lo ao longo de um caminho MPLS. Da mesma forma, o LSR pode aceitar um pacote de mais de um caminho MPLS, remover o rótulo e encaminhar o datagrama em uma rede convencional. As duas funções são conhecidas como *ingresso MPLS* e *egresso MPLS*.

A tabela dentro de um LSR que especifica uma ação para cada rótulo é conhecida como Tabela de Encaminhamento de Rótulo de Próximo Salto (*Next Hop Label Forwarding Table*), e cada entrada nessa tabela é chamada de Entrada de Encaminhamento de Rótulo de Próximo Salto (*Next Hop Label Forwarding Entry - NHLFE*). Cada NHLFE especifica dois itens, e pode especificar mais três, a saber:

- informação de próximo salto (ou seja, a interface de saída);
- operação a ser executada;
- uso do encapsulamento (opcional);
- como codificar o rótulo (opcional);
- outras informações necessárias para lidar com o pacote (opcional).

Um NHLFE contém um campo de operação que especifica se o pacote

está atravessando uma transição para ou de um nível de hierarquia MPLS para outro ou está simplesmente sendo comutado ao longo de um caminho dentro de um único nível. As possibilidades são:

- substitua o rótulo no topo da pilha por um novo rótulo especificado, e continue a encaminhar via MPLS no nível atual de hierarquia;
- abra a pilha de rótulo para descer um nível de hierarquia MPLS. Use o próximo rótulo na pilha para encaminhar o datagrama, ou use o encaminhamento convencional se o rótulo final foi removido;
- substitua o rótulo no topo da pilha por um novo rótulo especificado, e depois coloque um ou mais novos rótulos na pilha para aumentar o nível (ou níveis) de hierarquia MPLS.

16.12 Processamento de controle e distribuição de rótulo

A discussão anterior se concentrou no processamento do *caminho de dados* (ou seja, o encaminhamento de pacotes). Além disso, os engenheiros que definiram o MPLS consideraram mecanismos para um *caminho de controle*. O processamento do caminho de controle refere-se à configuração e ao gerenciamento – os protocolos do caminho de controle tornam mais fácil para um gerente criar ou gerenciar um *Caminho de Rótulo Comutado* (*Label Switched Path – LSP*).

A principal funcionalidade fornecida por um protocolo de caminho de controle é a seleção automática de rótulos. Ou seja, os protocolos permitem que um gerente estabeleça um caminho MPLS sem configurar manualmente cada LSR ao longo do caminho. Os protocolos permitem que pares de LSRs ao longo do caminho escolham um rótulo não usado para o link entre um par e preenchem a informação do NHLFE para o fluxo, de modo que os rótulos possam ser trocados a cada salto.

O processo de escolher rótulos ao longo de um caminho é conhecido como *distribuição de rótulo*. Dois protocolos foram criados para realizar a distribuição de rótulo para MPLS: o *Protocolo de Distribuição de Rótulo* (*Label Distribution Protocol – LDP*), que às vezes é chamado de *MPLS-LDP*, e o *LDP de Roteamento Baseado em Restrição* (*Constraint-based Routing LDP – CR-LDP*). A distribuição de rótulos lida com a tarefa de garantir que os rótulos estão atribuídos de forma consistente; a extensão de

roteamento baseado em restrição do LDP lida com a tarefa de construir caminhos ao longo das rotas que combinam com um conjunto de restrições administrativas. Além disso, os protocolos existentes, como OSPF, BGP e RSVP, também foram estendidos para fornecer distribuição de rótulo. Embora reconhecendo a necessidade de um protocolo de distribuição de rótulo, o grupo de trabalho do IETF que desenvolveu o MPLS não especificou qualquer um dos protocolos como o padrão obrigatório.

16.13 MPLS e fragmentação

Leitores observadores podem ter imaginado que existe um problema potencial com o MPLS e a fragmentação. Na verdade, eles interagem de duas maneiras. Primeiro, dissemos que, ao enviar um datagrama por um LSP, o MPSL acrescenta um cabeçalho de quatro ou mais octetos. Isso pode ter uma consequência interessante, especialmente no caso em que as redes subjacentes tiverem o mesmo MTU. Por exemplo, considere um LSR de ingresso que conecta uma Ethernet usando encaminhamento IP convencional para um centro MPLS que usa comutação de rótulo. Se um datagrama chegar por um caminho não MPLS com tamanho exatamente igual à MTU Ethernet, a inclusão de um cabeçalho MPLS de 32 bits fará com que a carga resultante exceda a MTU Ethernet. Como resultado, o LSR ingresso fragmenta o datagrama original, acrescenta o cabeçalho MPLS a cada fragmento e transmite dois pacotes através do centro MPLS, em vez de um.

Uma segunda interação entre o MPLS e a fragmentação ocorre quando um LSR ingresso recebe fragmentos IP em vez de um datagrama completo. Quando o datagrama deixa o centro MPLS, um roteador precisa examinar os números de porta TCP ou UDP, assim como o endereço IP, para decidir como processar o pacote. Assim, um LSR ingresso deve coletar fragmentos e reconstituir o datagrama ou confiar somente nos endereços de origem e destino IP. Um grande ISP que usa MPLS tem duas escolhas. De um lado, pode exigir que clientes usem uma MTU menor do que a rede permite (ou seja, uma MTU Ethernet de 1492 octetos deixa espaço para duas entradas de cabeçalhos MPLS sem diminuição significativa no resultado geral). Por outro lado, um ISP pode simplesmente resolver proibir o uso de MPLS com fragmentos. Ou seja, nos casos em que o roteador egresso deve examinar campos transport layer, o roteador ingresso examina cada datagrama e descarta qualquer um que seja muito grande para enviar pelo caminho MPLS sem fragmentação.

16.14 Topologia em malha e engenharia de tráfego

É interessante que muitos ISPs que usam MPLS sigam uma técnica simples: eles definem uma *malha completa* (*full mesh*) de caminhos MPLS. Ou seja, se o ISP tiver K sites e conexões com J outros ISPs, o ISP define um caminho MPLS para cada par de pontos possível. Como resultado, o tráfego que se movimenta entre qualquer par de sites trafega por um único caminho MPLS entre os sites. Um efeito colateral benéfico de definir caminhos separados surge da capacidade de medir (ou controlar) o tráfego que atravessa de um site para outro. Por exemplo, observando uma única conexão MPLS, um ISP pode determinar quanto tráfego atravessa de um dos seus sites por uma conexão par para outro ISP.

Alguns sites estendem a ideia de uma malha completa, definindo vários caminhos entre cada par de sites para acomodar vários tipos de tráfego. Por exemplo, um caminho MPLS com um mínimo de saltos poderia ser reservado para VoIP, que precisa de um mínimo de atraso, enquanto um caminho mais longo poderia ser usado para outro tráfego, como e-mail e tráfego da web. Muitos LSRs fornecem um mecanismo para garantir a um determinado caminho MPLS uma porcentagem da rede subjacente. Dessa forma, um ISP pode especificar que um caminho MPLS transportando tráfego de voz sempre receba N por cento da capacidade da rede. A classificação MPLS torna possível o uso de uma variedade de medidas para escolher um caminho para dados, incluindo o endereço de origem IP assim como os números das portas protocolo.

> *Como a classificação MPLS pode usar campos arbitrários em um datagrama, incluindo o endereço de origem IP, o serviço que um datagrama recebe pode depender de o cliente enviá-lo e também do tipo de dado sendo transportado.*

Como uma alternativa à atribuição de um único caminho MPLS para cada fluxo agregado, o MPLS permite a um ISP balancear o tráfego entre dois caminhos disjuntos. Além disso, para garantir alta confiabilidade, o ISP pode organizar o uso de dois caminhos como backup mútuo – se um caminho falhar, todo o tráfego pode ser encaminhado pelo outro.

Usamos o termo *engenharia de tráfego* para caracterizar o processo de usar caminhos MPLS para direcionar o tráfego ao longo de rotas que

alcançam uma política da organização. Um gerente cria um conjunto de caminhos MPLS, especifica o LSR ao longo de cada caminho e define regras que atribuem um datagrama a um dos caminhos. Algumas facilidades de engenharia de tráfego permitem aos gerentes usarem algumas das técnicas de *Qualidade de Serviço* (*Quality of Service – QoS*) definidas no Capítulo 26 para controlar a taxa de tráfego em cada caminho. Assim, é possível definir dois fluxos MPLS por uma única conexão física e usar técnicas de QoS para garantir que, se cada fluxo tiver tráfego para enviar, 75% da capacidade da rede subjacente será dedicada a um fluxo e os 25% restantes para o outro.

16.15 Resumo

Para conseguir alta velocidade, as tecnologias de comutação usam a indexação no lugar da pesquisa do prefixo mais longo. Como consequência, a comutação segue um paradigma de conexão orientada. Como os comutadores ao longo de um caminho podem reescrever rótulos, o rótulo atribuído a um fluxo pode mudar ao longo do caminho.

O padrão para a comutação de datagramas IP, conhecido como MPLS, foi criado pelo IETF. Ao enviar um datagrama por um LSP, o MPLS inclui um cabeçalho no início, criando uma pilha de um ou mais rótulos; LSRs subsequentes ao longo do caminho utilizam os rótulos para encaminhar o datagrama sem realizar pesquisas de tabela de roteamento. Um LSR ingresso classifica cada datagrama que chega de um host ou roteador não MPLS, e um LSR egresso pode passar datagramas de um caminho MPLS para um host ou roteador não MPLS.

EXERCÍCIOS

16.1 Considere o formato de datagrama IPv6 descrito no Capítulo 7. Que mecanismos se relacionam diretamente ao MPLS?

16.2 Leia sobre MPLS. O MPLS deverá acomodar o encaminhamento da camada 2 (ou seja, bridging) além do encaminhamento IP otimizado? Sim ou não? Por quê?

16.3 Se todo o tráfego do host X atravessar uma hierarquia MPLS de dois níveis, que ação poderia ser tomada para garantir que não ocorrerá fragmentação?

16.4 Leia mais sobre Protocolo de Distribuição de Rótulo (Label Distribution Protocol – LDP) e extensão de Roteamento baseado em Restrição (Constraint-based Routing). Quais são as possíveis

restrições que podem ser usadas?

16.5 Se um roteador no seu site admitir MPLS, habilite a comutação MPLS e meça a melhoria de desempenho em relação à pesquisa da tabela de roteamento convencional. (Dica: certifique-se de medir muitos pacotes para um determinado destino, a fim de evitar que as medições sejam afetadas pelo custo de tratamento do pacote inicial.)

16.6 É possível conduzir uma experiência para determinar se o seu ISP utiliza MPLS? (Considere que é possível transmitir pacotes arbitrários.)

16.7 A Cisco Systems, Inc. oferece uma tecnologia de comutação conhecida como *Multi-Layer Switching (MLS)*. Leia sobre MLS. De que maneiras o MLS difere do MPLS?

16.8 Se o seu site tem um comutador VLAN que oferece serviço MLS, habilite-o e teste o que acontece se alguém enviar um frame Ethernet válido que contenha um datagrama IP incorreto. Um comutador de camada 2 deve examinar os cabeçalhos IP? Sim ou não? Por quê?

16.9 Assuma que é possível obter uma cópia de todos os frames que viajam por uma Ethernet. Como você sabe se um determinado frame é MPLS? Se você encontrar um frame MPLS, como pode determinar o tamanho de um cabeçalho MPLS?

---

* Embora o MPLS possa ser utilizado para o IPv6, a presença de uma etiqueta de fluxo no cabeçalho IPv6 reduz a necessidade de MPLS.

** O tipo Ethernet 0x8848 foi atribuído para uso quando o MPLS é multicast, mas o MPLS não lida com multicast muito bem.

* Na média, o hashing só exige uma pesquisa para encontrar a entrada correta em uma tabela se o fator de carga da tabela for lento.

CAPÍTULO 17

Classificação de pacote

CONTEÚDOS DO CAPÍTULO

17.1 Introdução

Os capítulos anteriores descrevem os sistemas tradicionais de processamento de pacotes e explicam dois conceitos fundamentais. Primeiro, vimos como cada camada de software de protocolo em um host ou roteador usa um campo de tipo em um cabeçalho de protocolo para demultiplexação. O campo de tipo em um frame é usado para selecionar um módulo de Camada 3 para lidar com o frame, o campo de tipo em um cabeçalho IP é usado para selecionar um módulo de protocolo de camada de transporte, e assim por diante. Em segundo lugar, vimos como o IP realiza o encaminhamento de datagramas, pesquisando o endereço de destino para selecionar um próximo salto.

Este capítulo tem uma visão totalmente diferente de processamento de pacotes daquela apresentada em capítulos anteriores. No lugar de demultiplexação, vamos considerar uma técnica conhecida como *classificação*. Em vez de assumir que um pacote prossegue através de uma pilha de protocolos uma camada de cada vez, vamos examinar uma técnica que atravessa as camadas.

A classificação de pacotes é pertinente a três tópicos abordados em outros capítulos. Primeiro, o capítulo anterior descreve o MPLS, e vamos ver que os roteadores usam classificação ao escolher um caminho MPLS pelo qual enviar um datagrama. Em segundo lugar, capítulos anteriores descrevem comutadores Ethernet, e vamos aprender que comutadores usam

classificação em vez de demultiplexação. Finalmente, o Capítulo 28 irá completar nossa discussão de classificação, introduzindo o tema importante de *Software Defined Networking (SDN)*. Veremos que a classificação forma a base das tecnologias de SDN, e entender como uma rede definida por software inclui conceitos de MPLS, bem como conceitos de comutadores Ethernet.

17.2 Motivação para classificação

Para entender a motivação para classificação, consideremos um roteador com software de protocolo organizado em uma pilha de camadas tradicional, como a Figura 17.1 ilustra.

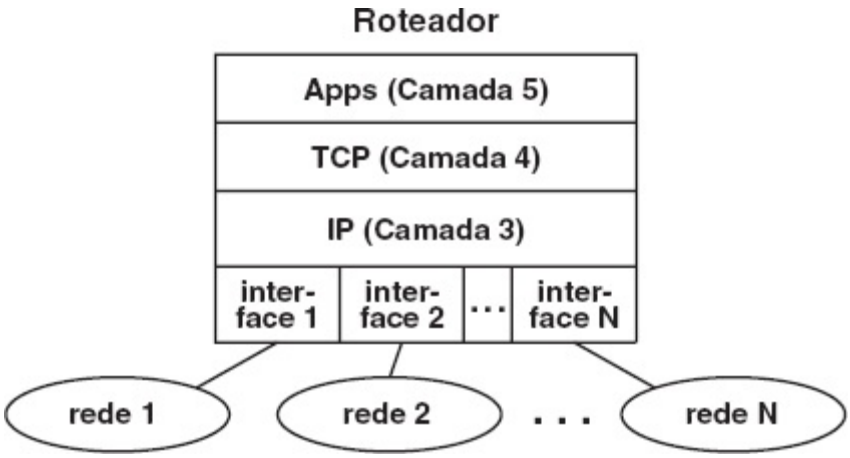


Figura 17.1 A pilha de protocolo em um roteador tradicional com camadas envolvidas em encaminhamento do trânsito do datagrama em destaque.

Como a figura indica, o encaminhamento de datagramas geralmente requer apenas protocolos até a Camada 3. O processamento de pacotes depende de *demultiplexação* em cada camada da pilha de protocolos. Quando um frame chega, o software de protocolo olha para o campo tipo para saber qual o conteúdo da carga útil do frame. Se ele leva um datagrama IP, a carga é enviada para o módulo de protocolo IP para o processamento. O IP utiliza o endereço de destino para selecionar um endereço de próximo salto. Se o datagrama está *em trânsito* (ou seja, passando pelo roteador a caminho de um destino), o IP encaminha o datagrama, enviando-o de volta para uma das interfaces. Um datagrama só alcança o TCP se for destinado ao próprio roteador.

Para entender o porquê de camadas tradicionais não resolverem todos os

problemas, considere processamento MPLS conforme descrito no capítulo anterior. Em particular, considere um roteador de borda entre a internet tradicional e um centro MPLS. Esse roteador deve aceitar pacotes que chegam da internet tradicional e escolher um caminho MPLS pelo qual enviar o pacote. Por que as camadas são pertinentes à seleção de caminho? Em muitos casos, os gerentes de rede usam números de porta de protocolo da camada de transporte ao escolher um caminho. Por exemplo, suponha que um gerente queira enviar todo o tráfego web para um caminho MPLS específico. Todo o tráfego da web vai usar a porta TCP 80, o que significa que a seleção deve examinar os números de portas TCP.

Infelizmente, em um esquema de demultiplexação tradicional, um datagrama não atinge a camada de transporte, a menos que seja destinado ao próprio roteador. Portanto, o software de protocolo deve ser reorganizado para lidar com seleção de caminho MPLS. Podemos fazer o resumo a seguir.

Uma pilha de protocolos tradicional é insuficiente para a tarefa de seleção de caminho MPLS, porque essa seleção muitas vezes envolve informações de camada de transporte e uma pilha tradicional não vai enviar datagramas de trânsito para a camada de transporte.

17.3 Classificação em vez de demultiplexação

Como poderia um protocolo de software ser estruturado para lidar com tarefas tais como seleção de caminho MPLS? A resposta está na tecnologia conhecida como *classificação*. Um sistema de classificação difere da demultiplexação tradicional de duas maneiras, que são:

- capacidade de cruzar múltiplas camadas;
- mais rapidez do que a demultiplexação.

Para entender classificação, imagine um pacote que foi recebido em um roteador e colocado na memória. Lembre-se de que encapsulamento significa que o pacote terá um conjunto de cabeçalhos de protocolo contíguos no início. Por exemplo, a Figura 17.2 ilustra os cabeçalhos em um pacote TCP (ou seja, uma requisição enviada a um servidor web) que chegou por uma Ethernet.

Figura 17.2 A organização de campos de cabeçalho de protocolo em um pacote TCP.

Dado um pacote na memória, como podemos determinar rapidamente se ele é destinado para a web? Uma abordagem simplista tão somente olha para um campo nos cabeçalhos: o número da porta TCP de destino. No entanto, pode ser que o pacote não seja na verdade um pacote de TCP. Talvez o frame esteja levando ARP em vez de IP. Ou talvez o frame contenha realmente um datagrama IP, mas, em vez de TCP, o protocolo de camada de transporte é UDP. Para ter certeza de que ele é destinado para a web, o software precisa verificar cada um dos cabeçalhos: o frame contém um datagrama IP, o datagrama IP contém um segmento TCP, e o segmento TCP é destinada para a web.

Em vez de cabeçalhos de protocolo de análise, pense no pacote como uma matriz de octetos na memória. Como exemplo, considere o IPv4.* Para ser um datagrama IPv4, o campo Tipo de Ethernet (localizado nas posições 12 a 13 da matriz) deve conter *0x0800*. O campo de protocolo IPv4, localizado na posição 23, deve conter *6* (o número de protocolo para TCP). O campo de porta de destino no cabeçalho TCP deve conter *80*. Para saber a posição exata do cabeçalho TCP, temos que saber o tamanho do cabeçalho IP. Portanto, verificamos o octeto de tamanho de cabeçalho do cabeçalho IPv4. Se o octeto contiver *0x45*, o número da porta TCP destino será encontrado nas posições de 36 a 37 da matriz.

Como outro exemplo, considere o tráfego *Voz sobre IP* (*VoIP*) que utiliza o *protocolo de transporte em tempo real* (*Real-Time Transport Protocol – RTP*). Porque RTP não está atribuído a uma porta UDP específica, os fornecedores empregam uma heurística para determinar se um determinado pacote transporta tráfego RTP: verificam a Ethernet e os cabeçalhos IP para averiguar se um pacote transporta UDP, e depois examinam os octetos em um offset conhecido na mensagem RTP para verificar que o valor no pacote coincide com o valor esperado por um codec conhecido.

Observe que todos os controles descritos nos parágrafos anteriores só exigem pesquisa na matriz. Ou seja, o mecanismo de pesquisa apenas olha para verificar que a localização *X* contém o valor *Y*, a localização *Z* contém o valor *W* e assim por diante – o mecanismo não precisa entender qualquer

um dos cabeçalhos de protocolo ou o significado dos octetos. Além disso, observe que o esquema de pesquisa atravessa várias camadas da pilha de protocolos.

Usamos o termo *classificador* para descrever um mecanismo que utiliza a abordagem de pesquisa descrita anteriormente, e dizemos que o resultado é uma *classificação* de pacotes. Na prática, um mecanismo de classificação geralmente leva uma lista de *regras* de classificação e as aplica até que seja encontrada uma correspondência. Por exemplo, um gerente pode especificar três regras: enviar todo o tráfego web para o caminho MPLS *1*, enviar todo o tráfego FTP para o caminho MPLS *2* e enviar todo o tráfego VPN para o caminho MPLS *3*.

17.4 Camadas quando se usa classificação

Se a classificação atravessa camadas de protocolo, como ela se relaciona com nossos diagramas de camadas vistos anteriormente? Pensamos em uma camada de classificação como uma camada extra que foi espremida entre a camada de interface de rede e a de IP. Uma vez que chega um pacote, ele passa do módulo de interface de rede para a camada de classificação. Todos os pacotes passam pelo classificador; nenhuma demultiplexação ocorre antes da classificação. Se qualquer uma das regras de classificação combina com o pacote, a camada de classificação segue a regra. Caso contrário, o pacote segue a pilha do protocolo tradicional. Por exemplo, a Figura 17.3 ilustra camadas quando a classificação é usada para enviar alguns pacotes através de caminhos MPLS.

Curiosamente, a camada de classificação pode incluir o primeiro nível de demultiplexação. Ou seja, em vez de apenas classificar os pacotes para caminhos MPLS, o classificador pode ser configurado com regras adicionais que verificam o campo de tipo em um frame para IP, ARP, RARP, e assim por diante.

17.5 Hardware de classificação e comutadores de rede

O que foi dito anteriormente descreve um mecanismo de classificação que é implementado em software – uma camada extra é adicionada a uma pilha de protocolos de software que classifica os frames, assim que chegam a um roteador. A classificação é também implementada em hardware. Em particular, comutadores Ethernet e outros dispositivos de hardware de processamento de pacotes contêm hardware de classificação que permite o

encaminhamento de pacotes para conseguir alta velocidade. As próximas seções explicam os mecanismos de classificação de hardware. O Capítulo 28 continua a discussão mostrando como as tecnologias de rede definida por software usam o mecanismo de classificação em comutadores para conseguir engenharia de tráfego em alta velocidade.

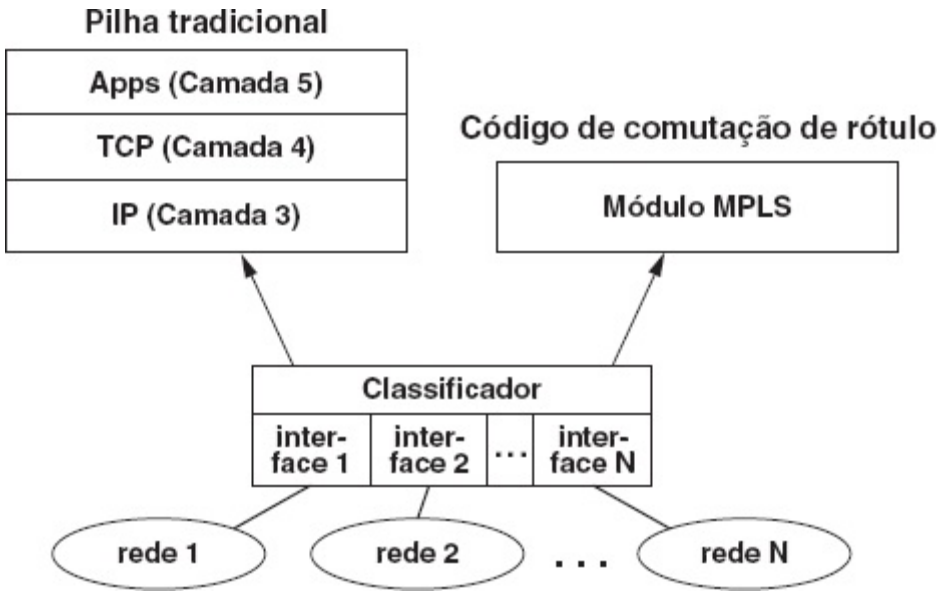


Figura 17.3 Camadas quando um roteador usa classificação para selecionar caminhos MPLS.

Pensamos em dispositivos de rede, por exemplo, comutadores, como divididos em categorias amplas pelo nível de cabeçalhos de protocolo que eles examinam e o consequente nível de funcionalidade que eles fornecem.

- Comutação de camada 2.
- Comutação de camada 2 VLAN.
- Comutação de camada 3.
- Comutação de camada 4.

O Capítulo 2 descreve comutadores da Camada 2. Em essência, esse comutador examina o endereço MAC de origem em cada frame de entrada para saber o endereço MAC do computador que está conectado a cada porta. Uma vez que um comutador aprende os endereços MAC de todos os computadores conectados, pode usar o endereço MAC de destino em cada frame para tomar uma decisão de encaminhamento. Se o frame é unicast, o comutador somente envia uma cópia do frame na porta à qual o computador especificado está conectado. Para um frame destinado à

transmissão ou um endereço de multicast, o comutador entrega uma cópia do frame para todas as portas.

Um *comutador* VLAN permite que o gerente atribua cada porta a uma VLAN específica. Comutadores VLAN estendem encaminhamento de uma forma mínima: em vez de enviar broadcasts e multicasts para todas as suas portas, consultam a configuração da VLAN e enviam para as portas dela como origem.

Um *comutador de Camada 3* age como uma combinação de um comutador VLAN e um roteador. Em vez de usar apenas o cabeçalho Ethernet ao encaminhar um frame, o comutador pode olhar para campos no cabeçalho IP. Em particular, o comutador observa o endereço IP de origem em pacotes de entrada para saber o endereço IP do computador conectado a cada porta do comutador. Este pode, então, usar o endereço de destino IP em um pacote para encaminhá-lo para seu destino correto.

Um *comutador de Camada 4* estende a análise de um pacote para a camada de transporte. Ou seja, o comutador pode incluir o TCP ou campos porta de origem e destino UDP ao tomar uma decisão de encaminhamento.

17.6 Decisões de comutação e VLAN tags

Todos os tipos de comutação de hardware dependem de classificação. Isto é, os comutadores funcionam em pacotes, como se um pacote fosse simplesmente uma matriz de octetos, e campos individuais no pacote são especificados dando offsets na matriz. Assim, em vez de pacotes de demultiplexação, um comutador trata um pacote sintaticamente pela aplicação de um conjunto de regras de classificação semelhantes com as regras descritas anteriormente

Surpreendentemente, mesmo processamento de VLAN é tratado de uma forma sintática. Em vez de simplesmente manter a informação de VLAN em uma estrutura de dados em separado que guarda metainformações, o comutador insere um campo extra em um pacote de entrada e coloca o número da VLAN do pacote no campo extra. Como ele é apenas mais um campo, o classificador pode referenciar o número da VLAN exatamente como qualquer outro campo de cabeçalho.

Nós usamos o termo *VLAN tag* como referência ao campo suplementar inserido em um pacote. A etiqueta contém o número da VLAN que o gestor atribuiu à porta através da qual o frame chegou. Para Ethernet, o padrão IEEE 802.1Q especifica colocar o campo VLAN tag após o campo de endereço MAC de origem. A Figura 17.4 ilustra o formato.

Figura 17.4 Ilustração de um campo VLAN tag IEEE 802.1Q inserido em um frame Ethernet depois que um frame chega a um comutador VLAN.

Um VLAN tag só é usado internamente – uma vez que o comutador tenha selecionado uma porta de saída e esteja pronto para transmitir o frame, a etiqueta é removida. Assim, quando computadores ligados a um comutador enviam e recebem frames, estes não contêm uma VLAN tag.

Uma exceção pode ser feita para a regra: um gerente pode configurar uma ou mais portas em um comutador para deixar VLAN tags em frames ao enviá-los. O objetivo é o de permitir que dois ou mais comutadores sejam configurados para funcionar como um único e grande comutador. Ou seja, os comutadores podem compartilhar um conjunto de VLANs – um gerente pode configurar cada VLAN para incluir portas em um ou ambos os comutadores.

17.7 Hardware de classificação

Podemos pensar em hardware em um comutador como dividido em três componentes principais: um classificador, um conjunto de unidades que executam ações e um componente de gestão que controla a operação global. A Figura 17.5 ilustra a organização geral e o fluxo de pacotes.

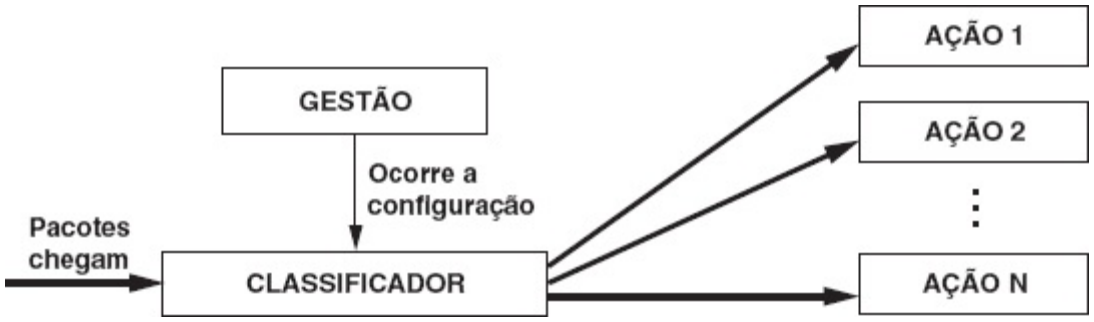


Figura 17.5 A organização conceitual do hardware em um comutador.

Como a figura indica, o classificador proporciona o caminho de dados de alta velocidade que os pacotes seguem. Um pacote chega e o classificador utiliza as regras que foram configuradas para escolher uma ação. O módulo de gerenciamento normalmente consiste em um

processador de uso geral que executa software de gestão. Um gerente de rede pode interagir com o módulo de gerenciamento para configurar o comutador, caso em que o módulo de gestão pode criar ou modificar o conjunto de regras que o classificador segue.

Tal como acontece com a classificação MPLS, um comutador deve ser capaz de lidar com dois tipos de tráfego: tráfego de trânsito e tráfego destinado ao próprio comutador. Por exemplo, para fornecer funções de gestão ou de roteamento, um comutador pode ter uma pilha de protocolos TCP/IP local, e pacotes destinados para o comutador devem ser passados para a pilha local. Por isso, uma das ações que um classificador toma pode ser *passar o pacote para a pilha local para demultiplexação.*

17.8 Classificação de alta velocidade e TCAM

Comutadores modernos podem permitir que cada interface opere a 10 Gbps. Nessa velocidade, um frame leva apenas 1,2 microssegundos para chegar, e um comutador normalmente tem muitas interfaces. Um processador convencional não consegue lidar com a classificação em tais velocidades. Então surge a pergunta: como um classificador hardware atinge alta velocidade? A resposta está em uma tecnologia de hardware conhecida como *memória de endereçamento de conteúdo ternário* (*Ternary Content Addressable Memory – TCAM*).

A TCAM usa paralelismo para alcançar alta velocidade – em vez de testar um campo de um pacote em um determinado momento, a TCAM verifica todos os campos ao mesmo tempo. Além disso, a TCAM realiza múltiplas verificações simultaneamente. Para entender como funciona a TCAM, pense em um pacote como uma sequência de bits. Imaginamos o hardware TCAM como tendo duas partes: uma parte mantém os bits de um pacote e a outra é uma matriz de valores que vão ser comparados aos pacotes. Entradas na matriz são conhecidas como *slots*. A Figura 17.6 ilustra a ideia.

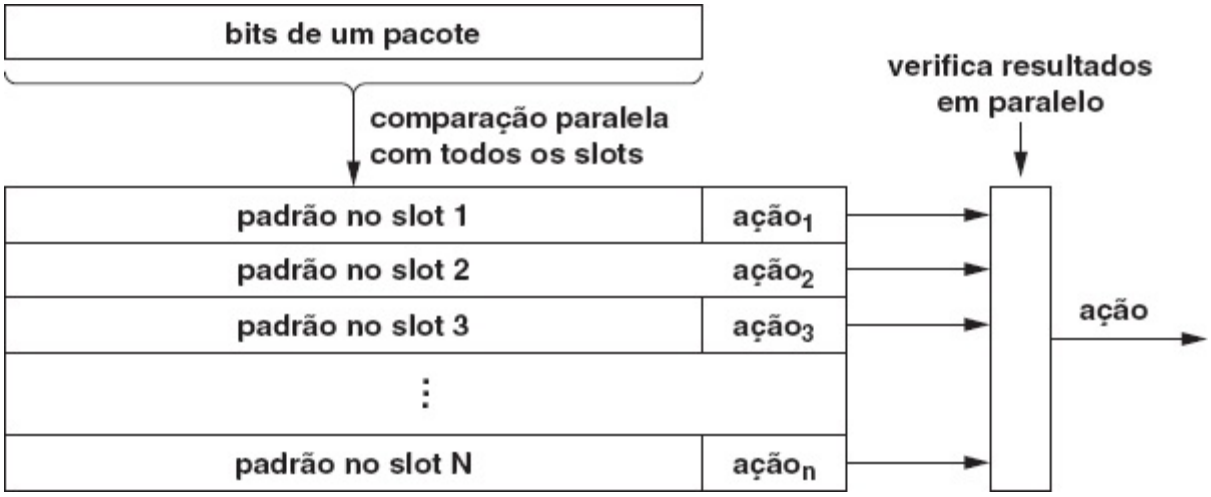


Figura 17.6 A organização conceitual de um classificador de hardware de alta velocidade que usa tecnologia TCAM.

Na figura, cada slot contém duas partes. A primeira parte consiste em hardware que compara os bits do pacote para o padrão armazenado no slot. A segunda parte armazena um valor que especifica uma ação a ser tomada se o padrão corresponde ao pacote. Se ocorrer uma correspondência, o slot de hardware passa a ação para o componente que verifica todos os resultados e anuncia uma resposta.

Um dos detalhes mais importantes diz respeito à forma como a TCAM lida com várias combinações. Em essência, o circuito de saída seleciona uma combinação e ignora as outras. Ou seja, se cada um dos vários slots passar uma ação para o circuito de saída, este só aceita uma e a passa como sendo a saída da classificação. Por exemplo, o hardware pode escolher o menor slot que combina. Em qualquer caso, a *ação* que a TCAM anuncia corresponde à ação de um dos slots que combinam.

A figura indica que um slot tem um *padrão* em vez de um valor exato. Em vez de se limitar a comparar cada bit do padrão com o correspondente bit no pacote, o hardware realiza uma correspondência de padrão. O adjetivo *ternário* é usado porque cada posição de bit em um padrão tem três valores possíveis: um, zero ou "não importa". Quando um slot compara o seu padrão com o do pacote, o hardware verifica apenas os bits um e zero no padrão – o hardware ignora os bits-padrão que contêm "não importa". Assim, um padrão pode especificar valores exatos para alguns campos em um cabeçalho do pacote e omitir outros campos.

Para entender a correspondência no padrão TCAM, considere um padrão que identifica os pacotes IP. Identificar esses pacotes é fácil, porque um

frame Ethernet que carrega um datagrama IP terá o valor 0x0800 no campo Tipo de Ethernet. Além disso, o campo tipo ocupa uma posição fixa no frame: bits de 96 a 111. Assim, podemos criar um padrão que comece com 96 "não importa" bits (para atender os endereços MAC de destino e origem) seguidos por dezesseis bits com o valor binário 0000100000000000 (o binário equivalente de 0x0800) para atender ao campo tipo. Todas as posições de bit remanescentes no padrão serão "não importa". A Figura 17.7 ilustra o padrão e pacotes de exemplo.

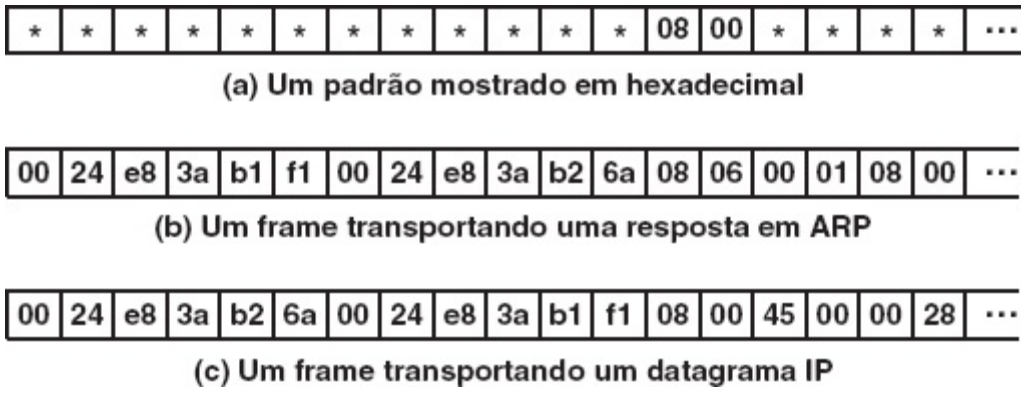


Figura 17.7 (a) Um padrão em um TCAM com asteriscos indicando "não importa", (b) um pacote ARP que não combina com o padrão e (c) um datagrama IP que combina com o padrão.

Embora um slot de hardware TCAM tenha uma posição para cada bit, a figura não exibe bits individuais. Em vez disso, cada caixa corresponde a um octeto, e o valor em uma caixa é um valor hexadecimal que corresponde a oito bits. Usamos hexadecimal simplesmente porque sequências (strings) binárias são muito longas para caber confortavelmente em uma página impressa.

17.9 O tamanho de uma TCAM

Surge uma questão: que tamanho tem uma TCAM? A questão pode ser dividida em dois aspectos importantes:

- o número de bits em um slot;
- o número total de slots.

Bits por slot. O número de bits por slot depende do tipo de comutador de Ethernet. Um comutador básico usa o endereço MAC de destino para classificar um pacote. Portanto, a TCAM em um comutador básico só

precisa de 48 posições de bit. Um comutador VLAN precisa de 128 posições de bit para cobrir a VLAN tag como a Figura 17.4 ilustra.* Um comutador de Camada 3 deve ter posições de bits suficientes para cobrir o cabeçalho IP, bem como o cabeçalho da Ethernet.

Total de slots. O número total de slots TCAM determina o número máximo de padrões que um comutador pode suportar. A TCAM em um comutador típico tem 32.000 entradas. Quando um comutador aprende o endereço MAC de um computador que foi conectado a uma porta, pode armazenar um padrão para o endereço. Por exemplo, se um computador com endereço MAC *X* é conectado na porta 29, pode criar um padrão no qual os bits de endereço de destino combinam com *X* e a ação é *envie o pacote para a porta de saída 29.*

Um comutador também pode usar padrões para controlar broadcasting. Quando um gerente configura uma VLAN, o comutador pode adicionar uma entrada para a broadcast VLAN. Por exemplo, se um gerente configura VLAN 9, uma entrada pode ser adicionada em que os bits de endereço de destino sejam todos 1 (ou seja, o endereço de broadcast Ethernet) e a VLAN tag seja 9. A ação associada com a entrada é transmitida por *broadcast na VLAN 9.*

Um comutador de Camada 3 pode aprender o endereço de origem IP dos computadores conectados ao comutador, e pode usar TCAM para armazenar uma entrada para cada endereço IP. Da mesma forma, é possível criar entradas que correspondam a números de porta de protocolo da Camada 4 (por exemplo, para direcionar todo o tráfego web para uma saída específica). O Capítulo 28 considera um outro uso interessante de hardware de classificação: um gerente pode colocar padrões no classificador para estabelecer caminhos através de uma rede e direcionar o tráfego ao longo dos caminhos. Como tais regras de classificação atravessam camadas múltiplas da pilha de protocolos, o número potencial de itens armazenados em uma TCAM pode ser grande.

A TCAM parece ser um mecanismo ideal porque é, ao mesmo tempo, extremamente rápida e versátil. No entanto, a TCAM tem uma desvantagem significativa: o custo. Além disso, uma vez que funciona em paralelo, consome muito mais energia do que a memória convencional. Por isso, os desenvolvedores minimizam a quantidade de TCAM para manter os custos e o consumo de energia baixos.

17.10 Encaminhamento generalizado de classificação habilitada

Talvez a vantagem mais significativa de um mecanismo de classificação decorra das generalizações que ele permite. Como a classificação examina campos arbitrários em um pacote antes de ocorrer qualquer demultiplexação, combinações de cross-layer são possíveis. Por exemplo, a classificação pode especificar que todos os pacotes de um determinado endereço MAC devem ser encaminhados para uma porta de saída específica, independentemente do tipo ou do conteúdo do pacote. Além disso, a classificação pode tornar as decisões de encaminhamento dependentes do endereço de origem em um pacote, assim como do endereço de destino. Um ISP pode usar endereços de origem para distinguir entre os clientes. Por exemplo, um ISP pode encaminhar todos os pacotes com endereço de origem IP X ao longo de um caminho, enquanto encaminha pacotes com endereço de origem IP Y ao longo de outro caminho, mesmo que todos os pacotes tenham o mesmo endereço de destino.

Os ISP utilizam a generalidade que a classificação oferece para lidar com a engenharia de tráfego que não está geralmente disponível em uma pilha de protocolo convencional. Em particular, a classificação permite aos provedores oferecer serviços diferenciados. Um ISP pode organizar a classificação de pacotes usando tanto o tipo de tráfego como a quantidade paga por um cliente. Uma vez que o pacote tenha sido classificado, todos os pacotes com a mesma classificação podem ser enviados ao longo do caminho apropriado.

17.11 Resumo

A classificação é uma otimização de desempenho fundamental que permite que um sistema de processamento de pacotes atravesse as camadas da pilha de protocolos sem demultiplexação. Um classificador trata cada pacote como uma matriz de bits e verifica o conteúdo dos campos em locais específicos na matriz.

A classificação é utilizada com MPLS, bem como em comutadores Ethernet. A maioria dos comutadores Ethernet implementa a classificação em hardware; uma tecnologia de hardware conhecida como TCAM usa paralelismo para executar classificação em altíssima velocidade.

EXERCÍCIOS

17.1 Leia sobre comutadores Ethernet e encontre o tamanho da TCAM usado.

17.2 Se o seu site utiliza MPLS, faça uma lista das regras de classificação que são utilizadas e declare o propósito de cada uma.

17.3 Se um comutador da Camada 2 tem P portas que se conectam a computadores, qual o número máximo de endereços de destino MAC que o comutador precisa para colocar em seu classificador? Tenha cuidado, porque a resposta não é óbvia.

17.4 Escreva regras de classificação que enviem todo o tráfego de VoIP por um caminho MPLS 1, o tráfego da web por um caminho MPLS 2, o tráfego *ssh* por um caminho MPLS 3 e todos os outros tráfegos por um caminho MPLS 4.

17.5 Um gerente quer enviar todo o tráfego multicast por um caminho MPLS 17 e todos os outros tráfegos por um caminho MPLS 18. Qual o conjunto mais simples de regras de classificação que pode ser usado?

17.6 Será que a sua resposta para a pergunta anterior muda se um site usa tanto o IPv4 quanto o IPv6?

17.7 O texto afirma que a classificação é necessária para processar os pacotes que chegam a 10 Gbps, porque um frame Ethernet leva apenas 1,2 microssegundos para chegar. Quantos bits têm na carga útil de tal estrutura?

17.8 No problema anterior, quantas instruções pode um processador de alta velocidade executar em 1,2 microssegundos?

17.9 Na maioria das redes, os menores pacotes contêm um ACK TCP viajando em um datagrama IPv4. Quanto tempo leva para chegar um datagrama em tal estrutura?

---

* Usamos IPv4 para manter os exemplos pequenos; embora os conceitos se apliquem para o IPv6, os cabeçalhos de extensão complicam os detalhes.

* Figura 17.4 pode ser encontrada na página 255.

CAPÍTULO 18

Mobilidade e IP móvel

CONTEÚDOS DO CAPÍTULO

18.1 Introdução

Os capítulos anteriores descrevem os esquemas de endereçamento e encaminhamento IP usados com computadores de mesa e um esquema endereçamento e encaminhamento que usa endereçamento baseado em rede.

Este capítulo considera uma tecnologia que permite a um computador portátil se mover de uma rede para outra. Vamos ver que a extensão pode trabalhar com redes com fio ou sem fio (wireless), tem versões que se aplicam ao IPv4 ou ao IPv6 e retém retrocompatibilidade com roteamentos de internet existentes.

18.2 Mobilidade, endereçamento e roteamento

No sentido mais geral, o termo *computação móvel* se refere a um sistema que permite que computadores se movam de um local para outro. Embora tecnologias sem fio possibilitem mobilidade rápida e fácil, não é exigido acesso sem fio – um viajante pode levar um computador laptop e se conectar a uma rede remota com fio (por exemplo, em um hotel).

O esquema de endereçamento IP, que foi projetado e otimizado para hosts estacionários, torna a mobilidade difícil. Um prefixo de cada endereço de host identifica a rede à qual o host se conecta, e roteadores usam o prefixo para encaminhar datagramas para a rede correta para entrega final.

Como resultado, mover um host para uma nova rede exige uma de duas mudanças possíveis descritas a seguir.

- O endereço do host deve mudar.
- O encaminhamento do datagrama deve mudar.

18.3 Mobilidade via mudança de endereço de host

A abordagem de mudar o endereço IP do host é amplamente utilizada na Internet global e funciona bem para a mobilidade lenta, semipermanente. Por exemplo, considere um usuário que leva um computador para um café e permanece por algum tempo tomando café enquanto usa a conexão Wi-Fi da loja. Ou considere um viajante que carrega um computador para um quarto de hotel, trabalha no hotel por dois dias e depois volta para casa. Como veremos no Capítulo 22, essa mobilidade é ativada com a *atribuição de endereço dinâmico*. Em particular, os hosts IPv4 usam o protocolo DHCP para obter um endereço IP e hosts IPv6 usam o protocolo de *Descoberta de Vizinho* IPv6 para gerar e verificar um endereço único.

A maioria dos sistemas operacionais faz a atribuição de endereços automaticamente sem informar ao usuário. Há duas condições que desencadeiam a aquisição de endereço dinâmico. A primeira, quando ele inicia, um host IPv4 sempre executa o DHCP e um host IPv6 gera um endereço unicast e valida a singularidade. A segunda, um sistema operacional atribui novamente um endereço quando detecta a perda e reaquisição da conectividade de rede. Assim, se um computador portátil permanece em execução enquanto ele é movido de um hotspot Wi-Fi para o outro, o sistema operacional irá detectar a desconexão da primeira rede Wi-Fi e a reconexão na segunda.

Embora ele funcione bem para usuários casuais, mudar o endereço de um host tem desvantagens. Uma mudança de endereço rompe todas as conexões da camada de transporte em curso. Por exemplo, uma conexão de transporte é usada para assistir a uma transmissão de vídeo ou para usar uma VPN. Em cada caso, alterar o endereço IP de um host quebra todas as conexões da camada de transporte e faz com que o sistema operacional informe os aplicativos que estão usando as conexões. Um aplicativo pode recuperar-se de perda de conexão, informando o usuário ou reiniciando a conexão automaticamente. Mesmo se um aplicativo reinicia uma conexão, reinício às vezes leva algum tempo, o que significa que um usuário pode notar uma interrupção no serviço.

Se um host oferece serviços de rede (ou seja, roda servidores), mudar um endereço IP tem consequências mais graves. Normalmente, cada aplicativo que executa um serviço deve ser reiniciado. Além disso, os computadores que rodam os serviços são geralmente atribuídos a um nome de domínio. Assim, quando o endereço IP do computador muda, a entrada DNS do host também deve ser atualizada.* Claro, um computador arbitrário não tem permissão para alterar uma entrada de DNS, o que significa que é necessária infraestrutura adicional para autenticar atualizações de DNS.

O ponto é:

> *Embora a atribuição de endereço dinâmico permita uma forma básica de mobilidade que possibilita ao usuário mover um host de uma rede para outra, mudar o endereço de um host tem a desvantagem de interromper conexões da camada de transporte.*

18.4 Mobilidade via mudança no encaminhamento de datagramas

Podemos permitir que um host mantenha o seu endereço IP original quando se move para uma nova rede? Em teoria, a resposta é sim – tudo o que precisamos fazer é mudar as tabelas de encaminhamento em roteadores em toda a Internet de tal forma que os datagramas destinados ao host serão encaminhados para a nova rede. Poderíamos até mesmo criar hardware de rede que detecte a presença de novos endereços IP e informe o sistema de roteamento sobre a sua presença.

Infelizmente, o esquema simplista descrito acima é impraticável porque o roteamento de um host específico não escala para o tamanho da Internet global. O roteamento da Internet só funciona porque os protocolos de roteamento trocam informações sobre redes, em vez de hosts, e porque as redes são estacionárias. Ou seja, o tamanho total da informação de roteamento é limitado e as informações são relativamente estáticas. Se os protocolos de roteamento são usados para lidar com hosts em vez de redes, a quantidade de tráfego de roteamento torna-se esmagador, mesmo que apenas uma pequena fração dos hosts mude de local a cada dia. O ponto importante é descrito a seguir.

> *Nós não podemos usar roteamento de host específico para lidar com mobilidade porque a Internet global não tem capacidade suficiente*

para propagar rotas de hosts específicos que mudam com frequência.

18.5 A tecnologia do IP móvel

O IETF criou uma tecnologia para permitir a mobilidade do IP; versões estão disponíveis tanto para o IPv4 como para o IPv6.* Oficialmente denominada *suporte à mobilidade IP*, a tecnologia fornece um compromisso. O IP móvel tem as vantagens de não mudar um endereço IP de host e não exigir roteamento de host específico, mas a desvantagem do encaminhamento de datagrama pode ser a ineficiência. As características gerais incluem as propriedades a seguir.

- *Transparência.* A mobilidade é transparente às aplicações, aos protocolos da camada de transporte e aos roteadores. Em particular, uma conexão TCP pode sobreviver a uma mudança de local. A única condição é que, se a transição levar muito tempo (ou seja, um host permanecer desconectado de todas as redes por um tempo), a conexão não poderá ser usada durante a transição. A razão é que o TCP irá desligar a conexão após no máximo dois segmentos de tempo de espera (lifetimes).
- *Retrocompatibilidade.* Um host usando IPv4 móvel pode interoperar com hosts estacionários que executam o soft ware IPv4 convencional, bem como com outros hosts IPv4 móveis. De forma similar, um host usando IPv6 mó vel pode interoperar com hosts estacionários que usam o IPv6 ou outros hosts IPv6 móveis. Ou seja, um host móvel usa o mesmo esquema de endereço IP que um host estacionário.
- *Escalabilidade.* A solução permite mobilidade através da Internet global.
- *Segurança.* O IP móvel pode garantir que todas as mensagens sejam autenticadas (isto é, para evitar que um computador arbitrário finja ser um host móvel).
- *Macromobilidade.* Em vez de tentar lidar com movimento rápido contínuo, tal como telefone celular em um carro, o IP móvel se concentra no problema das mudanças de longa duração (por exemplo, um usuário que leva um dispositivo portátil em uma viagem de negócios).

18.6 Visão geral da operação do IP móvel

Como pode um IP móvel permitir que um host retenha seu endereço sem exigir que os roteadores aprendam rotas específicas do host? O IP móvel resolve o problema permitindo que um host armazene dois endereços simultaneamente: um *endereço primário* permanente e fixo, usado pelas aplicações, e um *endereço secundário*, que é temporário e associado com uma determinada rede à qual o host está conectado. Um endereço temporário só é válido para um local; quando se move para outra localização, um host móvel deve obter um novo endereço temporário.

Supõe-se que um host móvel tenha uma *casa* permanente na Internet, e um endereço primário de host móvel é o endereço que o host teve atribuído à sua casa na rede. Além disso, para suportar a mobilidade, a rede da casa do host deve incluir um sistema de rede especial conhecido como um *agente de casa*. Normalmente, o software agente de casa é executado em um roteador convencional, mas isso não é estritamente necessário. Em essência, um agente de casa concorda em interceptar cada datagrama que chega para o endereço permanente do host e encaminhar o datagrama para a atual localização do host (seções posteriores discutem detalhes).

Como é que um agente de casa sabe a localização atual de um host móvel? Depois que se move para uma rede *externa*, um host móvel deve obter um endereço secundário (ou seja, temporário) e entrar em contato com seu agente de casa para informá-lo sobre sua localização atual. Dizemos que o host móvel *registra* um endereço secundário com o seu agente de casa.

O endereço secundário só é válido enquanto um host móvel permanece em um determinado local. Se o host se move novamente, ele deve obter um novo endereço secundário para o novo local e informar o agente de casa da mudança. Finalmente, quando um host móvel retorna para casa, ele deve contatar o agente de casa para *cancelar o registro*, o que significa que o agente vai parar de interceptar datagramas. Na verdade, um host móvel pode optar por cancelar o registro antes de retornar para casa (por exemplo, quando sai de um local remoto).

18.7 Sobrecarga e frequência de mudança

Dissemos que a tecnologia IP móvel é projetada para suportar *macromobilidade*. Em particular, o IP móvel não é indicado para

mudança de rede contínua de alta velocidade associada a um telefone inteligente que está sendo usado em um carro que se move por uma estrada. Assim, pensamos em um viajante usando IP móvel uma vez que chegar a um novo destino, em vez de em cada ponto ao longo da viagem.

A razão por que o IP móvel não suporta mudanças rápidas deveria ser óbvia: custo. Redes na Internet não monitoram dispositivos ou rastreiam seus movimentos. Mais importante, os sistemas de rede não se coordenam para realizar hand-off. Em vez disso, cada dispositivo móvel deve monitorar sua conexão de rede e detectar quando ele passou de uma rede para outra. Quando ele detecta a troca, o dispositivo móvel deve se comunicar através da rede externa para requisitar um endereço secundário para a rede. Uma vez que obteve o endereço secundário, deve comunicar ao seu agente de casa para registrar o endereço e estabelecer encaminhamento. Note que um dispositivo móvel pode estar arbitrariamente longe de sua rede de casa, o que significa que o registro pode envolver comunicação por distância arbitrária. O principal ponto é descrito a seguir.

Como exige sobrecarga considerável após cada alteração, IP móvel é destinado a situações em que um host se move com pouca frequência e permanece em um determinado local por um período relativamente longo de tempo (por exemplo, horas ou dias).

Os detalhes de endereçamento, registro e encaminhamento vão ficar claros conforme entendermos a tecnologia. Primeiro consideramos a mobilidade para o IPv4, que ilustra o conceito básico, e então consideramos por que tanta complexidade adicional é necessária par suportar o IPv6.*

18.8 Endereçamento IPv4 móvel

Ao utilizar IPv4, o endereço primário de um host móvel ou *endereço de casa* é um endereço IPv4 convencional que é atribuído e administrado como de costume. Aplicativos em um host móvel sempre utilizam o endereço primário; eles continuam ignorando qualquer outro endereço. O endereço secundário do host, que também é conhecido como *care-of address*, é um endereço temporário que é usado somente pelo software IP móvel no host. Um care-of address é válido apenas para uma determinada rede externa.

O IPv4 móvel suporta dois tipos de care-of address que diferem no método pelo qual o endereço é obtido e no modo que o encaminhamento ocorre:

- coalocado;
- agente externo.

Endereço care-of coalocado IPv4. Um endereço coalocado permite que um computador móvel lide com todo o encaminhamento e o tunelamento de datagrama sem qualquer assistência de hosts ou roteadores na rede externa. Na verdade, do ponto de vista de sistemas na rede externa, o host móvel parece ser um host convencional que segue padrões normais para obter, usar e então abandonar um endereço local. O endereço temporário é atribuído via DHPC como qualquer outro endereço.

A vantagem principal de endereço coalocado vem de sua aplicabilidade universal. Como os hosts móveis lidam com todos os detalhes de registro e comunicação, não são necessárias facilidades adicionais na rede externa. Assim, um host móvel pode usar endereçamento care-of coalocado em uma rede arbitrária, inclusive um hotspot Wi-Fi convencional tal como os encontrados em um café.

Existem duas desvantagens do método coalocado. Primeiro, a coalocação exige software extra no host móvel. Segundo, a rede externa não distingue um host que usa IP móvel de um visitante arbitrário. Vamos ver que a incapacidade de identificar um host que usa IP móvel pode afetar o encaminhamento.

Endereço care-of de agente externo IPv4. O segundo tipo de endereço temporário permite que uma rede externa saiba se um host está usando IP móvel porque um sistema nela participa de todos os desvios. O sistema é conhecido como um *agente externo*, e um endereço temporário utilizado com o esquema é conhecido como *endereço de agente externo care-of (foreign agent care-of address).* Para utilizar o método de agente externo, um host móvel não obtém um endereço local propriamente dito. Em particular, um host móvel *não* usa DHCP. Em vez disso, quando um host móvel chega a um site externo, ele utiliza um protocolo de descoberta para obter a identidade de um agente externo. O host móvel se comunica com o agente para aprender o endereço care-of a usar. Surpreendentemente, um agente externo não precisa atribuir a cada host móvel um único endereço. Em vez disso, o endereço do agente externo é o endereço IPv4. O agente,

então, entrega os datagramas de entrada para o host móvel visitante correto.

18.9 Descoberta do agente externo IPv4

O processo da *descoberta do agente externo* IPv4 utiliza o mecanismo de *descoberta de roteador* ICMP, em que cada roteador envia periodicamente uma mensagem ICMP de *anúncio de roteador* e permite que um host envie uma mensagem ICMP *solicitação de roteador* para pedir um anúncio.* Um roteador que atue como um anexo a uma *extensão de agente de mobilidade*** a cada mensagem; a extensão especifica o prefixo da rede, que o host móvel usa para determinar que se mudou para uma nova rede. Curiosamente, extensões de mobilidade não usam um tipo de mensagem ICMP em separado. Em vez disso, uma extensão móvel está presente se o tamanho de datagrama especificado no cabeçalho IP for maior que o tamanho especificado na mensagem de descoberta de roteador ICMP. A Figura l8.l ilustra o formato da extensão.

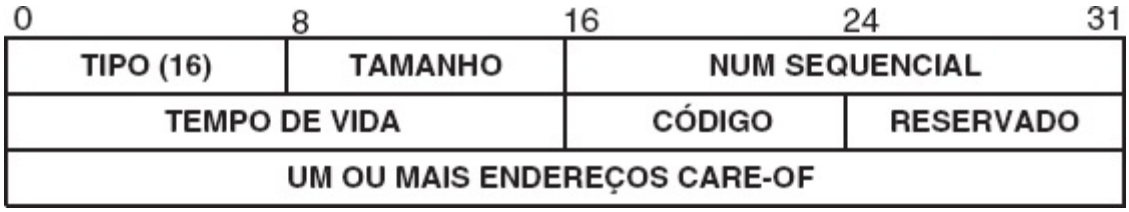


Figura 18.1 O formato de uma mensagem IPv4 *extensão de anúncio de agente móvel*. Essa extensão é anexada a uma mensagem ICMP anúncio de roteador.

Cada mensagem de extensão começa com um campo TIPO de l octeto seguido de um campo TAMANHO de l octeto. O campo TAMANHO especifica o tamanho da mensagem de extensão, em octetos, excluindo os octetos TIPO e TAMANHO. O campo TEMPO DE VIDA especifica o período de tempo máximo, em segundos, em que o agente está disponível para aceitar requisições de registro, com tudo l indicando *infinito*. O campo NUM SEQUÊNCIA especifica quando uma mensagem está perdida, e o último campo lista o endereço de pelo menos um agente externo. Cada bit no campo CÓDIGO define um recurso específico do agente, conforme listado na Figura l8.2.

Bit	Significado
7	O registro com um agente é necessário mesmo ao usar um endereço care-of colocado
6	O agente está ocupado e não está aceitando registros
5	Funções de agente como um agente de casa
4	Funções de agente como um agente externo
3	Agente usa encapsulamento mínimo
2	Agente usa encapsulamento no estilo GRE
1	Não usado (deve ser zero)
0	Agente aceita tunelamento reverse

Figura 18.2 Bits do campo CÓDIGO de um anúncio agente de mobilidade IPv4, com o bit 0 sendo o bit menos significativo do octeto.

Como a figura indica, o bit *2* e o bit *3* especificam o encapsulamento utilizado quando um host móvel se comunica com o agente externo. *Encapsulamento Mínimo* é um padrão para tunelamento IP em IP que abrevia campos do cabeçalho original para economizar espaço. *Encapsulamento de rota genérico* (*Generic Route Encapsulation – GRE*) é um padrão que permite que um protocolo arbitrário seja encapsulado; IP em IP é um caso particular.

18.10 Registro IPv4

Antes que possa receber datagramas em um local externo, um host móvel precisa estar registrado com seu agente de casa. Se estiver usando um endereço care-of de agente externo, o host móvel deve ser registrado com um agente externo.

O protocolo de *registro* permite que um host:

- registre-se com um agente na rede externa, se necessário;
- registre-se com seu agente de casa para requisitar encaminhamento;
- renove um registro que está para expirar;
- desfazer o registro após retornar para casa.

Se obtiver um endereço care-of coalocado, um host móvel realiza o registro diretamente; o endereço care-of coalocado é usado em toda a comunicação com o agente móvel de casa. Se obtiver um endereço care-of

de um agente externo, um host móvel permite que o agente externo se registre com seu agente de casa em nome do host.

18.11 Formato de mensagem de registro IPv4

Todas as mensagens de registro são enviadas via UDP; os agentes usam a porta 434. Uma mensagem de registro começa com um conjunto de campos de tamanho fixo seguido de *extensões* de tamanho variável. Cada requisição precisa conter uma *extensão de autenticação móvel de casa*, (*mobile-home authentication extension*) que permite que o agente de casa verifique a identidade móvel. A Figura l8.3 ilustra o formato da mensagem.

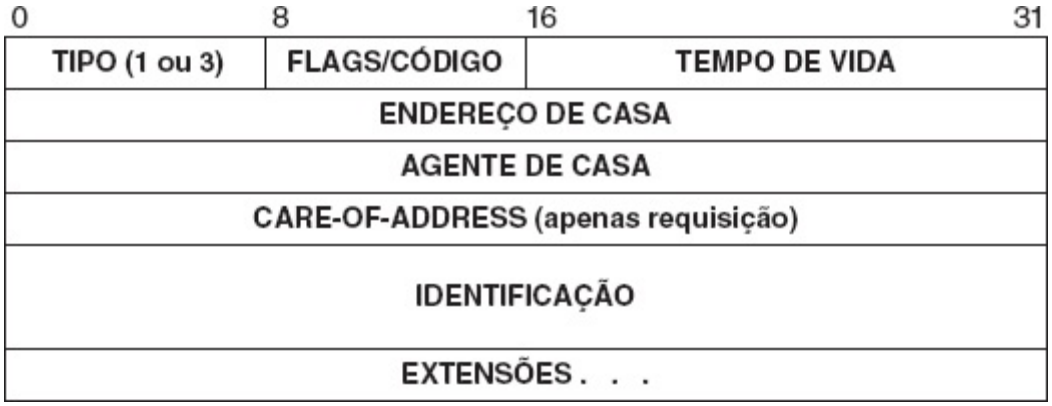


Figura 18.3 O formato de uma *requisição de registro de IP móvel* IPv4 ou mensagem de *resposta de IP móvel*.

O campo TIPO especifica se a mensagem é uma requisição de registro (*1*) ou uma resposta de registro (*3*). O campo TEMPO DE VIDA especifica o número de segundos em que o registro é válido (um zero requisita desfazer o registro imediato e tudo l especifica um tempo de vida infinito). Os campos ENDEREÇO DE CASA, AGENTE DE CASA e CARE-OF ADDRESS especificam os dois endereços IP móveis e o endereço de seu agente de casa, e o campo IDENTIFICAÇÃO contém um número de 64 bits gerado pelo móvel, que é usado para igualar requisições às respostas que chegam e para impedir que o móvel aceite mensagens antigas. Os bits do campo FLAGS/CÓDIGO são usados como um código de resultado em uma mensagem de resposta de registro, como se o registro correspondesse a uma requisição de endereço adicional (ou seja, nova) e ao encapsulamento que o agente deve usar quando encaminha datagramas para o móvel.

18.12 Comunicação com um agente externo IPv4

Dissemos que um agente externo pode atribuir um de seus endereços IPv4 para uso como um care-of address. A consequência é que o móvel não terá um endereço único na rede externa. A pergunta passa a ser: como um agente externo e um host móvel podem se comunicar por uma rede se o móvel não possui um endereço IP válido na rede? A comunicação exige um relaxamento das regras para o endereçamento IP e o uso de um esquema alternativo para vinculação de endereço: quando um host móvel envia para um agente externo, aquele tem permissão de usar seu endereço de casa como um endereço de origem IP; e quando um agente externo envia um datagrama para um móvel, o agente tem permissão de usar o endereço de casa do móvel como um endereço de destino IP. Para evitar o envio de uma requisição ARP inválida, um agente externo registra o endereço de hardware do móvel quando a primeira requisição chega e usa o endereço de hardware para enviar uma resposta. Portanto, embora não use ARP, o agente pode enviar datagramas para um móvel através de unicast de hardware. Podemos fazer o resumo conforme descrito a seguir.

Se um móvel não tiver um endereço externo único, um agente externo precisa usar o endereço de casa do móvel para comunicação. Em vez de se basear no ARP para vinculação de endereço, o agente registra o endereço de hardware do móvel quando uma requisição chega e usa as informações registradas para fornecer a vinculação necessária.

18.13 Suporte à mobilidade IPv6

A experiência com IPv4 móvel e o projeto do protocolo IPv6 levou o IETF a fazer mudanças significativas entre IPv4 móvel e IPv6 móvel. O IETF teve a intenção de integrar o suporte à mobilidade com mais força no protocolo, compensou alguns dos problemas e deficiências que haviam sido descobertos com IPv4 móveis e estimulou o uso. As diferenças podem ser caracterizadas como se segue.

- O IPv6 não usa um agente externo ou um care-of address de agente externo. Em vez disso, um host móvel IPv6 usa um care-of address coalocado e trata toda a comunicação diretamente com um agente de casa.
- Como ele permite que um host tenha múltiplos endereços IP, o IPv6

torna mais fácil para um host móvel ter simultaneamente um endereço de casa e care-of address coalocado.

- Como o IPv6 não transmite por broadcast um pedido para descobrir um agente de casa, um host IPv6 só recebe uma resposta de um agente. (O IPv4 pode receber respostas de cada agente na rede de casa.)
- Ao contrário de comunicação entre um host IPv4 móvel e um agente externo, o IPv6 móvel não depende de encaminhamento da camada de link.
- Como veremos mais tarde, o cabeçalho de extensão de roteamento IPv6 faz o encaminhamento para um host IPv6 móvel mais eficiente do que o encaminhamento a um host móvel IPv4.
- Um host móvel IPv6 não precisa de um agente externo porque pode gerar um endereço local e se comunicar com um roteador na rede externa.

18.14 Transmissão, recepção e tunelamento de datagrama

Uma vez registrado, um host móvel em uma rede externa pode se comunicar com um computador arbitrário, X. Há duas possibilidades. No caso mais simples, o host móvel cria e envia um datagrama de saída que tem endereço IP do computador X no campo de endereço de destino e o endereço de casa do móvel no campo de endereço de origem IP. O datagrama de saída segue um caminho mais curto a partir do host móvel para o destino X.

Tecnicamente, usar um endereço de casa como um endereço de origem viola os padrões TCP/IP porque um datagrama será transmitido por um host na rede N, e o endereço IP de origem no datagrama não irá coincidir com o prefixo IP para a rede de N. Se um gerente de rede opta por aplicar regras rígidas, pode configurar roteadores para proibir transmissões em que o endereço de origem não corresponde à rede local. Como podem essas restrições ser superadas? O IPv4 móvel utiliza uma técnica de duas etapas conhecida como *tunelamento (tunneling)*. Em essência, o host móvel usa o tunelamento para enviar um datagrama de saída de volta para seu agente de casa, e este transmite o datagrama como se o host móvel estivesse localizado na rede de casa.

Para usar tunelamento, um host móvel encapsula um datagrama de saída, D_1, em outro datagrama, D_2. O endereço de origem em D_2 é o care-of

address móvel, e o destino é o endereço do agente de casa móvel. Quando se recebe um datagrama em túnel, o agente de casa extrai o datagrama interno, D_1, e o encaminha ao seu destino. Ambas as etapas usam endereços válidos. A transmissão a partir do host móvel para o agente de casa tem um endereço de origem na rede externa. O datagrama interno, que viaja desde o agente de casa para o destino X, tem um endereço de origem na rede de casa.

Para o IPv4 móvel, a resposta não seguirá o caminho mais curto diretamente para o móvel. Em vez disso, a resposta irá sempre viajar para a rede de casa do móvel em primeiro lugar. O agente de casa aprende a localização do móvel a partir do registro, intercepta a resposta e usa o tunelamento para entregar a resposta para o móvel. Ou seja, quando se recebe um datagrama destinado a um host móvel, um agente de casa também usa tunelamento – ele encapsula a resposta em outro datagrama, D_3, usa o care-of address do móvel como destino para D_3 e envia o datagrama encapsulado para o móvel. A Figura 18.4 ilustra o caminho de uma resposta a partir de um computador, D, a um host móvel, M.

Podemos fazer o resumo conforme descrito a seguir.

Como um móvel usa o seu endereço de casa como um endereço de origem ao se comunicar com um destino arbitrário, cada resposta IPv4 é encaminhada para rede de casa do móvel, onde um agente intercepta o datagrama, o encapsula em outro datagrama e o encaminha, ou diretamente para o móvel ou para o agente externo que o móvel está usando.

O IPv6 móvel usa uma otimização interessante para evitar rotas ineficientes. Antes de se comunicar com o destino D, um host móvel informa seu agente de casa. O host, então, inclui um cabeçalho de mobilidade no datagrama que envia. O destino D pode se comunicar com o agente de casa, verificar a localização atual do host móvel e usar um cabeçalho de rota IPv6 para direcionar o datagrama para lá.

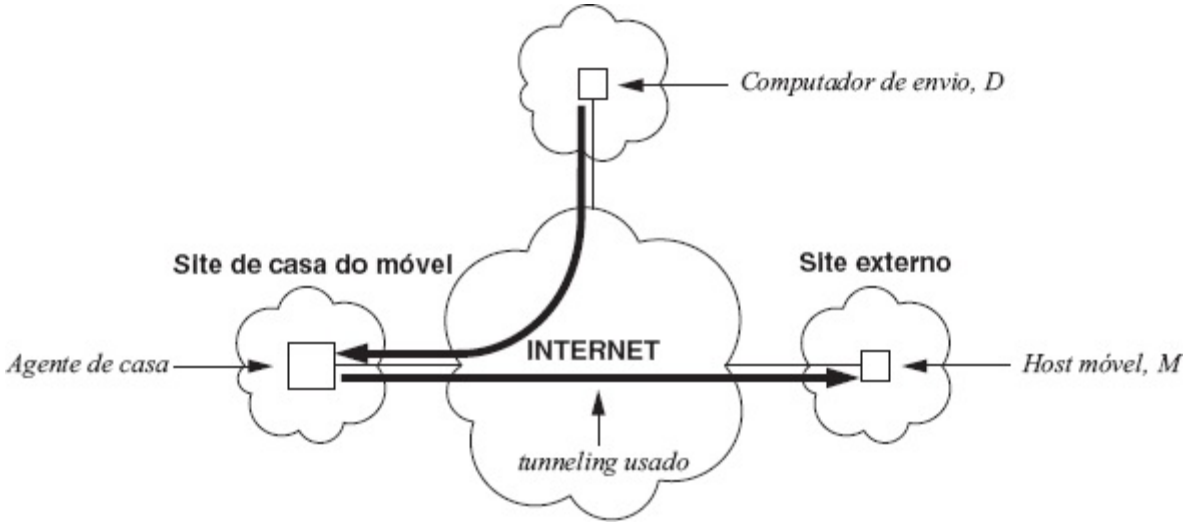


Figura 18.4 O caminho que uma resposta faz do computador *D* para o host móvel *M* quando o host está conectado a uma rede externa.

É claro que as trocas entre um host móvel, um agente de casa, e um destino devem ser seguras. Além disso, a troca deve ocorrer para cada destino. Portanto, todo o procedimento acarreta sobrecarga considerável, e só é adequado para situações em que um móvel permanece conectado a uma determinada rede externa por um longo tempo e se comunica extensivamente com um determinado destino. O ponto importante é descrito a seguir.

Para otimizar o encaminhamento de resposta, o IPv6 torna possível para um destino saber a localização atual de um host móvel e enviar datagramas diretamente para ele, sem passar por um agente de casa; como a otimização de rotas requer várias trocas de mensagens, ela só é útil para móveis que se movem com pouca frequência e tendem a se comunicar extensivamente com um determinado destino.

18.15 Acesso de mobilidade IP e problemas não resolvidos

Apesar das melhores intenções da IETF, o IP móvel não tem sido um sucesso esmagador. Uma das razões para a falta de interesse surgiu a partir de uma mudança no tipo de mobilidade que os utilizadores usufruem. Quando a mobilidade IP foi concebida, era limitada a computadores portáteis volumosos – um usuário podia transportar um computador

portátil para um local remoto e, em seguida, usar o computador. Agora, muitos usuários móveis têm telefones inteligentes que permitem a contínua mobilidade, *on-line*.

Duas outras influências conspiraram ainda mais para desestimular o uso de IP móvel. Primeiro, a tecnologia VPN (abordada no próximo capítulo) foi inventada. Uma VPN permite que um dispositivo remoto mantenha um endereço de casa e tenha acesso completo à sua rede de casa, *como se o dispositivo remoto estivesse ligado diretamente a ela*. Em segundo lugar, alguns aplicativos confiam em endereços IP ou em pesquisa de DNS reverso. Em vez disso, porque os esquemas de autenticação que usam senhas permitem que um usuário acesse serviços como o e-mail de um computador com um endereço IP arbitrário, manter um endereço IP não é tão importante como era antes. Mais importante: usar um endereço arbitrário permite roteamento eficiente. Por exemplo, quando um usuário viaja para uma cidade distante, se conecta a redes Wi-Fi hotspot e acessa uma página web, datagramas viajam diretamente entre o dispositivo do usuário e o servidor web, sem um desvio para rede de casa do usuário.

As deficiências do esquema IP móvel podem ser resumidas a seguir.

- Falta de entrega (hand-off) e rotas hierárquicas.
- Problemas com autenticação em redes externas.
- Encaminhamento reverso ineficiente, especialmente para IPv4 móvel.
- Detecção de endereço duplicado em IPv6 móvel.
- Comunicação com hosts na rede de casa do móvel.

As próximas seções consideram cada um desses itens com mais detalhe.

18.15.1 Falta de entrega (hand-off) e rotas hierárquicas

Quando pensavam em mobilidade, os desenvolvedores imaginavam computadores portáteis sendo utilizados em locais remotos. Consequentemente, o IP móvel não se comporta como um sistema celular. Ele não tem facilidades para entregas de alta velocidade entre torres locais de celular, nem fornece um sistema de roteamento hierárquico que poderia restringir o alcance das mudanças de rota durante a migração a partir de uma rede de acesso a uma rede de acesso adjacente.

18.15.2 Problemas com autenticação em redes externas

Embora algumas redes externas permitam o acesso irrestrito, muitas não

permitem. Pelo contrário, as redes, muitas vezes, exigem que um usuário se autentique antes que o acesso seja concedido. Por exemplo, um hotel pode exigir que um convidado entre com o número do quarto e seu último nome antes de ter o acesso concedido. Normalmente, a autenticação requer que o usuário obtenha um endereço IP e, em seguida, use o endereço para lançar um navegador web. O hotel intercepta a solicitação da web e exibe uma página de autenticação para o usuário. Uma vez que a autenticação tenha sido concluída, é concedido acesso à Internet global ao dispositivo do usuário.

O IP móvel não consegue lidar com a autenticação de acesso baseada na web por duas razões. Primeiro, o IP móvel sempre começa pelo registro com um agente de casa. Uma rede remota que requer autenticação não encaminhará pacotes para o agente de casa até que a autenticação seja concluída. Em segundo lugar, o IP móvel especifica que os aplicativos devem sempre usar o endereço de casa do host móvel. Assim, mesmo que o usuário lance um browser de web, este irá tentar usar um endereço de IP de uma rede de casa, e o mecanismo de autenticação rejeitará a conexão.

18.15.3 Encaminhamento reverso ineficiente, especialmente para IPv4 móvel

Como vimos, uma resposta enviada para um host móvel IPv4 será sempre encaminhada para a rede de casa do móvel em primeiro lugar e, em seguida, para sua atual localização. O problema é especialmente grave porque a comunicação do computador exibe *localização espacial de referência* – um host móvel visitando uma rede externa tende a se comunicar com computadores nessa rede. Para entender por que localidade espacial é um problema, considere a Figura 18.5.

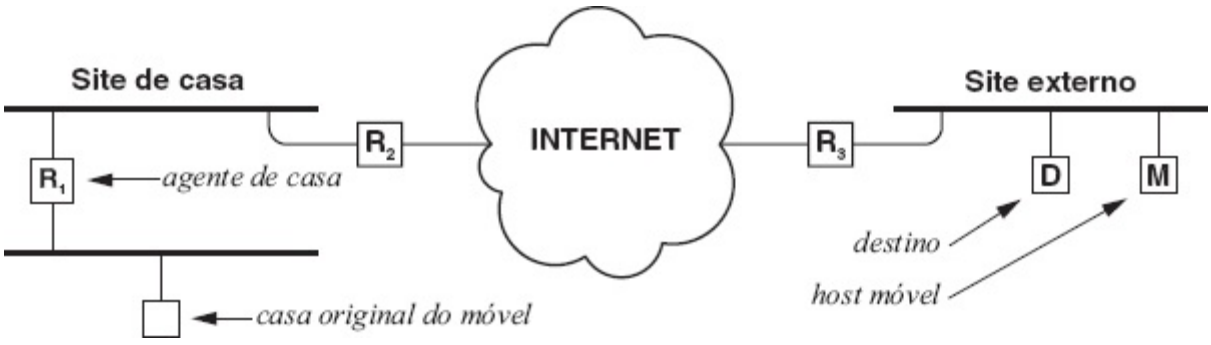


Figura 18.5 Uma topologia na qual o roteamento móvel IPv4 é absurdamente ineficiente. Quando um host móvel, *M*, se comunica com um destino local, *D*, a resposta de *D* viaja através da Internet até o

agente de casa do móvel, R_1, e então retorna ao host móvel.

Na figura, o móvel M passou de sua rede de casa para uma rede externa. O móvel foi registrado com o seu agente de casa, o roteador R_1, e o agente de casa concordou em encaminhar datagramas. Quando o host móvel se comunica com destino D, que está localizado no mesmo local que o móvel, respostas enviadas de D para M seguem um caminho por R_3, através da Internet, até a rede de casa do móvel e, em seguida, voltam em túnel através da Internet para o host móvel. Isto é, um datagrama enviado entre dois computadores adjacentes cruza a Internet duas vezes. Porque o tempo para cruzar a Internet pode ser ordens de magnitude maior do que o tempo para entrega local, a situação descrita anteriormente é chamada às vezes de *problema de travessia dupla* (*two-crossing problem*). Se o destino D não está na mesma rede que o móvel, ocorre uma versão um pouco menos grave do problema, que é conhecido como *encaminhamento triângulo*, (*triangle forwarding*) ou o *encaminhamento de dog-leg* (*dog-leg forwarding*).

Se um site sabe que um determinado host móvel ficará em visita por um longo tempo e espera que o host móvel interaja com computadores locais, o gerente da rede pode instalar rotas de host específicas para evitar o encaminhamento ineficiente. Cada roteador no local deve ter uma rota de host específica para o host móvel visitante. A desvantagem de um arranjo desses surge da falta de atualizações automáticas: quando o host móvel deixa o site, o gerente tem que remover manualmente as rotas de host específico ou o host se torna inalcançável para computadores no site. Podemos fazer o resumo conforme descrito a seguir.

O IP móvel introduz uma ineficiência de roteamento como problema de travessia dupla (two-crossing problem), que ocorre quando um móvel visitante se comunica com um computador em um site externo, ou próximo de um. Cada datagrama enviado para o móvel viaja através da Internet para a casa do agente móvel, que, em seguida, o encaminha de volta para o site externo. Eliminar a ineficiência requer propagação de uma rota de host específico.

18.15.4 Detecção de endereço duplicado em IPv6 móvel

No IPv6, quando um host se junta a uma nova rede, passa por três etapas: encontra o prefixo de rede (ou prefixos) que está sendo usado na rede, gera um endereço unicast e verifica se o endereço é único. A primeira e a terceira etapas requerem uma troca de pacotes e incluem tempo limite. Um host móvel deve realizar a detecção de endereço duplicado a cada vez que troca de redes e obtém um care-of address para a nova rede. Ironicamente, as normas especificam que um host móvel IPv6 também deve gerar e verificar um único endereço link-local. A sobrecarga de detecção de endereços duplicados torna o IPv6 inadequado para movimento rápido.

18.15.5 Comunicação com hosts na rede de casa do móvel

Outro problema interessante surge quando um computador da rede de casa do móvel tenta se comunicar com um móvel que está visitando uma rede externa. Dissemos que, quando um computador móvel está atualmente fora de casa, seu agente de casa intercepta todos os datagramas destinados para o móvel que chegam ao site da casa. Interceptar datagramas que chegam ao site é relativamente simples: um gerente de rede escolhe executar o software do agente de casa no roteador que conecta o site de casa com o resto da Internet.

Um caso especial surge, no entanto, quando um host na rede de casa de um móvel envia um datagrama para o móvel. Já que o IP especifica entrega direta através da rede local, o remetente não irá encaminhar o datagrama através de um roteador. Em vez disso, um remetente IPv4 vai usar ARP para encontrar o endereço de hardware do móvel, e um host IPv6 usará descoberta de vizinho. Em ambos os casos, o host vai encapsular o datagrama em um frame e transmiti-lo diretamente para o móvel.

Se um móvel tiver se mudado para uma rede externa, hosts e roteadores da rede de casa não poderão enviar datagramas diretamente para esse móvel. Portanto, o agente de casa deve organizar a captura e encaminhamento de todos os datagramas destinados para o móvel, incluindo as enviadas pelos hosts locais. Um agente de casa IPv4 utiliza uma forma de ARP *proxy* para lidar com interceptação local: sempre que um computador da rede de casa faz um ARPs para um host móvel IPv4 que se mudou para uma rede externa, o agente de casa responde à solicitação ARP e fornece seu próprio endereço de hardware.

Ou seja, hosts IPv4 locais são enganados para encaminhar qualquer datagrama destinado para o móvel para o agente de casa; este pode, então, encaminhar o datagrama para o móvel na rede externa.

Para IPv6, a transmissão local representa um problema maior que requer suporte adicional ao protocolo. Em particular, os computadores em uma rede IPv6 usam o Protocolo de Descoberta de Vizinho (*Neighbor Discovery Protocol – NDP*) para saber quais vizinhos estão presentes. Se um móvel deixa a rede de casa, outros computadores usando NDP vão rapidamente declarar que o móvel está inalcançável.* Portanto, o IP móvel deve arranjar uma maneira para que hosts na rede de casa possam ser informados quando um host móvel tenha se mudado para outro local.

Para resolver o problema, o IPv6 móvel modifica a descoberta de vizinho. Em essência, um agente de casa funciona como um *proxy* quando um móvel está fora. O agente de casa informa aos computadores na rede local que um host específico é móvel. Se eles encontram datagramas destinados ao host móvel, outros computadores na rede de casa encaminham os datagramas de acordo. Quando o móvel volta para casa, o encaminhamento precisa ser removido.

18.16 Tecnologias de separação identificador-localizador alternativo

O problema fundamental que um desenvolvedor enfrenta quando adiciona suporte para a mobilidade IP surge a partir de um princípio fundamental de endereçamento IP: o prefixo de um endereço IP amarra o endereço a uma rede específica. Isto é, um endereço de IP serve como um *localizador*. A vantagem de um localizador reside num sistema de encaminhamento eficiente que encaminha cada datagrama para a rede de destino correta. A desvantagem de um localizador decorre da sua incapacidade para se acomodar a movimento: se a localização muda, o endereço deve mudar. O endereçamento MAC Ethernet ilustra a alternativa: um endereço Ethernet é um valor global e único, mas não contém qualquer informação sobre o local onde o computador está localizado. Isto é, um endereço serve como um *identificador único*. A desvantagem de utilizar um identificador vem da ineficiência de roteamento: são necessárias rotas de host específico.

Como se pode conceber um esquema de endereçamento que combina as vantagens de ambos, localizadores e identificadores? Vimos a ideia fundamental: um endereço de host precisa de duas peças conceituais. A primeira parte é um identificador global único que nunca muda, e o segundo é um localizador que muda quando o anfitrião se move para uma nova rede. No IP móvel, as duas peças conceituais são representadas por

dois endereços de IP independentes. Ou seja, um host armazena dois endereços, usa o seu endereço de casa como um identificador e seu care-of address como um localizador.

Várias propostas foram criadas para formalizar a ideia de um par *identificador-localizador*. As abordagens diferem no tamanho dos dois elementos, o modo como os valores são atribuídos, se as duas partes são consideradas como campos de bits de um único endereço grande ou como dois elementos separados, e se ambas as partes são visíveis para aplicações. Por exemplo, a Cisco Systems definiu o *localizador/protocolo ID de separação* (*Locator/ID Separation Protocol – LISP*), que usa um par de endereços IP semelhante à maneira como IP móvel usa endereços. O IETF definiu um protocolo chamado *interconexão transparente de lotes de links* (*Transparent Interconnection of Lots of Links – Trill*) que estende a ideia de pontes aprendendo a mobilidade em uma internet de ampla área.

18.17 Resumo

O IP móvel permite que um computador se desloque de uma rede para outra sem alterar o seu endereço IP e sem a necessidade de roteadores para propagar uma rota de host específico. Quando ele se move de sua rede de casa original para uma rede externa, um computador móvel deve obter um endereço temporário adicional, conhecido como care-of address. Os aplicativos usam o endereço original de casa do móvel; o care-of address só é usado pelo software de rede subjacente para permitir o encaminhamento e entrega na rede externa.

Uma vez que se detecta que se moveu, um móvel IPv4 ou obtém um care-of address coalocado ou descobre um agente de mobilidade externa e solicita que este atribua um care-of address. Um móvel IPv6 pode gerar um care-of address coalocado sem a necessidade de um agente externo. Depois de obter o endereço, o móvel registra com seu agente de casa (diretamente ou indiretamente através do agente externo) e solicita que o agente encaminhe os datagramas.

Feita a inscrição, um móvel pode usar o seu endereço de casa para se comunicar com um computador arbitrário na Internet. Datagramas enviados pelo móvel são encaminhados diretamente para o destino especificado. Responder roteamento pode ser ineficiente porque um datagrama enviado para o móvel será encaminhado para rede doméstica do móvel onde é interceptado pelo agente de casa, encapsulado em IP e, em seguida,

enviado por um túnel para o celular.

O esquema para IP móvel foi projetado para o movimento lento, como uma visita a um hotel. Quando aplicado a dispositivos que se movem rapidamente, o IP móvel tem inconvenientes graves e não foi amplamente adotado.

EXERCÍCIOS

18.1 Compare os esquemas de encapsulamento em RFCs 2003 e 2004. Quais são as vantagens e desvantagens de cada um?

18.2 Leia a especificação IP móvel com cuidado. Com que frequência deve um roteador enviar um anúncio de agente de mobilidade? Por quê?

18.3 Consulte a especificação IP móvel. Quando um agente externo encaminha um pedido de registro para um agente de casa de um móvel, quais portas protocolo são usadas? Por quê?

18.4 A especificação para IP móvel permite que um único roteador funcione como um agente de casa para uma rede e um agente externo que suporta os visitantes na rede. Quais são as vantagens e as desvantagens do uso de um único roteador para ambas as funções?

18.5 Leia a especificação para o IPv6 móvel. Quantos formatos de mensagens separadas são definidos?

18.6 Suponha que um provedor de celular adota o IPv6 móvel para uso com seus telefones. Calcule o número de pacotes enviados quando um telefone passa de uma rede para outra.

18.7 Estenda o exercício anterior. Se N usuários de telefones celulares ativos dirigirem em uma estrada a 95 Km/h e cada um deve mudar de uma torre de celular para outra dentro de uma área de 450 metros a meio caminho entre duas torres, estime a capacidade de rede necessária para lidar com as mensagens móveis que o IPv6 gera para realocar os telefones de uma torre de celular para outra.

18.8 Leia as especificações para IPv4 e IPv6 móveis para determinar como um host móvel se junta a um grupo de *multicast*. Como são roteados datagramas multicast para o móvel em cada caso? Qual abordagem é mais eficiente? Explique.

18.9 Compare IPv4 e IPv6 móveis com o protocolo LISP da Cisco. Quais são as diferenças em termos de funcionalidade?

18.10 Compare IPv4 e IPv6 móvel com o protocolo TRILL. O que o TRILL oferece?

18.11 Leia sobre protocolos hand-off usados em uma rede celular. Protocolos semelhantes podem ser usados com IP? Sim ou não? Por quê?

18.12 Considere os aplicativos que você usa. Algum deles requerem que você mantenha um endereço IP (ou seja, se o seu dispositivo pessoal de Internet precisa de um endereço de residência permanente)? Explique.

---

* O Capítulo 23 explica o Sistema de Nome de Domínio (*Domain Name System – DNS*).

* Quando é necessário distinguir, usamos os termos *IPv4 móvel* e *IPv6 móvel*.

* O padrão para a mobilidade IPv6, RFC 6275, compreende 169 páginas, define muitos tipos de mensagens e fornece algumas regras para operação de protocolo.

* Um host móvel também pode efetuar multicast para *todo o grupo de agentes* (224.0.0.11).

** Um agente móvel também anexa uma *extensão de prefixo* em cada mensagem.

* O Capítulo 22 discute NDP, que é às vezes escrito *IPv6-ND* para enfatizar que o protocolo é uma parte integrante do IPv6.

CAPÍTULO 19

Virtualização de rede: VPNs, NATs e sobreposições

CONTEÚDOS DO CAPÍTULO

19.1 Introdução

Os capítulos anteriores descrevem uma internet como uma abstração de um único nível, que consiste de redes interconectadas por roteadores. Este capítulo considera uma alternativa — uma arquitetura de internet de dois níveis que virtualiza a Internet. O primeiro nível é composto por uma internet convencional que fornece conectividade universal. Uma organização utiliza a conectividade subjacente para construir um segundo nível que esteja de acordo com suas necessidades.

O capítulo examina três tecnologias que empregam virtualização. Uma tecnologia permite que uma empresa possa se conectar a vários sites em toda a Internet global ou que um funcionário use a Internet global para acessar a rede corporativa a partir de uma localização remota arbitrária, mantendo toda a comunicação confidencial. Uma segunda forma permite que um site forneça acesso global à Internet para muitos hosts, enquanto utiliza apenas um único endereço IP globalmente válido. A terceira tecnologia permite que uma organização crie uma topologia de rede arbitrária no topo da topologia Internet.

19.2 Virtualização

Utilizamos o termo *virtualização* para descrever uma abstração que é usada para ocultar detalhes de implementação e fornecer funcionalidade de

alto nível. Em geral, os mecanismos de virtualização utilizam um mecanismo subjacente que não inclui a funcionalidade necessária desejada.

Já vimos tecnologias e protocolos que proporcionam um nível de virtualização de rede. Por exemplo, um comutador Ethernet VLAN permite que um gerente configure um comutador para agir como um conjunto de comutadores Ethernet independentes. O TCP fornece a abstração de uma conexão confiável fim a fim (end-to-end). Em cada caso, no entanto, o serviço é uma ilusão – o mecanismo subjacente não oferece o serviço que a virtualização cria. O TCP, por exemplo, constrói entrega confiável de conexão orientada em um transporte sem conexão não confiável.

Este capítulo mostra que várias formas de virtualização de rede são úteis, bem como populares. Vamos considerar a motivação e os usos de virtualizações, bem como os mecanismos utilizados para criar cada forma. O Capítulo 28 continua a discussão considerando uma tecnologia que pode ser usada para criar caminhos virtuais através de uma internet.

19.3 Redes privadas virtuais (VPNs)

A comutação de pacotes usada na Internet global tem a vantagem de baixo custo, mas a desvantagem de que os pacotes de vários usuários viajam através de uma determinada rede. Como resultado, a Internet global não pode garantir que a comunicação realizada através dela continue a ser *privada*. Em particular, se uma organização é composta por vários sites, o conteúdo de datagramas que viajam através da Internet entre os sites pode ser visto por pessoas de fora, porque eles passam através de redes de propriedade de pessoas de fora (ou seja, ISPs).

Ao pensar sobre privacidade, os gerentes de rede, muitas vezes, classificam redes em uma arquitetura de dois níveis, que distingue entre as redes que são *internas* a uma organização e as redes que são *externas*. Como a organização pode controlar as redes internas de sua propriedade, ela pode ter garantias sobre a forma como os dados são encaminhados e impedir que se tornem visíveis para os outros. Assim, as redes internas podem garantir privacidade, enquanto as externas não podem.

Se uma organização tem vários sites, como ela pode garantir a privacidade para o tráfego enviado entre eles? A abordagem mais fácil consiste na construção de uma rede completamente isolada, que pertença e seja operada pela organização. Usamos o termo *rede privada* ou *intranet privada* para uma rede desse tipo. Como uma rede privada utiliza circuitos digitais alugados para interligar locais e pelo fato de as empresas de

telefonia garantirem que ninguém de fora tenha acesso a esses circuitos, todos os dados permanecem privados enquanto viajam de um site para outro.

Infelizmente, uma intranet totalmente privada pode não ser suficiente por dois motivos. Em primeiro lugar, a maioria das organizações precisa de acesso à Internet global (por exemplo, contatos com clientes e fornecedores). Em segundo lugar, os circuitos digitais alugados são caros. Consequentemente, muitas organizações buscam alternativas que oferecem um custo menor. Uma abordagem usa uma forma de virtualização que o Capítulo 16 discute: MPLS. Uma conexão MPLS pode custar significativamente menos do que um circuito digital alugado com a mesma capacidade.

Apesar de ser menos caro do que um circuito digital, um caminho MPLS é muito mais caro do que uma conexão tradicional Internet. Assim, a escolha é clara: menor custo pode ser alcançado por meio do envio de tráfego pela Internet global, e a maior privacidade pode ser conseguida por conexões dedicadas. Surge a pergunta: é possível alcançar um alto grau de privacidade e o baixo custo de conexões convencionais de Internet? Dizendo de outra maneira, se poderia perguntar:

> *como pode uma organização que usa a Internet global conectar seus sites e garantir que toda a comunicação seja mantida privada?*

A resposta está em uma tecnologia conhecida como uma *rede privada virtual*, (*Virtual Private Network – VPN*). A ideia de uma VPN é simples: enviar datagramas através da Internet global, mas criptografar o conteúdo. O termo *privado* surge porque o uso de criptografia significa que a comunicação entre qualquer par de computadores permanece oculta de estranhos. O termo *virtual* surge porque uma VPN não requer circuitos alugados dedicados para conectar um site a outro. A Figura 19.1 ilustra o conceito.

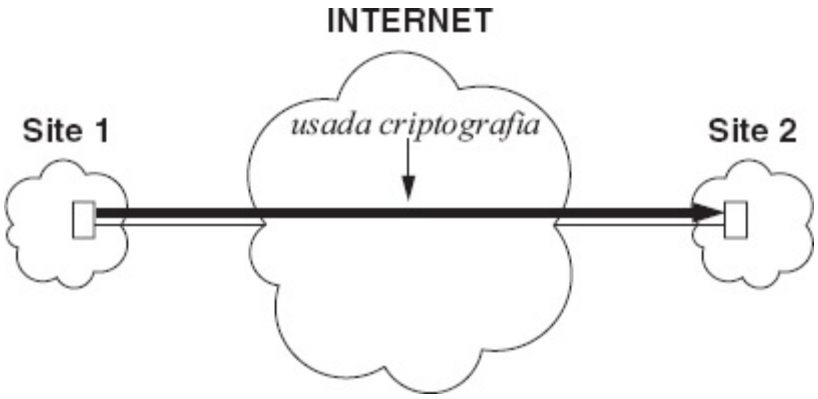


Figura 19.1 Ilustração de uma VPN que usa criptografia ao mandar dados através da Internet global entre dois roteadores em dois sites de uma organização.

19.4 Tunelamento VPN e encapsulamento IP em IP

Uma técnica mencionada no capítulo anterior desempenha um papel importante na tecnologia VPN: *tunelamento (tunneling)*. Uma VPN usa o tunelamento pela mesma razão do IP móvel: para enviar um datagrama através da Internet entre dois sites. Porque não simplesmente encaminhar o datagrama normalmente? A resposta está na maior privacidade (ou seja, a confidencialidade). Criptografar a carga em um datagrama não garante privacidade absoluta porque um estranho pode usar os campos de IP de origem e de destino, o campo tipo datagrama, bem como a frequência e volume de tráfego para descobrir quem está se comunicando. Uma VPN criptografa um datagrama inteiro, incluindo o cabeçalho IP. Na verdade, para ocultar informações de pessoas de fora, alguns VPNs preenchem todos datagramas com octetos extras antes de criptografá-los, o que significa que uma pessoa de fora não pode usar o comprimento do datagrama para deduzir o tipo de comunicação. A consequência é que a criptografia significa que o cabeçalho do datagrama não pode ser usado para encaminhamento.

A maioria das VPNs usa tunelamento IP em IP. Ou seja, o datagrama original é cifrado, e o resultado é colocado na seção de carga de outro datagrama para transmissão. A Figura 19.2 ilustra o encapsulamento.

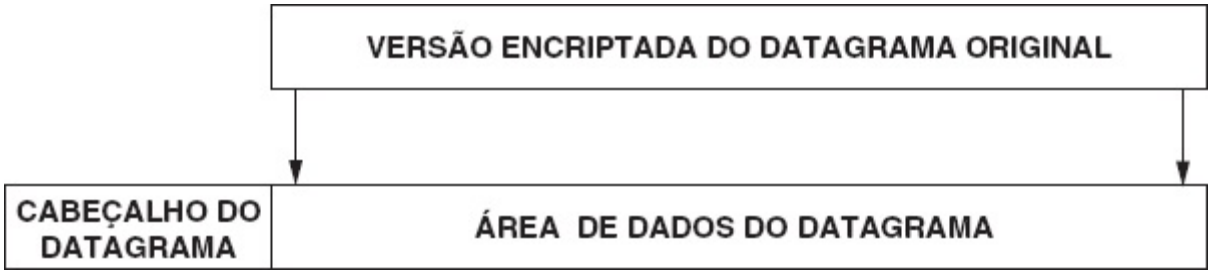


Figura 19.2 Ilustração do encapsulamento IP em IP usado com uma VPN. O datagrama original é criptografado antes de ser enviado.

Uma organização pode ter muitos computadores em cada site. Para a privacidade máxima, computadores individuais não participam de uma VPN. Em vez disso, um gerente organiza o encaminhamento para que datagramas enviados através do túnel VPN sempre viajem de um roteador em um site a um roteador no outro site. Quando um datagrama chega pelo túnel, o roteador recebedor decifra a carga útil para reproduzir o datagrama original e, em seguida, o encaminha dentro do site. Embora os datagramas atravessem redes arbitrárias enquanto passam através da Internet, terceiros não podem decodificar o conteúdo, pois não têm a chave de criptografia. Além disso, mesmo as identidades da fonte original e do destino final estão ocultas, porque o cabeçalho do datagrama original é criptografado. Assim, apenas dois endereços no cabeçalho exterior do datagrama são visíveis: o endereço da fonte é o endereço de IP do roteador numa extremidade do túnel, e o endereço de destino é o endereço de IP do roteador na outra extremidade do túnel. Consequentemente, uma pessoa de fora não pode deduzir quais computadores nos dois sites estão se comunicando.

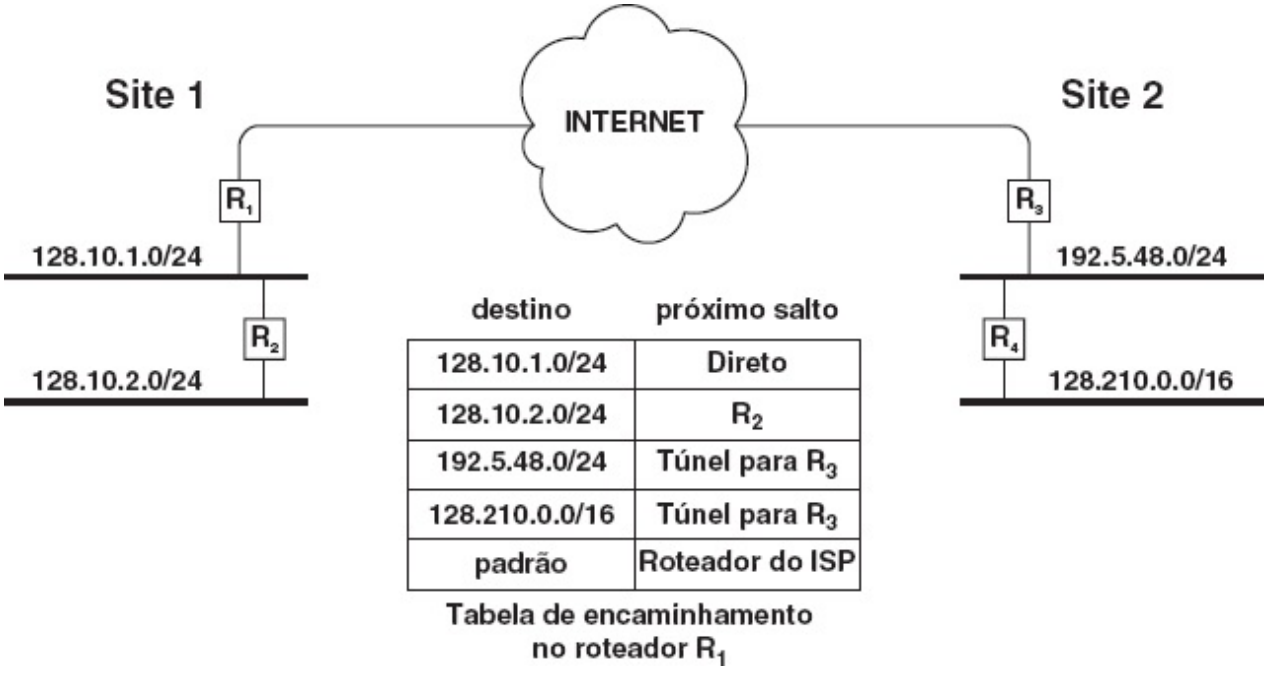
Resumindo em poucas palavras a seguir.

Apesar de uma VPN enviar dados através da Internet em nível mundial, pessoas externas não podem deduzir quais computadores nos dois sites estão se comunicando ou que dados eles estão trocando.

19.5 Endereçamento e encaminhamento VPN

A maneira mais fácil de entender endereçamento e roteamento VPN é pensar em cada túnel VPN como um substituto para um circuito alugado entre dois roteadores. Como é habitual, a tabela de encaminhamento em cada um dos dois roteadores contém entradas para destinos dentro da organização. A tabela de encaminhamento também contém uma interface de

rede que corresponde ao túnel VPN, e datagramas enviados para outro site são direcionados através do túnel. A Figura 19.3 ilustra a ideia mostrando redes em dois sites e a tabela de encaminhamento para um roteador que cuida do tunelamento VPN. Embora o exemplo use IPv4, a mesma abordagem pode ser usada para o IPv6.



destino	próximo salto
128.10.1.0/24	Direto
128.10.2.0/24	R_2
192.5.48.0/24	Túnel para R_3
128.210.0.0/16	Túnel para R_3
padrão	Roteador do ISP

Tabela de encaminhamento no roteador R_1

Figura 19.3 Uma VPN que abrange dois sites e a tabela de roteamento de R_1 com um túnel de R_1 a R_3 configurado como um circuito ponto a ponto.

A figura mostra uma entrada-padrão na tabela de encaminhamento R_1 com um ISP como o próximo salto. A ideia é que os computadores no site 1 possam acessar computadores no site 2 ou computadores na Internet global. O túnel é usado apenas para acesso site a site; outros datagramas são encaminhados para o ISP.

Como exemplo do encaminhamento em uma VPN, considere um datagrama enviado de um computador na rede l28.l0.2.0 para um computador na rede l28.2l0.0.0. O host emissor encaminha o datagrama para R_2, que o encaminha para R_1. De acordo com a tabela de roteamento em R_1, o datagrama precisa ser enviado pelo túnel para R_3. Portanto, R_1 criptografa o datagrama, o encapsula na área de dados de um datagrama externo com destino R_3. Então, R_1 encaminha o datagrama externo através

do ISP local pela Internet. O datagrama chega em R_3, que o reconhece como tunelado de R_1. R_3 descriptografa a área de dados para produzir o datagrama original, consulta o destino em sua tabela de encaminhamento e encaminha o datagrama para R_4 para entrega.

19.6 Extensão da tecnologia VPN a hosts individuais

Muitas corporações usam a tecnologia VPN para permitir que os funcionários trabalhem a partir de locais remotos. É fornecido ao empregado um software VPN que funciona em um dispositivo móvel (por exemplo, um laptop). Para utilizar o software de VPN, um usuário inicia seu aparelho, se conecta a uma rede arbitrária e obtém um endereço IP de um provedor de rede local, normalmente. Se um usuário está trabalhando em casa, ele se conecta ao ISP que fornece seu serviço de Internet residencial. Se está trabalhando em um hotel, pode obter o serviço a partir do ISP que serve os hóspedes do hotel, e assim por diante. Uma vez que uma conexão de rede foi obtida, o usuário inicia o software de VPN, que é pré-configurado para formar um túnel para um roteador na rede corporativa.

O software VPN reconfigura a pilha de protocolo no computador do usuário. Quando inicia, o software VPN forma um túnel para a rede corporativa e se comunica através do túnel para obter um segundo endereço IP (ou seja, um endereço na rede corporativa). O software, então, configura a pilha de protocolos para restringir toda a comunicação para ir pelo túnel VPN. Ou seja, os aplicativos no computador só enxergam o endereço IP que foi obtido da rede corporativa. Todos os datagramas que as aplicações enviam são transferidos através do túnel para a rede corporativa, e apenas datagramas vindos do túnel são entregues às aplicações. Portanto, do ponto de vista de um aplicativo, o computador do usuário parece estar ligado diretamente à rede corporativa.

Para garantir que a comunicação seja confidencial, todos os datagramas que viajam através do túnel são criptografados. No entanto, há uma potencial falha de segurança: ao contrário de um roteador, um laptop pode ser roubado facilmente. Se o software de VPN pode lidar com a criptografia e a descriptografia, um estranho que rouba o laptop seria capaz de obter acesso à rede corporativa. Portanto, o software VPN emitido para os usuários normalmente requer uma senha. A senha é combinada com a hora do dia para gerar uma chave de cifra única que é usada para uma única sessão. Sem a senha correta VPN, um laptop roubado não pode ser usado

para obter acesso à rede corporativa.

19.7 Usar uma VPN com endereços IP privados

Curiosamente, apesar de uma VPN usar a Internet global ao se comunicar entre os sites, a tecnologia torna possível a criação de uma intranet privada que não fornece conectividade com a Internet global para os hosts da rede corporativa. Para ver como, imagine atribuir endereços não roteáveis de hosts (por exemplo, um endereço de site específico IPv6 ou um endereço IPv4 privado). A um roteador em cada site é atribuído um endereço IP globalmente válido, e o roteador é configurado para formar um túnel VPN para o roteador no outro site. A Figura 19.4 ilustra o conceito.

Figura 19.4 Exemplo de uma VPN que interliga dois sites na Internet global, enquanto os computadores em cada local usam endereços não roteáveis (ou seja, privados).

Na figura, a organização optou por utilizar o prefixo IPv4 não roteável 10.0.0.0/8 (um prefixo que foi reservado para uso em uma rede privada). O site 1 usa sub-rede 10.1.0.0/16, enquanto o site 2 usa a sub-rede 10.2.0.0 / 16. Apenas dois endereços IP globalmente válidos são necessários para tornar uma VPN possível. Um é atribuído à conexão do roteador R_1 à Internet e o outro é atribuído à conexão de R_2 com a Internet. Os dois sites podem estar distantes um do outro e podem obter o serviço a partir de dois ISPs independentes, o que significa que os dois endereços globalmente válidos podem ser independentes. Roteadores e hosts dentro de cada site usam o espaço de endereço privado; apenas os dois roteadores que participam no tunelamento VPN precisam conhecer ou usar endereços IP globalmente válidos.

19.8 Tradução de endereço de rede (NAT)

As seções anteriores descrevem uma forma de virtualização que permite que uma organização conecte sites através da Internet global, mantendo toda a

comunicação confidencial. Esta seção considera uma tecnologia que inverte a virtualização, fornecendo acesso em nível de IP entre hosts em um site e a Internet global, sem a necessidade de cada máquina no local ter um endereço IP globalmente válido. Conhecida como *tradução de endereço de rede* (*Network Address Translation – NAT*), a tecnologia tornou-se extremamente popular tanto para os consumidores como para pequenas empresas. Por exemplo, roteadores sem fio (wireless) usados em casas empregam NAT.

A ideia por trás da NAT é simples. Um site coloca um *dispositivo NAT*, informalmente chamado de *NAT box*,* entre a rede (s) no site e o resto da Internet como a Figura 19.5 ilustra.

Do ponto de vista dos hosts no site, a NAT box parece ser um roteador que se conecta à Internet. Ou seja, o encaminhamento no site está configurado para direcionar datagramas de saída para a NAT box. Para um ISP que presta serviço para o site, a NAT box parece ser um único host. Ou seja, a NAT box obtém um único endereço IP globalmente válido e parece usar o endereço para enviar e receber datagramas.

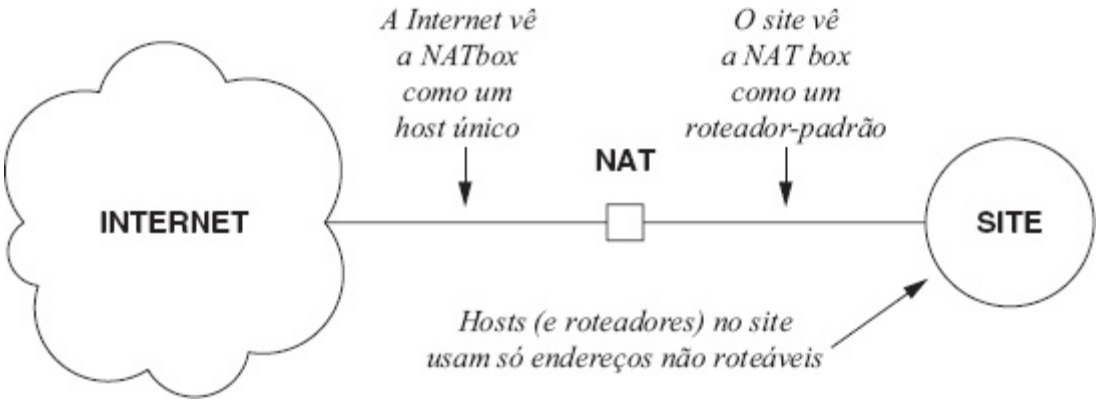


Figura 19.5 Ilustração de uma NAT box que permite a um site usar endereços IP não roteáveis.

A chave para a tecnologia NAT surge porque esta *traduz* (ou seja, altera) datagramas que viajam em qualquer direção. Quando um host no site envia um datagrama para um destino na Internet, a NAT coloca informações sobre o datagrama de saída (incluindo um registro do remetente original) em uma tabela, muda os campos no cabeçalho e envia o datagrama modificado para o seu destino. Em particular, a NAT altera o endereço IP de origem para fazer parecer que o datagrama veio da NAT box.

Assim, se um site na Internet que recebe um datagrama de resposta, a resposta será enviada para a NAT box. Quando se recebe um datagrama a partir da Internet, a NAT box consulta a sua tabela, encontra o remetente original, muda os campos no cabeçalho e encaminha o datagrama. Em particular, a NAT muda o endereço IP de destino para o endereço privado que o host no site está usando.

Devido a ambos os datagramas de entrada e saída viajarem através da NAT box, o software NAT pode enganar tanto os hosts atrás da NAT box como os hosts arbitrários na Internet. Quando um host na Internet recebe datagramas do site, eles parecem ter se originado na NAT box. Quando um host atrás da NAT box obtém um endereço IP, o endereço é não roteável (ou site-local). No entanto, o host pode usar o endereço de origem não roteável ao enviar e receber datagramas com um host arbitrário da Internet.

A principal vantagem da NAT surge de sua combinação de generalidade e transparência. A NAT é mais geral do que um conjunto de gateways de aplicação, pois permite que um host interno arbitrário acesse um serviço arbitrário em um computador na Internet global. A NAT é transparente porque hosts no site executam software de protocolo totalmente padrão. Resumindo em poucas palavras a seguir.

A tecnologia de tradução de endereço de rede (Network Address Translation – NAT) fornece acesso transparente em nível de IP à Internet para um host que tem um endereço IP privado, não roteável.

19.9 Criação de tabela de tradução da NAT

Nosso resumo da NAT desconsidera alguns detalhes porque não especifica como a NAT sabe que o host interno deve receber um datagrama que chega da Internet. Dissemos que a NAT mantém uma tabela de tradução que usa para fazer encaminhamento para um host. Que informação a NAT coloca na tabela de tradução e quando as entradas da tabela foram criadas?

A forma mais amplamente usada da NAT* armazena seis itens em sua tabela de tradução conforme descrito a seguir.

- *IP interno*. O endereço IP não roteável usado por um computador interno.
- *Porta interna*. O número da porta protocolo usada por um computador interno.

- *IP externo*. O endereço IP de um computador externo localizado em algum lugar na Internet.
- *Porta externa*. O número da porta protocolo usada por um computador externo.
- *Tipo de carga*. O tipo de protocolo de transporte (por exemplo, TCP, UDP ou ICMP).
- *Porta NAT*. O número da porta protocolo usado pela NAT box (para evitar situações em que dois computadores internos escolhem o mesmo número de porta).

Quando um datagrama chega da Internet, a NAT pesquisa a tabela. Se o endereço de origem IP do datagrama corresponde ao *IP externo*, o número de porta de protocolo origem corresponde à *Porta NAT*, e o tipo do datagrama corresponde ao *tipo de carga útil*, a NAT usa a entrada da tabela. A NAT substitui o endereço IP de destino do datagrama (que é sempre o endereço da própria NAT box) com o *IP Interno*, substitui o número de porta de protocolo de destino com a *porta interna*, e envia o datagrama para o host interno.

Claro que uma entrada de tabela deve ser criada antes de um datagrama chegar da Internet; caso contrário, a NAT não tem como identificar o host interno correto para o qual o datagrama deve ser encaminhado. Como e quando é inicializada a tabela? A descrição acima implica que datagramas de saída são sempre utilizados para criar as entradas da tabela. Embora tenhamos descrito a forma mais utilizada de NAT, existem outras. As possibilidades incluem as descritas a seguir.

- *Inicialização manual*. Um gerente configura a tabela de tradução manualmente antes que qualquer comunicação ocorra.
- *Datagramas de saída*. A tabela é construída como um efeito colateral de um host interno enviando um datagrama. A NAT usa o datagrama de saída para criar uma entrada na tabela de tradução que registra os endereços de origem e destino.
- *Consultas de nome de entrada*. A tabela é construída como um efeito colateral de manipular consultas de nome de domínio. Quando um host na Internet consulta o nome de domínio de um host interno,* o

software DNS envia o endereço da NAT box como resposta e cria uma entrada na tabela de tradução NAT para poder encaminhar para o host interno correto.

Cada técnica de inicialização apresenta vantagens e desvantagens. A inicialização manual fornece mapeamentos permanentes e permite que hosts arbitrários na Internet alcancem serviços no site. Usar um datagrama de saída para inicializar a tabela tem a vantagem de tornar a comunicação de saída completamente automática, mas tem a desvantagem de não permitir que a comunicação tenha início externo. Usar consultas de nome de domínio de entrada acomoda as comunicações iniciadas de fora, mas exige que o software de nome de domínio seja modificado.

Como dissemos anteriormente, a maioria das implementações de NAT usam datagramas de saída para inicializar a tabela; a estratégia é especialmente popular para roteadores sem fio usados em Wi-Fi hot spots. O roteador pode ser conectado diretamente a um ISP, exatamente como um host. Por exemplo, o roteador sem fio pode ser conectado ao modem DSL ou modem a cabo suprido pelo ISP e oferece conexões de rádio Wi-Fi. Quando um host móvel se conecta via Wi-Fi, o software NAT em execução no roteador sem fio atribui ao host móvel um endereço IP não roteável privado e atribui seu endereço como um endereço do roteador-padrão. Um host móvel pode se comunicar com qualquer host na Internet apenas enviando datagramas para o roteador sem fio através da rede Wi-Fi. A Figura 19.6 ilustra a arquitetura.

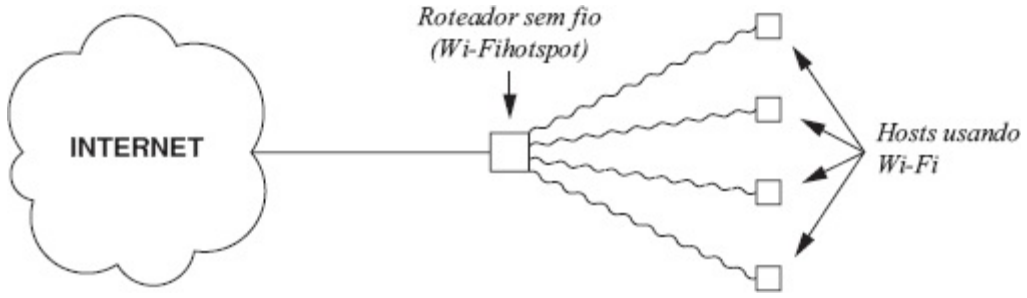


Figura 19.6 O uso de NAT por um roteador sem fio. Cada host é atribuído com um endereço IP privado.

Um roteador sem fio deve atribuir um endereço IP a um host sempre que este se conecta. Por exemplo, se um host usa IPv4, o roteador pode atribuir ao primeiro host 192.168.0.1, ao segundo 192.168.0.2 e assim por diante. Quando um host envia um datagrama para um destino na Internet, encaminha o datagrama através da rede Wi-Fi e o roteador sem fio aplica o

mapeamento NAT de saída antes de enviar o datagrama através da Internet. Da mesma forma, quando uma resposta chega da Internet, o roteador sem fio aplica a tradução NAT de entrada e encaminha o datagrama através da rede Wi-Fi para o host correto.

19.10 Variantes da NAT

Existem muitas variantes da NAT, e muitos nomes têm sido utilizados para descrevê-las. Até agora, descrevemos *NAT simétrica* que permite que um host arbitrário no site se contate com uma porta de protocolo arbitrária em um host na Internet. Muitas das variantes propostas se concentram em servidores operando por trás de uma NAT box para permitir que os hosts externos da Internet iniciem a comunicação (ou seja, podem chegar pacotes antes de um host do site enviar um pacote). Por exemplo, uma variante conhecida como *port restricted cone NAT* usa um pacote de saída para estabelecer uma porta externa, e, então, envia todos os pacotes que chegam à porta para o host interno. Assim, se um host interno H_1 envia um pacote da porta de origem X e a NAT box a mapeia para a porta externa Y, todos os pacotes recebidos destinados à porta Y são direcionados para a porta X em H_1, não importa qual host na Internet envia os pacotes.

19.11 Um exemplo de tradução NAT

Um exemplo esclarecerá tradução NAT. Lembre-se de que a versão do NAT que temos discutido é chamada NAPT porque traduz portas de protocolo, bem como endereços IP. A Figura 19.7 ilustra o conteúdo de uma tabela de tradução IPv4 usada com NAPT quando quatro computadores em um site criaram seis conexões TCP para sites externos na Internet.

Endereço IP Interno	Porta Interna	Endereço IP Externo	Porta Externa	Porta NAT	Tipo de Carga
192.168.0.5	38023	128.10.19.20	80	41003	tcp
192.168.0.1	41007	128.10.19.20	80	41010	tcp
192.168.0.6	56600	207.200.75.200	80	41012	tcp
192.168.0.6	56612	128.10.18.3	25	41016	tcp
192.168.0.5	41025	128.10.19.20	25	41009	tcp
192.168.0.3	38023	128.210.1.5	80	41007	tcp

Figura 19.7 Um exemplo de tabela de tradução usada por NAPT. A figura inclui números de porta bem como endereços IPv4.

A figura ilustra três casos interessantes: um único host no site formou conexões com dois hosts na Internet, dois hosts no site formaram, cada um, uma conexão com o mesmo servidor web na Internet, e dois hosts no site estão usando o mesmo número de porta de origem simultaneamente.

Na figura, cada entrada corresponde a uma conexão TCP. O host da Internet 192.168.0.6 formou duas conexões para hosts externos, uma com um servidor web (porta 80) no host externo 207.200.75.200 e a outra com um servidor de e-mail (porta 25) no host externo 128.10.18.3. Dois hosts internos, 192.168.0.5 e 192.168.0.1, estão ambos acessando a porta de protocolo 80 no computador externo 128.10.19.20. Como cada host do site é livre para escolher um número de porta de origem, não se pode garantir unicidade. No exemplo, as conexões TCP de 192.168.0.5 e 192.168.0.3 têm ambas a porta de origem 38023. Assim, para evitar conflitos potenciais, a NAT atribui uma porta NAT única para cada comunicação que é usada na Internet. Além disso, uma porta NAT não está relacionada com a porta de origem que um host de envio escolhe. Assim, a conexão do host 192.168.0.1 pode usar a porta 41007, e a NAT box pode usar a porta 41007 para uma conexão completamente diferente.

Lembre-se de que o TCP identifica cada conexão com um 4-tupla que representa o endereço IP e o número da porta protocolo de cada extremidade. Por exemplo, as duas primeiras entradas na tabela correspondem a conexões TCP com os seguintes 4-tuplas:

(192.168.0.5, 38023, 128.10.19.20, 80)

(192.168.0.1, 41007, 128.10.19.20, 80)

No entanto, quando o computador 128.10.19.20 na Internet receber datagramas, a NAT box terá traduzido o endereço de origem, o que significa que as mesmas duas conexões terão as seguintes 4-tuplas:

(G, 41003, 128.10.19.20, 80)

(G, 41010, 128.10.19.20, 80)

onde *G* é o endereço global válido da NAT box.

A principal vantagem da NAPT reside na generalidade que ela alcança com um único endereço IP válido globalmente; a principal desvantagem surge porque restringe comunicação para TCP, UDP, ICMP.* Já que quase

todos os aplicativos usam TCP ou UDP, a NAPT é transparente. Um computador no site pode usar números de porta de origem arbitrários, e pode acessar vários computadores externos simultaneamente. Enquanto isso, vários computadores no site podem acessar a mesma porta em um determinado computador externo simultaneamente sem interferência. Resumindo, a seguir, em poucas palavras.

Embora existam diversas variantes da NAT, a forma NAPT é a mais popular porque traduz os números de porta de protocolo, bem como endereços IP.

19.12 Interação entre NAT e ICMP

Embora tenhamos descrito NAT como operando em endereços IP e números de porta de protocolo, a tradução de endereços pode afetar outras partes de um pacote. Por exemplo, considere ICMP. Para manter a ilusão de transparência, a NAT deve entender e alterar o conteúdo de uma mensagem ICMP. Por exemplo, suponha que um host interno usa *ping* para testar a alcançabilidade de um destino na Internet. O host espera receber uma *resposta de eco* ICMP para cada mensagem de *solicitação de eco* ICMP que envia. Assim, a NAT deverá encaminhar respostas de eco de entrada para o host correto. No entanto, não encaminha todas as mensagens ICMP que chegam da Internet (por exemplo, se as rotas na NAT box estão incorretas, uma mensagem ICMP deve ser processada localmente). Assim, quando uma mensagem de ICMP chega da Internet, a NAT deve primeiro determinar se a mensagem deve ser tratada localmente ou enviada para um host interno.

Antes de encaminhar para um host interno, a NAT deve traduzir toda a mensagem ICMP. Para entender a necessidade de tradução ICMP, considere uma mensagem ICMP de *destino inalcançável*. A mensagem contém o cabeçalho de um datagrama, *D*, que causou o erro. Infelizmente, a NAT traduz endereços antes de enviar *D*, de modo que o endereço de origem em *D* é o endereço válido globalmente da NAT box em vez do endereço do host interno. Assim, antes de enviar a mensagem ICMP de volta para o host interno, a NAT deve abrir a mensagem ICMP e traduzir os endereços em *D* para que eles apareçam exatamente da forma que o host

interno usou. Depois de fazer a mudança, a NAT deve recalcular o checksum em *D*, o checksum no cabeçalho ICMP e o checksum no cabeçalho exterior do datagrama.

19.13 Interação entre NAT e aplicativos

Embora o ICMP complique a NAT, proporcionar transparência para alguns protocolos de aplicação requer um esforço substancial. Em geral, a NAT não funcionará com qualquer aplicativo que envia os endereços IP ou portas de protocolo como dados. Em particular, o *Protocolo de Transferência de Arquivos* (*File Transfer Protocol – FTP*), que é usado para fazer download de arquivos grandes, estabelece uma conexão de controle entre o cliente e o servidor e, em seguida, faz uma nova conexão TCP para cada transferência de arquivos. Como parte do protocolo, um lado obtém uma porta de protocolo na máquina local, converte o número em ASCII e envia o resultado através da conexão de controle para o outro lado. O outro lado, em seguida, forma uma conexão TCP com a porta especificada. Considere o que acontece quando a comunicação entre os dois lados passa por uma NAT box. Um host por trás da NAT box pode formar uma conexão de controle. No entanto, se o host obtém uma porta local e passa as informações para o outro lado, a NAT não esperará que cheguem pacotes e irá descartá-los. Portanto, FTP só pode funcionar se a NAT monitora o conteúdo da conexão de controle, seleciona um número de porta e altera o fluxo de dados para refletir o novo número.

Muitas implementações de NAT reconhecem protocolos de aplicativos populares, incluindo FTP, e fazem as mudanças necessárias no fluxo de dados. Como alternativa, as variantes de aplicativos foram criadas para evitar a transmissão dessas informações no fluxo de dados (ou seja, evitar fazer conexões no sentido inverso). Por exemplo, uma versão de FTP que não requer que o servidor se conecte de volta com o cliente é conhecida como *FTP passiva*. Os programadores de aplicativos devem estar cientes da NAT e, para tornar seus aplicativos totalmente gerais, devem evitar passar endereços ou números de porta no fluxo de dados. Resumindo, a seguir, em poucas palavras.

A NAT afeta o ICMP e os protocolos de aplicação; com exceção de algumas aplicações-padrão como FTP, um protocolo de aplicação que passa endereços IP ou números de porta de protocolo como dados não

irá funcionar corretamente na NAT.

Alterar itens num fluxo de dados aumenta a complexidade do NAPT de duas maneiras. Em primeiro lugar, isso significa que NAPT deve ter conhecimento detalhado de cada aplicação que transfere essas informações. Em segundo lugar, se os números de porta estão representados em ASCII, como é o caso com o FTP, alterar o valor pode mudar o número de octetos transferidos. Inserir mesmo um octeto adicional em uma conexão TCP é difícil porque cada octeto no fluxo tem um número de sequência. Um remetente não sabe que foram inseridos dados adicionais e continua a atribuir números de sequência sem esses dados. Quando recebe dados adicionais, o receptor gera reconhecimentos que representam os dados. Assim, depois de inserir dados adicionais, a NAT deve utilizar uma técnica conhecida como *TCP splicing* para traduzir os números de sequência de cada segmento de saída e em cada aviso de entrada.

19.14 NAT na presença de fragmentação

A descrição anterior da NAT fez uma importante suposição sobre o IP: um sistema NAT recebe datagramas IP completos, e não fragmentos. O que acontece se um datagrama for fragmentado? Endereços IP não representam um problema porque cada fragmento contém os endereços de origem e host de destino. Infelizmente, fragmentação tem uma consequência significativa para NAPT (a variante mais usada do NAT). Como a Figura 19.7 mostra, a consulta de tabela NAPT utiliza números de porta de protocolo do cabeçalho de transporte, bem como endereços IP do cabeçalho IP. Infelizmente, números de porta não estão presentes em todos os fragmentos, pois somente o primeiro fragmento de um datagrama leva o cabeçalho de protocolo de transporte. Assim, antes de realizar a consulta, um sistema NAPT precisa receber e examinar o primeiro fragmento do datagrama. A semântica IP complica ainda mais a NAT, pois os fragmentos podem chegar fora de ordem. Assim, o fragmento que leva o cabeçalho de transporte pode não chegar antes de outros fragmentos.

Um sistema NAPT pode seguir um de dois projetos: o sistema pode salvar os fragmentos e tentar reconstituir o datagrama, ou o sistema pode descartar os fragmentos e processar apenas datagramas completos. Nenhuma das duas opções é desejável. A reconstituição exige informações de estado, o que significa que o sistema não pode escalar para alta velocidade ou altos fluxos (e é susceptível a ataques maliciosos). O

descarte de fragmentos significa que o sistema não processará tráfego arbitrário. Na prática, apenas os sistemas NAPT que são projetados para redes de baixa velocidade escolhem a reconstituição; muitos sistemas rejeitam datagramas fragmentados.

19.15 Domínios de endereço conceitual

Descrevemos NAT como uma tecnologia que pode ser usada para conectar uma rede privada à Internet global. Na verdade, a NAT pode ser usada para interconectar quaisquer dois *domínios de endereço*. Portanto, pode ser usada entre duas corporações que possuem, cada uma, uma rede privada usando o endereço 10.0.0.0. Mais importante, pode ser usada em dois níveis: entre os domínios de endereço privado de um cliente e os domínios de endereço privado de um ISP, bem como entre o domínio de endereço do ISP e a Internet global. Finalmente, a NAT pode ser combinada com a tecnologia VPN para formar uma arquitetura híbrida em que endereços privados são usados dentro da organização e a NAT é usada para fornecer conectividade entre cada site e a Internet.

Como exemplo de múltiplos níveis da NAT, considere um indivíduo que trabalha em casa com vários computadores que estão conectados a uma LAN. Esse indivíduo pode atribuir endereços privados aos computadores em casa e usar a NAT entre a rede doméstica e a intranet corporativa. A corporação também pode atribuir endereços privados e usar a NAT entre sua intranet e a Internet global.

19.16 Versões da NAT para Linux, Windows e Mac

Adicionalmente, existem dispositivos autônomos (stand-alone), tais como roteadores sem fio e implementações de software da NAT que permitem que um computador convencional execute funções de NAT. Por exemplo, a Microsoft usa o nome *Compartilhamento de Conexão de Internet* (*Internet Connection Sharing*) para o software NAT que usuários podem configurar; software adicional está disponível para servidores Windows. Várias versões de NAT foram criadas para o sistema operacional Linux. Em particular, *iptables* e *IP Masquerade*, ambos implementam NAT. Em geral, o software NAT consiste em uma combinação: ferramentas de aplicação que permitem ao usuário configurar NAT e suporte do kernel para reescrita de pacotes e firewall. A maioria dos softwares NAT suporta múltiplas variantes. Por exemplo, porque oferece inspeção de estado dos

pacotes, o *iptables* pode ser configurado para lidar com NAT básico ou NAPT.

19.17 Redes sobrepostas

O tema deste capítulo é virtualização: o uso de tecnologias que fornecem serviços abstratos para um subconjunto de computadores que imitam o serviço prestado por hardware dedicado. Por exemplo, a tecnologia VPN permite aos usuários que conectem seus computadores a uma rede remota, *como se* houvesse uma conexão direta. A tecnologia NAT permite que um conjunto de computadores com endereços privados se comuniquem com destinos arbitrários na Internet *como se* cada computador no conjunto tivesse um endereço IP globalmente válido. Podemos estender ainda mais a virtualização? Veremos que é possível criar uma rede virtual que se conecte a um grande conjunto arbitrário de hosts através de muitos sites e permitir que os hosts se comuniquem *como se* estivessem conectados a uma única rede de hardware.

Como uma rede sobreposta (overlay) pode ser útil? Nós já discutimos um excelente exemplo quando consideramos VPNs: segurança. Suponha uma corporação que tenha seis sites e queira criar uma intranet corporativa. A corporação gostaria de garantir que a comunicação entre os sites seja mantida em sigilo. A tecnologia VPN permite configurar os roteadores em cada site para usar a Internet em vez de links individuais e datagramas criptografados que fluem através dos links virtuais.

O overlay de rede estende a virtualização de duas maneiras. Em primeiro lugar, em vez de links individuais, a tecnologia de overlay pode ser usada para criar uma rede inteira. Em segundo lugar, em vez de criar uma internet virtual, a tecnologia de overlay pode ser usada para criar uma rede virtual de camada 2.

Para entender como funciona uma rede sobreposta, imagine nossa corporação exemplo com seis sites. Tal como acontece com uma VPN, a corporação configura um ou mais roteadores em cada site para tunelar datagramas para outros sites. Mais importante, a corporação pode impor uma topologia de roteamento arbitrária e políticas de tráfego arbitrárias. Por exemplo, se ela observa que o tráfego de vídeo enviado diretamente do site 1 para o site 4 tem problema de altos atrasos, o roteamento overlay pode ser organizado como se houvesse uma ligação direta entre cada par de sites, com exceção do 1 e 4. Alternativamente, se os sites 1 – 3 estão na costa Leste e os sites 4 – 6 estão na Costa Oeste, a corporação pode organizar a

rede sobreposta para emular três links de longa distância. A Figura 19.8 ilustra as duas possíveis topologias de roteamento.

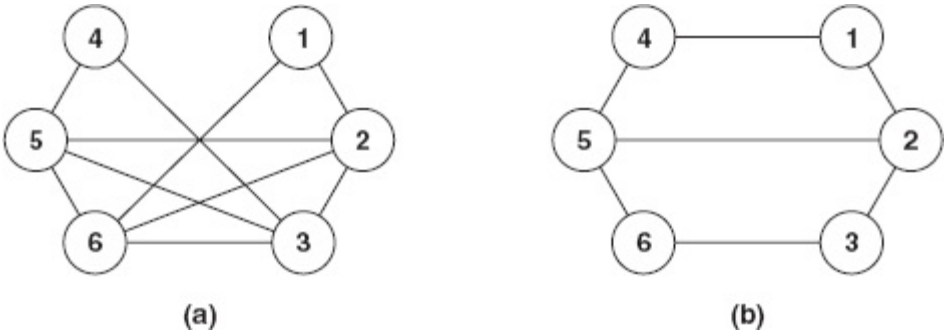


Figura 19.8 Exemplo de duas topologias possíveis que podem ser alcançadas com tecnologia overlay.

É importante compreender que as topologias mostradas na figura são virtuais, não reais. Na prática, não há links. Em vez disso, todos os sites se conectam à Internet e a utilizam para entregar datagramas. Assim, apenas a configuração de sobreposição na Figura 19.8 (a) pode evitar que o site 1 envie os datagramas diretamente para o site 4.

19.18 Sobreposições múltiplas simultâneas

As topologias da Figura 19.8 podem parecer um tanto sem sentido. Afinal, se todos os sites se conectam à Internet global, datagramas podem ser encaminhados diretamente para o site de destino correto. No entanto, uma organização pode ter razões para preferir uma topologia sobre outra. Para entender, vamos supor que a organização estabeleceu políticas de roteamento para subgrupos. Por exemplo, uma política pode exigir que todo o tráfego financeiro deva ser isolado do tráfego de clientes ou que os computadores do departamento jurídico em cada site devem estar em uma rede privada isolada da Internet.

As duas ideias básicas por trás da tecnologia overlay são:

- operação dentro de um site;
- operação simultânea de várias overlays.

Operação dentro de um site. Embora tenhamos descrito as sobreposições para topologias WAN que conectam sites, a tecnologia de sobreposição se estende a hosts e roteadores dentro de um site. Assim, é possível criar uma rede sobreposta separada que conecta os computadores

pertencentes ao departamento financeiro e outra sobreposição que conecta os computadores pertencentes ao departamento jurídico.

Operação simultânea de várias sobreposições. A segunda ideia-chave na tecnologia de sobreposição é a operação simultânea de múltiplas sobreposições. Ou seja, várias redes de sobreposição podem coexistir (ou seja, operar ao mesmo tempo), o que torna possível para uma organização criar múltiplas redes virtuais que só se encontram em pontos de interconexão específicos. No nosso exemplo, os computadores do departamento jurídico podem ser completamente isolados de outros.

O Capítulo 28 continua a discussão da virtualização e da sobreposição de redes e combina os conceitos de VPN e sobreposição de rede apresentados neste capítulo com os conceitos de classificação e comutação discutidos no Capítulo 17. Veremos que o resultado é uma tecnologia que pode ser usada para configurar os caminhos virtuais.

19.19 Resumo

As tecnologias de virtualização nos permitem criar redes artificiais com propriedades desejáveis pela imposição de restrições à comunicação via Internet convencional. Examinamos três tecnologias de virtualização: VPN, NAT e redes sobrepostas (overlay).

Embora uma rede privada garanta privacidade, o custo pode ser alto. A tecnologia da *rede privada virtual* (*VPN*) oferece uma alternativa de baixo custo que permite que uma organização use a Internet para interconectar múltiplos sites e use criptografia para garantir que o tráfego entre sites permaneça privado. Como uma rede privada tradicional, uma VPN pode ser completamente isolada (caso em que endereços privados são atribuídos aos hosts) ou uma arquitetura híbrida que permite aos hosts se comunicarem com destinos na Internet global.

A Tradução de Endereço de Rede (Network Address Translation) provê acesso transparente em nível de IP para a Internet global a partir de um host que tem endereço privado. A NAT provê acesso em nível de IP transparente à Internet global de um host que tem um endereço privado e é especialmente popular entre os roteadores usados em hot spots Wi-Fi. Além disso, a NAT traduz (ou seja, reescreve) datagramas que viajam de um host no site até a Internet ou de volta da Internet para um host no site. Embora existam muitas variantes de NAT, a mais popular é conhecida como *tradução de porta e rede* (*Network And Port Translation – NAPT*). Adicionalmente à reescrita de endereços IP, a NAPT reescreve portas

protocolo do nível de transporte que fornecem generalidade completa e permite que aplicações arbitrárias operando em computadores arbitrários no site acessem simultaneamente serviços na Internet.

Tecnologias de rede de sobreposição permitem que uma organização defina uma rede entre vários sites, como se estivessem conectados por circuitos digitais alugados. A sobreposição define as possíveis interconexões entre sites. Com uma sobreposição, protocolos de roteamento convencionais podem ser usados para encontrar rotas ao longo dos caminhos de sobreposição.

EXERCÍCIOS

19.1 Sob que circunstâncias uma VPN transfere substancialmente mais pacotes do que o IP convencional ao enviar os mesmos dados através da Internet? (Dica: pense no encapsulamento.)

19.2 Implementações de software de NAT que são usadas para fornecer a funcionários remotos acesso à intranet de uma organização muitas vezes reduzem o MTU da rede que é relatado para aplicações locais. Explique o porquê.

19.3 Procure a definição de *cone* como aplicada ao NAT. Quando um sistema NAT é considerado cone cheio?

19.4 A NAT traduz tanto os endereços IP de origem como os de destino. Quais endereços são traduzidos em datagramas que chegam a partir da Internet?

19.5 Considere uma mensagem ICMP de host inalcançável enviada através de duas NAT box que interligam três domínios de endereços. Quantas traduções de endereço ocorrerão? E quantas traduções de números de porta de protocolo?

19.6 Imagine que decidimos criar uma nova Internet paralela à Internet existente que aloca os endereços a partir do mesmo espaço de endereço. A tecnologia NAT pode ser usada para conectar as duas Internets arbitrariamente grandes que usam o mesmo espaço de endereço? Se sim, explicar como. Se não, explique por quê.

19.7 A NAT é completamente transparente para um host? Para responder à pergunta, tente encontrar uma sequência de pacotes que um host pode transmitir para determinar se ele está localizado por trás de uma NAT box.

19.8 Quais são as vantagens de combinar a tecnologia NAT com a

tecnologia VPN? E as desvantagens?

19.9 Configure NAT em um sistema Linux entre um domínio de endereço privado e a Internet. Quais serviços bem conhecidos funcionam corretamente e quais não?

19.10 Leia sobre uma variante da NAT chamada *duas vezes NAT*, que permite que uma comunicação seja iniciada a partir de qualquer um dos lados da NAT box a qualquer momento. Como a *duas vezes NAT* garante que as traduções sejam consistentes? Se duas instâncias de *duas vezes NAT* são utilizadas para interligar três domínios de endereços, o resultado é completamente transparente para todos os hosts?

19.11 Desenhe um diagrama de camada de protocolo usada com uma rede sobreposta.

19.12 A tecnologia de sobreposição pode ser utilizada com Camada 2, assim como com Camada 3. Desenhe um sistema que utilize uma sobreposição para formar um grande VLAN Ethennet que inclua múltiplos sites.

---

* Embora o nome implique que a NAT requer hardware de propósito específico, é possível executá-lo em um computador de uso geral (por exemplo, um PC).

* Tecnicamente, a versão da NAT descrita aqui é tradução de endereço e porta de rede (*Network Address and Port Translation – NAPT*).

* O Capítulo 23 descreve como opera o *Sistema de Nome de Domínio*.

* A próxima seção explica que uma NAT box traduz e encaminha algumas mensagens ICMP e cuida de outras mensagens localmente.

CAPÍTULO 20

Modelo de interação cliente-servidor

CONTEÚDOS DO CAPÍTULO

20.1 Introdução

Capítulos anteriores apresentaram os detalhes da tecnologia de TCP/IP, incluindo os protocolos que fornecem serviços básicos e aqueles que os roteadores usam para propagar roteamento de informações. Agora que entendemos a tecnologia básica, vamos nos voltar para as questões de como os programas de aplicação se beneficiam a partir do uso cooperativo dos protocolos TCP/IP e da Internet global. Embora as aplicações de exemplo sejam práticas e interessantes, elas não formam a ênfase principal porque nenhum aplicativo Internet dura para sempre. Novos aplicativos são criados e outros mais antigos desaparecem. Portanto, nosso foco recai sobre os padrões de interação entre programas aplicativos de comunicação.

O padrão básico de interação entre aplicativos que usam a rede é conhecido como o modelo *cliente-servidor*. A interação cliente-servidor forma a base da comunicação de rede e serve como alicerce para os serviços de aplicativos. Várias extensões de alto nível para o modelo cliente-servidor foram criadas, incluindo a rede *peer-to-peer* (*par a par*, ou *ponto a ponto*) e o processamento *map-reduce*. Apesar da jogada de marketing, as extensões não substituem as interações cliente-servidor. Em vez disso, os novos modelos simplesmente sugerem novas maneiras de organizar grandes sistemas distribuídos – nos níveis mais baixos, eles dependem de interações cliente-servidor.

Este capítulo considera o modelo básico cliente-servidor; capítulos mais à frente descreverão sua utilização em aplicações específicas.

20.2 O modelo cliente-servidor

Um *servidor* é um programa aplicativo que oferece um serviço atrás de uma rede. Um servidor aceita uma requisição que chega, forma uma resposta e retorna o resultado para o requisitante. Para os serviços mais simples, cada requisição chega em um único datagrama, e o servidor retorna uma resposta em outro programa.

Um programa em execução se torna um *cliente* quando envia uma requisição para um servidor e espera uma resposta. Como o modelo cliente-servidor é uma extensão conveniente e natural de comunicação entre processos utilizada em uma única máquina, torna-se fácil para os programadores construírem programas que usam o modelo para interagir.

Os servidores podem realizar tarefas simples ou complexas. Por exemplo, um servidor *time-of-day* (*servidor de hora*) apenas retorna o tempo atual sempre que um cliente envia a ele um pacote. Um *servidor web* recebe solicitações de navegadores para buscar cópias de páginas da web; o servidor retorna a página solicitada para cada solicitação.

Nós afirmamos que o servidor é um programa de aplicação. Na verdade, um servidor é um aplicativo em execução, normalmente chamado de *processo*. A vantagem de implementar servidores como programas aplicativos é que um servidor pode ser executado em qualquer computador com sistema que aceite comunicação TCP/IP. No momento em que a carga em um servidor aumenta, o servidor pode ser executado em uma CPU mais rápida. Existem tecnologias que permitem que um servidor seja replicado em vários computadores físicos para aumentar a confiança ou o desempenho – solicitações de entrada são espalhadas entre os computadores que executam o servidor para reduzir a carga. Se a principal finalidade de um computador for o suporte de um programa servidor específico, o termo "servidor" pode ser aplicado ao computador e também ao programa servidor. Portanto, é comum ouvir afirmações como "o computador *A* é o nosso servidor web".

20.3 Um exemplo trivial: servidor de eco UDP

A forma mais simples de interação cliente-servidor usa distribuição de

datagrama para transmitir mensagens de um cliente para um servidor e vice-versa. Por exemplo, a Figura 20.1 ilustra a interação em um servidor de eco UDP. Um *servidor de eco UDP* é iniciado e deve estar em execução antes que o cliente envie uma solicitação. O servidor especifica que ele usará a porta reservada para o serviço de eco UDP, a porta UDP 7. O servidor então entra em um loop infinito que tem três etapas descritas a seguir.

(1) Aguardar um datagrama chegar à porta eco UDP.
(2) Inverter os endereços de origem e destino* (incluindo os endereços IP de origem e destino, bem como os números das portas UDP).
(3) Retornar o datagrama resultante ao seu emissor original.

Uma vez que ele está sendo executado, o programa aplicativo de servidor pode fornecer o serviço eco. Em alguns sites da Internet, um aplicativo torna-se um cliente do serviço eco UDP enviando um datagrama.

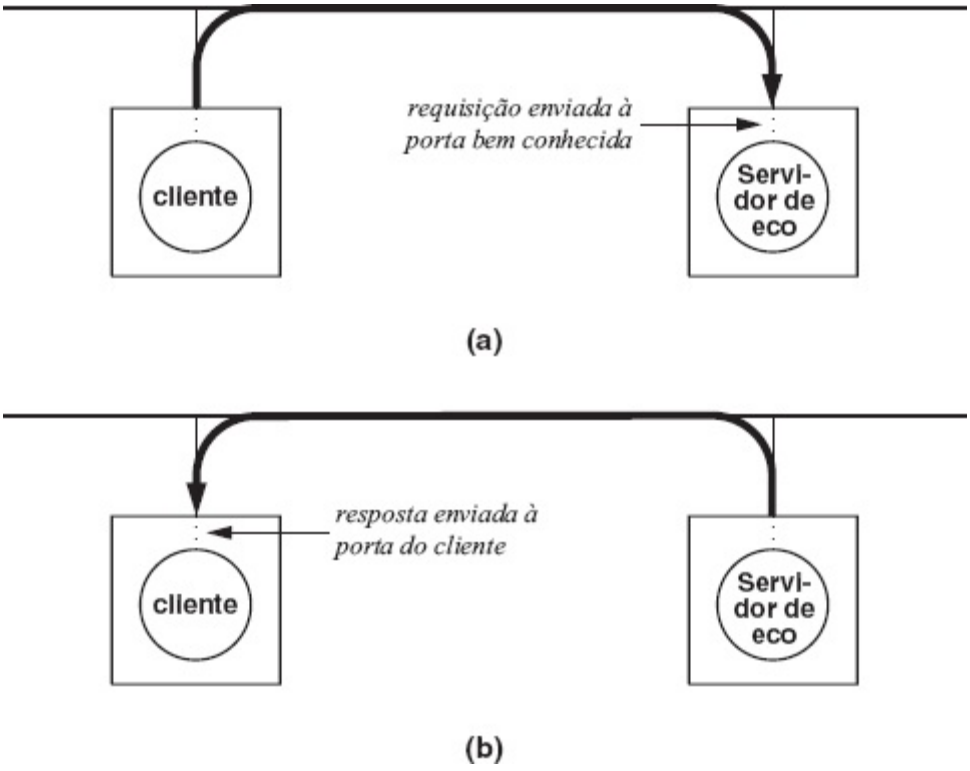


Figura 20.1 O serviço de eco UDP como um exemplo do modelo cliente-servidor. Em (a), o cliente envia uma solicitação para o endereço IP do servidor e porta UDP. Em (b), o servidor retorna uma resposta.

Quem usaria um serviço eco? Esse não é o tipo de serviço que o usuário comum acharia interessante. Entretanto, os programadores que projetam, implementam, medem ou modificam o protocolo de rede, ou gerentes de rede que testam rotas e depuram problemas de comunicação, normalmente usam os servidores de eco em testes. Por exemplo, um serviço eco pode ser usado para determinar se é possível alcançar uma máquina remota. Além disso, um cliente receberá de volta exatamente os mesmos dados que enviou. Um cliente pode verificar os dados em uma resposta para determinar se datagramas estão sendo danificados em trânsito.

A principal distinção entre clientes e servidores decorre do uso de números de porta de protocolo. Um servidor usa o número de porta bem conhecida associado com o serviço que oferece. Um cliente sabe que um servidor eco UDP usará a porta 7 porque os padrões a reservam para o serviço de eco. Um cliente, entretanto, não usa uma porta bem conhecida. Em vez disso, obtém uma porta de protocolo UDP não utilizada de seu sistema operacional local e usa o número desta como a porta de origem durante o envio de uma mensagem UDP. O cliente espera a resposta e o servidor usa a porta de origem em mensagens recebidas para enviar essa resposta de volta ao cliente correto.

O serviço eco UDP ilustra dois aspectos importantes que geralmente são verdadeiros sobre a interação cliente-servidor. O primeiro se refere à diferença entre o tempo de vida dos servidores e dos clientes.

> *Um servidor inicia a execução antes de a interação começar e continua a aceitar requisições e enviar respostas sem nunca terminar. Um cliente é qualquer programa que faz uma requisição e espera uma resposta; ele (normalmente) termina após usar um servidor por um número finito de vezes.*

O segundo aspecto, que é mais técnico, diz respeito ao uso de identificadores de porta reservados e não reservados.

> *Um servidor espera requisições em uma porta bem conhecida que tenha sido reservada para o serviço que ele oferece. Um cliente aloca uma porta arbitrária não usada e não reservada para sua comunicação.*

É importante perceber que a interação cliente-servidor requer apenas uma das duas portas (a usada por um servidor) a serem reservadas. A ideia é decisiva para o paradigma global, pois permite que vários aplicativos em um determinado computador se comuniquem com um servidor simultaneamente. Por exemplo, suponha que dois usuários estejam usando um grande sistema de *tempo compartilhado* (*timesharing*) simultaneamente. Suponha, ainda, que cada usuário execute um cliente eco UDP e todos eles enviem uma mensagem para o mesmo servidor eco UDP. Haverá confusão? Não, porque cada cliente obtém um número local de porta único e, assim, nunca há ambiguidade nas respostas do servidor.

20.4 Serviço de hora e data

O servidor de eco é extremamente simples, e pouco código é necessário para implementar o lado do servidor ou do cliente (desde que o sistema operacional ofereça um meio razoável de acessar os protocolos UDP/IP subjacentes). Nosso segundo exemplo, um servidor de hora, é igualmente trivial. No entanto, o exemplo mostra que mesmo aplicativos triviais podem ser úteis, e o serviço de hora coloca a questão de representação de dados.

Um servidor de hora resolve automaticamente a definição do relógio de um computador. Quando ele inicia, um computador entra em contato com um servidor de hora e usa a resposta para definir seu relógio local. Se for necessária precisão adicional, um computador poderá contatar um servidor de hora periodicamente.

20.4.1 Representação de dados e modelo Network Byte Order

Como a hora pode ser representada? Uma representação útil armazena a hora e a data em um único (e grande) inteiro indicando uma contagem de segundos desde uma data específica. Os protocolos TCP/IP definem a data inicial em 1º de janeiro de l900 e armazenam a hora em um inteiro de 32 bits. A representação acomoda todas as datas por algumas décadas no futuro. No momento em que um inteiro de 32 bits se esgotar, espera-se que a maioria dos computadores tenha capacidade para um inteiro de 64 bits.

Especifica-se de modo simples que um valor de um inteiro de 32 bits é insuficiente porque a representação de inteiros varia entre computadores. A maioria dos projetos de protocolo de aplicação segue a mesma abordagem que os protocolos TCP/IP: inteiros são representados em *ordem de byte de rede padrão* (*network standard byte order*). Ou seja, antes de enviar

uma mensagem, o aplicativo de envio traduz cada número inteiro da ordem de bytes da máquina local para network byte order, e, ao receber uma mensagem, o aplicativo receptor traduz cada inteiro do network byte order para a ordem de bit host local. Desse modo, dois computadores com diferentes representações de inteiros podem trocá-los sem ambiguidade.

A maioria dos aplicativos também segue os padrões TCP/IP quando escolhe o seu padrão de network byte order: eles usam representação *big endian*. Nela, o byte mais significativo do inteiro vem em primeiro lugar, seguido pelo próximo byte mais significativo, e assim por diante. Pode parecer que o uso de uma ordem de byte padrão de rede introduza uma sobrecarga extra ou que a escolha da ordem big endian seja insuficiente. No entanto, a experiência tem demonstrado que a sobrecarga envolvida na tradução entre ordem de byte local e network byte order é trivial em comparação com outros custos de processamento de mensagens. Além disso, usar um único e bem conhecido padrão de ordem de bytes evita muitos problemas e ambiguidades.

20.4.2 Interação do servidor de hora

A interação entre um cliente e um servidor de hora ilustra uma interessante troca dessa interação. Um servidor de hora opera de modo semelhante a um servidor eco. O servidor começa primeiro e espera para ser contatado. Entretanto, o protocolo de hora não define uma mensagem de solicitação. Em vez disso, o servidor de hora usa o tempo de chegada de uma mensagem UDP para desencadear uma resposta. Isto é, ele assume que *qualquer* mensagem UDP que chega, independentemente do seu tamanho ou conteúdo, é uma requisição de hora atual. Portanto, o servidor responde a cada mensagem recebida por meio do envio de uma resposta que contém a hora atual em um inteiro de 32 bits. A Figura 20.2 ilustra a interação.

Resumindo, a seguir, em poucas palavras.

> *O envio de um datagrama arbitrário a um servidor de hora equivale a fazer uma requisição da hora atual; o servidor responde retornando uma mensagem UDP que contém a hora atual em rede padrão de ordem de byte.*

20.5 Servidores sequenciais e concorrentes

Os exemplos anteriores ilustram os servidores sequenciais básicos (ou seja, um servidor que processa uma requisição de cada vez). Após aceitar uma requisição, um servidor sequencial envia uma resposta antes de aguardar outra requisição chegar. A ideia de servidores sequenciais levanta uma questão importante sobre software de protocolo: o que acontece se uma requisição posterior chega enquanto um servidor está ocupado tratando de uma requisição anterior?

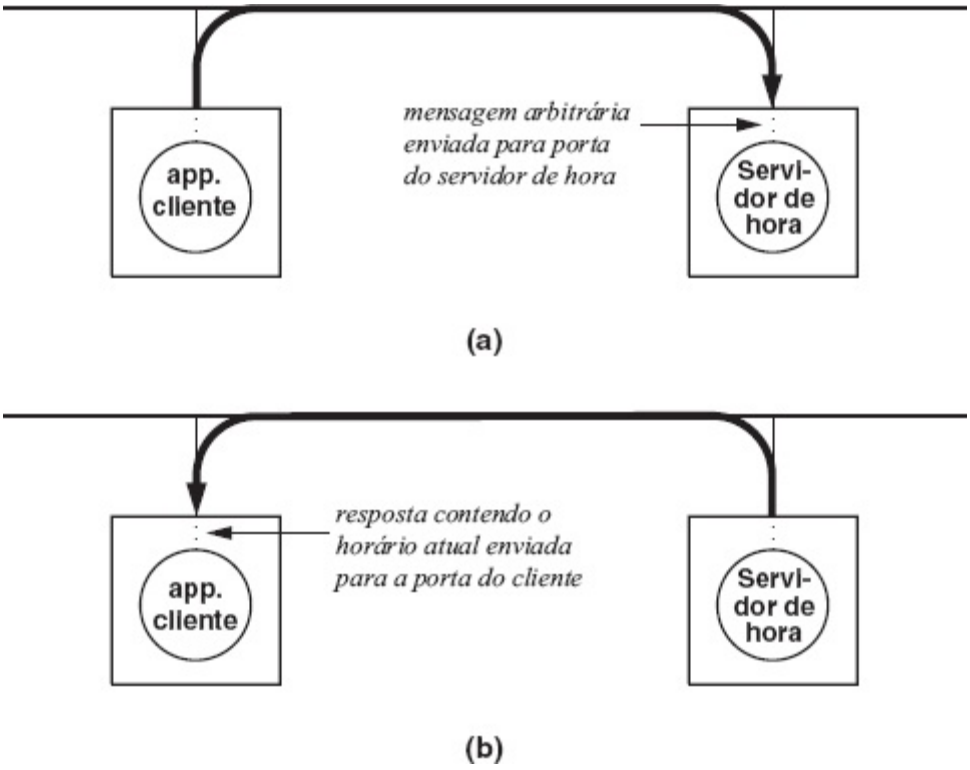


Figura 20.2 Ilustração da interação usada com um servidor de hora. O protocolo não define uma mensagem de requisição porque um datagrama UDP arbitrário irá engatilhar uma resposta.

Para nossos exemplos triviais, a questão é irrelevante. Para servidores como um servidor de download de vídeo, em que uma única solicitação pode levar minutos ou horas para atender, a questão torna-se importante. Em geral, devem-se projetar os servidores para atender à demanda esperada. Duas técnicas apresentadas a seguir podem ser usadas para acomodar muitas requisições.

- Os pedidos de entrada podem ser colocados em uma fila.
- Um servidor pode atender a várias requisições ao mesmo tempo.

Fila de requisições. Se um servidor sequencial está ocupado processando uma requisição quando um pedido subsequente chega, ele não pode colocar a requisição de entrada em uma fila. Infelizmente, pacotes tendem a chegar em pequenos grupos, o que significa que várias solicitações podem chegar em rápida sucessão. Para lidar com rajadas, o software de protocolo é projetado para fornecer uma fila a cada aplicação. Como filas são apenas destinadas a lidar com rajadas, os tamanhos típicos de filas são extremamente pequenos (por exemplo, alguns sistemas operacionais limitam uma fila para cinco ou menos entradas). Por isso, a fila só é suficiente para as aplicações em que o tempo de processamento esperado é insignificante.

Servidores concorrentes. Para lidar com várias requisições simultâneas, a maioria dos servidores de produção são *concorrentes*. Um servidor concorrente pode lidar com várias requisições "ao mesmo tempo". Usamos o termo concorrente, em vez de *simultâneo*, para enfatizar que todos os clientes compartilham os recursos computacionais da rede subjacente. Um servidor concorrente pode lidar com um arbitrariamente grande conjunto de clientes em um determinado momento, mas o serviço que cada cliente recebe degrada-se em proporção ao número de clientes.

Para entender por que a concorrência é importante, imagine o que aconteceria com um servidor sequencial se um cliente solicitasse um download de vídeo através de uma rede extremamente lenta. Nenhuma outra requisição de clientes poderia ser atendida. No entanto, com um projeto concorrente, um servidor irá honrar outros pedidos ao mesmo tempo em que continua a enviar pacotes através da lenta conexão.

A chave para um servidor concorrente reside na abstração do sistema operacional de um *processo concorrente* e na capacidade de criar processos dinamicamente. A Figura 20.3 apresenta os passos básicos que um servidor concorrente segue.

Abrir porta
O servidor abre a porta bem conhecida em que pode ser alcançado.

Espera pelo cliente
O servidor espera a requisição do próximo cliente chegar.

Iniciar cópia
O servidor inicia uma cópia independente e concorrente de

si mesmo para tratar a requisição (ou seja, um processo ou thread). A cópia trata uma requisição e, depois, termina.

Continuar

O servidor original retorna para a etapa esperar e continua aceitando novas requisições enquanto a cópia recém-criada trata a requisição anterior concorrentemente.

Figura 20.3 As etapas realizadas por um servidor concorrente, que permitem ao servidor manipular múltiplas requisições ao mesmo tempo.

20.6 Complexidade do servidor

A principal vantagem de um servidor concorrente é a capacidade de lidar com as requisições prontamente: um pedido que chega mais tarde não precisa esperar requisições que iniciaram mais cedo para completar. A maior desvantagem é a complexidade: um servidor concorrente é mais difícil de construir.

Além da complexidade que resulta de os servidores tratarem requisições concorrentes, também existe a complexidade causada pelo fato de que os servidores precisam impor regras de autorização e proteção. Como leem arquivos de sistema, mantêm logs e dados de acesso protegido, os aplicativos de servidor geralmente precisam ser executados com maior privilégio. Um programa privilegiado deve ser concebido com cuidado, porque o sistema operacional não irá restringir um programa de servidor se ele tentar acessar um usuário de arquivos arbitrários, um banco de dados arbitrários ou enviar um pacote arbitrário. Portanto, os servidores não podem aceitar cegamente requisições de outros sites. Em vez disso, cada servidor assume a responsabilidade de impor políticas de proteção e de acesso ao sistema. Por exemplo, um servidor de arquivos deve examinar um pedido e decidir se o cliente é autorizado a acessar o arquivo especificado.

Finalmente, os servidores são complexos para projetar e implementar porque precisam se proteger de requisições malformadas ou daquelas que farão o aplicativo do servidor abortar. Muitas vezes, é difícil prever possíveis problemas. Por exemplo, um projeto na Purdue University criou um servidor de arquivos que permitia que sistemas operacionais de alunos acessassem arquivos em um sistema de tempo compartilhado (*timesharing*) UNIX. Os alunos descobriram que requisitar que o servidor abrisse um arquivo chamado */dev/tty* fazia com que o servidor abortasse porque, no UNIX, o nome se refere ao terminal de controle que um programa está usando. Entretanto, como ele era disparado na inicialização,*

o servidor de arquivos não tinha terminal de controle. Portanto, uma tentativa de abrir o terminal de controle fazia o sistema operacional abortar o processo do servidor.

Podemos resumir nossa discussão dos servidores da forma a seguir.

Os servidores normalmente são mais difíceis de construir do que os clientes. Embora possam ser implementados como programas aplicativos, os servidores precisam impor todas as políticas de acesso e proteção do computador em que estão sendo executados, e precisam proteger a si mesmos contra todos os erros possíveis.

20.7 Difundindo uma requisição

Até agora, todos os nossos exemplos de interação cliente-servidor exigem que o cliente saiba o endereço completo do servidor. Em alguns casos, no entanto, um cliente não conhece o endereço do servidor. Por exemplo, ao inicializar, um computador pode usar o DHCP para obter um endereço, mas o cliente não sabe o endereço de um servidor.** Em vez disso, o cliente difunde sua requisição.

O principal ponto é descrito a seguir.

Para protocolos em que o cliente não sabe o local de um servidor, o modelo cliente-servidor permite que os programas clientes difundam requisições.

20.8 Alternativas e extensões de cliente-servidor

Nós dissemos que a interação cliente-servidor é a base para quase toda comunicação na Internet. No entanto, surge a pergunta: que variações são possíveis? Utilizam-se três abordagens gerais que são apresentadas a seguir.

- Proxy cache.
- Prefetching (pré-busca).
- Acesso ponto a ponto.

Proxy cache. Dissemos que o paradigma convencional cliente-servidor requer um aplicativo para se tornar um cliente e contatar um servidor

sempre que ele precisa de informações a partir do próprio servidor. No entanto, a latência pode ser um problema, especialmente em casos em que o servidor está distante do cliente. Se as requisições são susceptíveis de serem repetidas, o cache pode melhorar a performance da interação cliente-servidor, reduzindo a latência e o tráfego na rede. Por exemplo, considere um conjunto de funcionários de uma empresa com acesso à web. Caso um funcionário encontre uma página da web útil ou interessante, ele deverá passar a URL para alguns poucos amigos que, em seguida, visualizam a página e passam a URL para outros amigos. Desse modo, uma determinada página da web pode ser acessada uma dúzia de vezes. Com uma abordagem convencional cliente-servidor, cada acesso requer que a página seja buscada a partir do servidor.

Para melhorar o desempenho, a empresa pode instalar um *proxy cache web*. Quando recebe uma requisição, o proxy cache olha para o seu disco para ver se o item solicitado está disponível. Se não estiver, os contatos de proxy de um servidor web apropriado para obter o item colocarão uma cópia em disco, e retornarão uma cópia ao browser que fez a requisição. Como o proxy pode satisfazer cada pedido posterior de seu disco local, o servidor web só é contatado uma vez. É óbvio que os clientes devem concordar em usar o proxy ou a abordagem não vai funcionar. Ou seja, cada usuário deve configurar seu browser para contatar o proxy para requisições (a maioria dos browsers da web tem uma configuração que usa um proxy). Os indivíduos são fortemente motivados para o uso de um proxy, pois este não injeta uma sobrecarga significativa e pode melhorar consideravelmente o desempenho.

Há outros exemplos de cache do cliente-servidor. Por exemplo, o protocolo ARP apresentado no Capítulo 6 também segue o modelo cliente-servidor. ARP usa um cache para evitar requisições repetidamente ao endereço MAC de um vizinho. Se ARP não usasse um cache, o tráfego de rede seria o dobro.

Prefetching. Embora o cache melhore o desempenho, ele não muda a essência da interação cliente-servidor – a informação é buscada somente quando o primeiro cliente faz uma requisição. Isto é, um aplicativo executa até que necessite de informações e, em seguida, atua como um cliente para obtê-las. Ter uma visão do mundo baseada em demanda é natural e surge da experiência. O uso de cache ajuda a amenizar o custo de obter informações reduzindo o custo de recuperação para buscas subsequentes, mas não reduz o custo da primeira busca.

Como podemos diminuir o custo da recuperação de informações para a

requisição inicial? A resposta está no *prefetching* – organizar para coletar e armazenar informações *antes* que qualquer programa específico as requisite. O prefetching reduz a latência para a requisição inicial. Mais importante, a pré-coleta de informações significa que um cliente pode obter uma resposta mesmo que a rede esteja temporariamente desconectada ou congestionada quando o cliente fizer a requisição.

Um antigo programa UNIX chamado *ruptime* ilustra a ideia de prefetching que agora é usado em muitos centros de sistemas de gerenciamento de dados. Esse programa fornecia a carga da CPU de todos os computadores na rede local. Um cliente ruptime sempre operava instantaneamente porque a informação era pré-buscada por um aplicativo de segundo plano. Cada computador na rede iria difundir sua carga atual periodicamente, e os programas de segundo plano coletavam avisos enviados por outros computadores. Informações sobre a performance de pré-coleta são importantes porque um computador sobrecarregado não pode responder a uma requisição rapidamente.

A pré-coleta tem duas desvantagens. Primeiro, ela usa processador e recursos de rede, mesmo que nenhum cliente vá acessar os dados que estão sendo coletados. No nosso exemplo de performance, cada máquina deve participar transmitindo seu status e coletando transmissões (broadcast) de outras máquinas. Se apenas algumas máquinas participaram da transmissão, os custos de pré-coleta serão insignificantes. Em um aglomerado de centro de dados de grandes dimensões que inclui centenas de máquinas, o tráfego de transmissão gerado pela pré-coleta pode impor carga significativa na rede. Desse modo, a pré-coleta é reservada a casos especiais, em que a sobrecarga de custos de processamento e de rede pode ser limitada.

Acesso ponto a ponto (peer-to-peer). A terceira variação de interação cliente-servidor é conhecida como redes ponto a ponto (*peer-to-peer – P2P*). O P2P foi popularizado por aplicativos de compartilhamento de arquivos que permitem aos usuários a troca de arquivos, tais como os de MP3 que contenham músicas ou vídeos.

A ideia por trás da abordagem ponto a ponto é simples: em vez de um único servidor, muitos servidores recebem uma cópia de cada arquivo, o que permite a um usuário fazer o download a partir do servidor mais próximo. A mesma ideia é usada pelas tecnologias *Content Distribution Network* (*CDN*), como a pioneira da Akami. No entanto, as redes ponto a

ponto acrescentam uma característica interessante: em vez de computadores servidores especiais, um sistema ponto a ponto confia nos computadores dos usuários. Ou seja, em troca de acesso a arquivos, o usuário concorda em permitir que seu computador seja usado como um servidor. Quando um cliente faz uma requisição, o sistema ponto a ponto sabe o conjunto de usuários que fez o download de uma cópia e seleciona aquele que dará acesso mais rápido para o novo pedido. Alguns sistemas de arquivos ponto a ponto iniciam o download a partir de um punhado de localizações e, em seguida, usam o download que prossegue mais rápido. Outros sistemas ponto a ponto dividem arquivos em blocos e organizam um bloco para ser baixado a partir de um único local, enquanto outros blocos são transferidos para outros locais. Em qualquer caso, uma vez que um usuário tenha baixado um item (ou seja, um arquivo completo ou um bloco de arquivo), seu computador torna-se uma fonte potencial do item para outros usuários baixarem.

20.9 Resumo

Muitos aplicativos modernos usam redes de computadores e a Internet para se comunicar. O padrão primário de uso é conhecido como cliente-servidor. Um servidor aguarda uma requisição, realiza ações com base nela e retorna uma resposta. Um programa cliente formula uma requisição, envia-a a um servidor e aguarda uma resposta. Alguns clientes enviam requisições diretamente, enquanto outros as transmitem; o broadcasting é especialmente útil quando um aplicativo não sabe o endereço de um servidor.

Examinamos alguns exemplos triviais de clientes e servidores, como um serviço de hora e um de eco UDP. O primeiro ilustra a importância do modelo network byte order e, além disso, mostra que um serviço não precisa definir um formato de requisição.

Embora os serviços triviais usem uma abordagem sequencial, a maioria dos servidores de produção permite processamento concorrente. Porque isso cria um processo separado para lidar com cada requisição, um servidor concorrente não requer um cliente que espere pedidos anteriores para ser servido. A concorrência é especialmente importante para os serviços, tais como download de vídeo, em que se pode demorar minutos ou horas para satisfazer uma única solicitação.

Nós consideramos alternativas e extensões para o paradigma cliente-servidor, incluindo o armazenamento em cache, prefetching e interações ponto a ponto. Cada técnica pode aumentar a performance, dependendo do padrão de requisições repetidas e do tempo necessário para acessar os

dados.

EXERCÍCIOS

20.1 Construa um cliente de eco UDP que envie um datagrama para um servidor de eco especificado, espere uma resposta e a compare com a mensagem original.

20.2 Considere cuidadosamente como um servidor de eco UDP forma uma resposta. Sob que condições é *incorreto* criar novos endereços IP invertendo os endereços IP de origem e de destino?

20.3 Embora a maioria dos servidores seja implementada por programas aplicativos separados, um servidor de eco ICMP é normalmente construído no software de protocolo no sistema operacional. Quais são as vantagens e as desvantagens de ter um programa aplicativo (processo de usuário) por servidor?

20.4 Suponha que você não saiba o endereço IP de uma máquina local executando um servidor de eco UDP, mas saiba que ela responde a requisições enviadas para a porta 7. Existe um endereço IP que você possa usar para alcançá-la? Explique.

20.5 Construa um cliente UDP para o tempo de hora de Internet e demonstre que ele funciona corretamente.

20.6 Pode um servidor executar no mesmo computador host físico como um cliente? Explique.

20.7 Considere um agrupamento de centro de dados com 200 computadores. Se cada computador transmite sua carga atual a cada 5 segundos e cada mensagem contém 240 octetos de informação (mais os cabeçalhos associados), quanto da capacidade da rede é utilizado pelas transmissões?

20.8 Que servidores estão sendo executados nos computadores em seu local? Se você não tem acesso aos arquivos de configuração do sistema que listam os servidores iniciados para um determinado computador, veja se seu sistema possui um comando que imprima uma lista das portas TCP e UDP abertas (por exemplo, o utilitário *netstat* do UNIX).

20.9 Alguns servidores permitem que um gerente os desligue ou reinicie elegantemente. Qual é a vantagem do desligamento elegante do servidor?

20.10 Suponha que um servidor concorrente segue o algoritmo apresentado

na Figura 20.3 (na página 289). Que vulnerabilidade o servidor exibe?

---

* Um dos exercícios sugere considerar este passo mais detalhadamente.

* *Nota do revisor técnico*: O arquivo */dev/tty* é aberto no momento em que o usuário inicia uma sessão, local ou remota. No caso citado, não havia uma sessão que associasse um terminal, acarretando o erro.

** O Capítulo 22 examina o DHCP.

CAPÍTULO 21

A API socket

CONTEÚDOS DO CAPÍTULO

21.1 Introdução

Os capítulos anteriores discutem os princípios e conceitos que fundamentam os protocolos de TCP/IP, e o capítulo anterior considera o paradigma cliente-servidor usado pelos aplicativos para se comunicar através de uma internet TCP/IP. Os capítulos, no entanto, omitem um detalhe importante: eles não especificam a interface exata que programas aplicativos usam para interagir com o software de protocolo. Este capítulo completa a discussão ao considerar uma *Application Program Interface* (*API*), que se tornou, de fato, um padrão para a Internet. O capítulo descreve a abordagem geral adotada e analisa o suficiente das funções API para explicar um exemplo. Assim, abordamos muitos detalhes e nos concentramos em entender o básico; isso nos ajudará a compreender o código necessário para uma trivial aplicação cliente-servidor. O Volume 3 amplia a discussão, mostrando mais detalhes e mais exemplos de aplicações cliente e servidor que usam a API.

Existem duas razões que nos levaram a adiar a discussão das APIs até este capítulo. Primeiro, os padrões TCP/IP não especificam a interface exata que os aplicativos usam para acessar os serviços de rede; os detalhes dependem do sistema operativo. Segundo, é importante distinguir entre a funcionalidade que os protocolos fornecem e a funcionalidade que é disponibilizada por meio de uma interface específica. Por exemplo, o TCP é

projetado para lidar com um caso extremo, em que cada uma de duas extremidades tenta formar uma conexão TCP simultaneamente. Entretanto, nenhum dos APIs amplamente utilizados permitia isso.

21.2 Versões da API socket

Examinaremos a *API socket*, que informalmente é chamada de *socket*. A interface socket foi criada originalmente como parte do sistema operacional BSD Unix. Versões sockets aparecem em sistemas operacionais BSD mais recentes, Linux e Mac OS X; a Microsoft adaptou uma versão de sockets conhecida como *Windows Sockets** para seus sistemas operacionais.

O capítulo apresenta uma visão geral da API socket que se aplica a todos os sistemas e mostra um exemplo básico que segue o estilo BSD.

21.3 O paradigma de E/S do UNIX e a E/S da rede

Desenvolvido no final da década de l960 e início da década de l970, o UNIX foi projetado originalmente como um sistema de tempo compartilhado (timesharing) para computadores de único processador. Ele é um sistema operacional orientado a processo, em que cada programa aplicativo é executado como um processo em nível de usuário. Um programa aplicativo interage com o sistema operacional fazendo *chamadas do sistema*. Do ponto de vista do programador, as chamadas do sistema veem e se comportam exatamente como outras funções de chamadas. Uma chamada de sistema pode apanhar argumentos e retornar um ou mais resultados. Os argumentos podem ser valores (por exemplo, inteiros) ou ponteiros para objetos no programa aplicativo (por exemplo, um buffer a ser preenchido com caracteres).

Derivadas daquelas no Multics e em sistemas mais antigos, as primitivas de entrada e saída (E/S; em inglês, input and output I/O) do UNIX seguem um paradigma às vezes referenciado como *open-read-write-close*. Antes que um aplicativo possa realizar operações de E/S, ele chama *open* para especificar o arquivo ou dispositivo a ser usado e obtém permissão. A chamada a *open* retorna um *pequeno descritor de arquivo inteiro*** que o aplicativo usa para executar operações E/S. Uma vez que tenha aberto um arquivo (ou dispositivo), um aplicativo invoca as operações *read* ou *write* para transferir dados. A chamada tem argumentos que especificam o descritor a usar, o endereço de um buffer e o número de bytes a serem

transferidos. Depois que todas as operações de transferência estiverem completas, o processo do usuário chama *close* para informar ao sistema operacional que terminou de usar o arquivo ou o dispositivo.

21.4 Acrescentando E/S de rede ao UNIX

O grupo que acrescentou protocolos de rede ao BSD UNIX decidiu fazer duas opções de projetos. A primeira delas surgiu a partir da rica funcionalidade dos protocolos de rede. Como os protocolos de rede oferecem muito mais possibilidades do que dispositivos convencionais e arquivos, a interação entre aplicativos e protocolos de rede precisou especificar novas funções. Por exemplo, uma interface de protocolo deve permitir aos programadores que criem tanto o código do servidor (que espera conexões passivamente) quanto o código do cliente (que forma conexões ativamente). Além disso, um programa aplicativo que envia um datagrama pode querer especificar o endereço de destino, juntamente com cada datagrama, em vez de vincular o endereço destino para o socket. Para lidar com todos os casos, os projetistas escolheram abandonar o paradigma open-read-write-close tradicional do UNIX e acrescentaram vários novos sistemas de chamadas. O projeto aumentou a complexidade da interface de E/S (I/O), substancialmente, mas era necessário.

A segunda decisão de projeto surgiu porque havia muitos protocolos, e não era óbvio que o TCP/IP seria tão bem-sucedido. Assim, os projetistas decidiram criar um mecanismo geral para acomodar todos os protocolos. Por exemplo, a generalidade torna possível que o sistema operacional inclua software para outros pacotes de protocolo, assim como TCP/IP, e permita que um programa aplicativo use um ou mais deles de cada vez. Como consequência, o programa aplicativo não pode simplesmente fornecer um valor binário e esperar que os protocolos o interpretem como um endereço IP. Em vez disso, um aplicativo deve especificar o tipo de endereço (ou seja, a família de endereços) explicitamente. A generalidade valeu a pena para o IPv6 – em vez de redesenhar a interface socket, engenheiros precisaram simplesmente adicionar opções para endereços IPv6.

21.5 A abstração e as operações socket

A base para a E/S de rede na API socket gira em torno de uma abstração conhecida como *socket*. Pensamos em um socket como um mecanismo que oferece um aplicativo com um descritor que pode ser usado para

comunicação em rede. Sockets são dinâmicos – um programa aplicativo solicita um socket quando isso é necessário e o libera quando ele tiver terminado a execução de E/S (I/O).

Sockets compartilham uma coisa com outra E/S – a um socket é dado um descritor como um arquivo aberto. Na maioria dos sistemas, um único conjunto de descritores é usado. Assim, descritores 5 e 7 podem corresponder a um arquivo aberto, e descritor 6 pode corresponder a um soquete de uma conexão TCP.

21.5.1 Criando um socket

A função *socket* cria sockets por demanda. Ela utiliza três argumentos inteiros e retorna um descritor inteiro:

```
descritor = socket(pfam, tipo, protocolo)
```

O argumento *pfam* especifica a família de protocolos a ser usada com o socket (ou seja, ele especifica como interpretar endereços). As famílias mais importantes são IP versão 4 (*PF_INET*)* e IP versão 6 (*PF_INET6*).

O argumento *tipo* especifica o tipo de comunicação desejado. Os tipos possíveis incluem o serviço de entrega de fluxo confiável (*SOCK_STREAM*), o serviço de remessa de datagrama sem conexão (*SOCK_DGRAM*) e um tipo bruto (*SOCK_ RAW*), que permite que programas privilegiados acessem protocolos de baixo nível ou interfaces de rede.

Como uma única família de protocolos pode ter vários protocolos que oferecem o mesmo tipo de comunicação, a chamada *socket* possui um terceiro argumento, que pode ser usado para selecionar um protocolo específico; se a família de protocolos só tiver um protocolo de determinado *tipo* (por exemplo, somente TCP fornece um serviço *SOCK_STREAM* para IPv4 e IPv6), o terceiro argumento pode ser definido como zero.

21.5.2 Herança e término de socket

Em sistemas UNIX, as chamadas do sistema *fork* e *exec* são usadas para criar um processo executando um programa aplicativo específico. Na maioria dos sistemas, quando um novo processo é criado, ele herda o acesso para os sockets abertos. Um servidor concorrente utiliza a herança de

socket para criar um novo processo a fim de lidar com um novo cliente.

Tanto o processo antigo como o novo compartilham direitos de acesso para descritores existentes. Assim, ambos podem acessar o socket para um determinado cliente. Desse modo, é responsabilidade do programador garantir que os dois processos usem o socket compartilhado significativamente.

Quando um processo termina de usar um socket, ele chama o *close*, que tem este formato:

```
close(descriptor)
```

O argumento *descriptor* especifica um descritor de um socket a ser fechado. Quando um processo termina por qualquer motivo, o sistema fecha todos os sockets que permanecem abertos. Internamente, uma chamada a *close* decrementa o contador de referência para um socket e destrói o socket se a contagem atinge zero.

21.5.3 Especificando um endereço local

Inicialmente, um socket é criado sem qualquer associação com endereços locais ou de destino. Para os protocolos TCP/IP, isso significa que um novo socket não começa com endereços IP locais ou remotos, ou números de porta de protocolo. Os programas cliente não se importam com o endereço local que utilizam, e desejam permitir que o protocolo preencha o endereço IP do computador e escolha um número de porta. Porém, os processos servidores que operam em uma porta bem conhecida precisam ser capazes de especificar essa porta ao sistema. Quando um socket tiver sido criado, um servidor utilizará a função *bind* para estabelecer um endereço local para ele.** *Bind* tem o seguinte formato:

```
bind(descriptor, localaddr, addrlen)
```

O argumento descritor (*descriptor*) é o descritor inteiro do socket a ser vinculado. O argumento *localaddr* é uma estrutura que especifica o ponto de extremidade local ao qual o socket deve ser vinculado, e o argumento *addrlen* é um inteiro que especifica o tamanho da estrutura medida em bytes. Em vez de dar o ponto de extremidade apenas como uma se quência de bytes, os projetistas optaram por definir uma estrutura. A Figura 21.1

ilustra a estrutura *sockaddr_in* usada para pontos de extremidade IPv4.

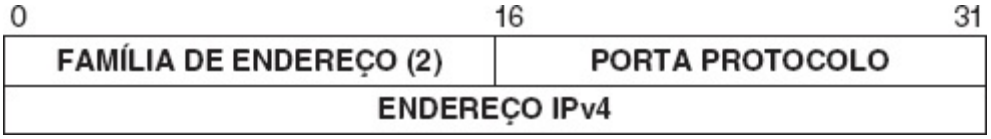


Figura 21.1 A estrutura *sockaddr _in* usada ao passar de um ponto de extremidade IPv4 para uma função socket.

A estrutura começa com um campo FAMÍLIA DE ENDEREÇO de l6 bits que identifica o conjunto de protocolos ao qual o endereço pertence; cada família de protocolo define o layout do resto da estrutura. O valor 2 no campo FAMÍLIA DE ENDEREÇO indica que a estrutura é usada para IPv4 e, portanto, o restante dela é constituído por um número da porta do protocolo de 16 bits e um endereço IPv4 de 32 bits. Quando passada como um argumento, a estrutura deve ser convertida para uma estrutura genérica, *sockaddr*.

Para endereços IPv6, um aplicativo pode precisar fornecer duas peças adicionais de informação: um identificador para um fluxo IPv6 ou o escopo de um endereço (por exemplo, link local, site local ou global). A Figura 21.2 ilustra a estrutura *sockaddr_in6* usada para representar um ponto de extremidade IPv6.

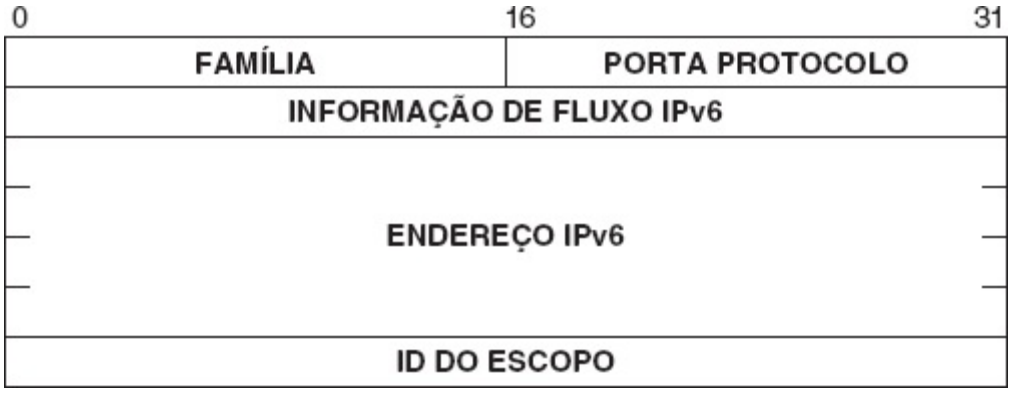


Figura 21.2 A estrutura *sockaddr_in6* usada ao passar um ponto de extremidade IPv6 para uma função socket.

Embora seja possível especificar valores arbitrários na estrutura de endereço ao chamar *bind*, nem todos os vínculos possíveis são válidos. Por exemplo, quem chama poderia requisitar uma porta de protocolo local que

já está em uso por outro programa, ou poderia requisitar um endereço IP inválido. Nesses casos, a chamada *bind* falha e retorna um código de erro.

21.5.4 Conectando um socket a um ponto de extremidade de destino

Inicialmente, um socket é criado no *estado não conectado*, o que significa não está associado a qualquer destino remoto. A função *connect* vincula um terminal remoto permanente a um socket, colocando-o no *estado conectado*. Um programa aplicativo precisa chamar *connect* para estabelecer uma conexão antes que possa transferir dados por um socket de fluxo confiável. Os sockets usados com serviços de datagrama sem conexão não precisam estar conectados antes de serem usados, mas isso possibilita transferir dados sem especificar o destino a cada vez.

A função *connect* tem o seguinte formato:

```
connect(descriptor, destaddr, addrlen)
```

O argumento *descriptor* é o descritor inteiro do socket a ser conectado. O argumento *destaddr* é uma estrutura de endereço de socket que especifica o endereço de destino ao qual o socket deve ser vinculado. O argumento *addrlen* especifica o tamanho do endereço de destino, medido em bytes.

A semântica de *connect* depende dos protocolos subjacentes. A seleção do serviço de entrega de fluxo confiável na família PF_INET ou PF_INET6 significa a escolha do TCP. Nesses casos, *connect* monta uma conexão TCP com o destino e retorna um erro se não puder. No caso do serviço sem conexão, *connect* não faz nada mais do que armazenar o ponto de extremidade de destino localmente.

21.5.5 Enviando dados por um socket

Quando um programa aplicativo estabelece um socket, ele pode usá-lo para transmitir dados. Existem cinco funções possíveis para escolher: *send*, *sendto*, *sendmsg*, *write* e *writev*. *Send*, *write* e *writev* só funcionam com sockets conectados, pois não permitem que quem chama especifique um

endereço de destino. As diferenças entre as três são pequenas. Uma chamada para enviar (send) utiliza quatro argumentos:

```
send(descriptor, buffer, length, flags)
```

O argumento *descriptor* é um descritor de socket inteiro, o argumento *buffer* contém o endereço dos dados a serem enviados, o argumento *length* especifica o número de bytes a enviar, e o argumento *flags* controla a transmissão. Um valor para *flags* permite ao emissor especificar que os dados devem ser enviados fora de faixa (por exemplo, dados urgentes do TCP). Uma chamada *to send* bloqueia até que os dados possam ser transferidos (por exemplo, ela bloqueia se buffers internos do sistema para a tomada estão cheios). Outro valor para *flags* permite a quem chama requisitar que a mensagem seja enviada sem o uso da tabela de encaminhamento local. A intenção é permitir a quem chama assumir o controle do encaminhamento, tornando possível escrever um software de depuração de rede. Claro está que nem todos os sockets suportam todas as requisições de programas arbitrários. Algumas requisições exigem um programa que tenha privilégios especiais; outras simplesmente não são suportadas em todos os sockets. Como a maioria dos sistemas de chamadas, *send* retorna um código de erro para o aplicativo de chamada, permitindo que o programador saiba se a operação teve êxito.

21.5.6 Recebendo dados por um socket

Semelhante às cinco operações de saída diferentes, a API socket provê cinco funções que um processo pode usar para receber dados por um socket: *read*, *readv*, *recv*, *recvfrom* e *recvmsg*. As operações de entrada *recv* e *read* só podem ser usadas quando o socket está conectado. *Recv* tem o formato:

```
recv(descriptor, buffer, length, flags)
```

O argumento *descriptor* especifica um descritor de socket a partir do qual os dados devem ser recebidos. O argumento *buffer* especifica o endereço na memória em que a mensagem deverá ser colocada, o argumento *length* especifica o tamanho da área do buffer, o argumento

flags permite que quem chama controle a recepção. Entre os possíveis valores para o argumento *flags*, está um que permite a quem chama olhar em frente, extraindo uma cópia da próxima mensagem de entrada sem remover a mensagem do socket.

Para formar uma resposta, um servidor UDP precisa obter mais do que o conteúdo de carga UDP – ele também deve obter o endereço IP do emissor e o número da porta do protocolo. Para isso, ele usa a função socket *recvfrom*. A chamada tem o formato:

recvfrom(descriptor, buffer, length, flags, fromaddr, fromlen)

Os dois argumentos adicionais, *fromaddr* e *fromlen*, são ponteiros para uma estrutura de endereço de socket e um inteiro. O sistema operacional registra informações do ponto de extremidade do emissor no local *fromaddr* e o tamanho da informação do ponto de extremidade no local *fromlen*. Quando envia uma resposta, um servidor UDP pode passar as informações de ponto de extremidade para a função *sendto*. Portanto, a formação de uma resposta é simples.

21.5.7 Obtendo informação de extremidade socket local e remota

Dissemos que processos recém-criados herdam o conjunto de sockets abertos do processo que os criou. Às vezes, um processo recém-criado precisa extrair o endereço remoto do ponto de extremidade a partir de um socket que foi herdado. Além disso, porque o sistema operativo preenche a informação do ponto de extremidade local automaticamente, um processo pode ter de determinar o endereço do ponto de extremidade local que é usado para um socket. Duas funções oferecem essa informação: *getpeername* e *getsockname* (apesar dos nomes das funções, ambas lidam com o que chamamos de pontos de extremidades de endereços, e não nomes de domínios).

Um processo chama *getpeername* para determinar o ponto de extremidade do peer (ou seja, a aplicação remota à qual um socket se conecta). A chamada tem o seguinte formato:

getpeername(descriptor, destaddr, addrlen)

O argumento *descriptor* especifica o socket para o qual o destino do ponto de extremidade é desejado. O argumento *destaddr* é um ponteiro para uma estrutura do tipo *sockaddr* (veja Figuras 21.1 e 21.2) que receberá informação do ponto de extremidade.

Finalmente, o argumento *addrlen* é um ponteiro para um inteiro que receberá o tamanho da estrutura do ponto de extremidade. A *getpeername* só funciona com sockets conectados.

A função *getsockname* retorna o local do ponto de extremidade associado com um socket. A chamada tem este formato:

```
getsockname(descriptor, localaddr, addrlen)
```

Conforme esperado, o argumento *descriptor* especifica o socket para o qual o local do ponto de extremidade é desejado. O argumento *localaddr* é um ponteiro para uma estrutura do tipo *sockaddr*, que conterá o ponto de extremidade, e o argumento *addrlen* é um ponteiro para um inteiro que conterá o tamanho da estrutura do ponto de extremidade.

21.6 Obtendo e definindo as opções socket

Além de vincular um socket a um endereço local ou conectá-lo a um endereço de destino, surge a necessidade de um mecanismo que permita que os programas aplicativos controlem o socket. Por exemplo, ao usar protocolos que usam timeout e retransmissão, o programa aplicativo pode querer obter ou definir os parâmetros de timeout. O aplicativo também pode querer controlar a alocação de espaço de buffer, determinar se o socket permite a transmissão de broadcast, ou controlar o processamento de dados fora de faixa. Em vez de incluir novas funções para cada nova operação de controle, os projetistas decidiram criar um único mecanismo, que possui duas operações: *getsockopt* e *setsockopt*.

A função *getsockopt* permite que a aplicação requisite informações sobre o socket. Quem chama especifica o socket, a opção de interesse e um local em que a informação requisitada será armazenada. O sistema operacional examina suas estruturas de dados internas para o socket e passa a informação requisitada a quem chamou. A chamada tem este formato:

```
getsockopt(descriptor, level, optionid, optionval, length)
```

O argumento *descriptor* especifica o socket para o qual as informações são necessárias. O *level* do argumento identifica se a operação se aplica ao próprio socket ou aos protocolos subjacentes sendo usados. O argumento *optionid* especifica uma única opção à qual a requisição se aplica. O par de argumentos *optionval* e *length* especifica dois ponteiros. O primeiro indica o endereço de um buffer no qual o sistema coloca o valor requisitado e o segundo indica o endereço de um inteiro no qual o sistema coloca o tamanho do valor da opção.

A função *setsockopt* permite que um programa aplicativo defina uma opção de socket usando o conjunto de valores obtidos com *getsockopt*. Quem chama especifica um socket para o qual a opção deve ser definida, a opção a ser mudada e um valor para a opção. A chamada a *setsockopt* tem o formato

```
setsockopt(descriptor, level, optionid, optionval, length),
```

no qual os argumentos são como aqueles para *getsockopt*, exceto que o argumento *length* é um número inteiro que especifica o número de bytes na opção sendo passada para o sistema. Quem chama precisa fornecer um valor válido para a opção, além de um tamanho correto para esse valor. Naturalmente, nem todas as opções se aplicam a todos os sockets. A exatidão e a semântica das requisições individuais dependem do estado atual do socket e dos protocolos subjacentes sendo utilizados.

21.6.1 Especificando um tamanho de fila para um servidor

Dissemos que o sistema operacional mantém uma fila de requisições de entrada. Uma fila é especialmente importante para os servidores sequenciais, mas também pode ser necessária para lidar com rajadas de pacotes para servidores concorrentes. Uma das opções que se aplicam a sockets é usada com tanta frequência que uma função separada foi dedicada a ela: definir o tamanho da fila.

A função *listen* do socket permite que os servidores preparem um socket para as conexões que chegam. Em termos de protocolos subjacentes, *listen* coloca o socket em um modo passivo, pronto para aceitar conexões. Somente servidores usam *listen*. Além disso, ao colocar um protocolo em

modo passivo, *listen* contém um argumento que configura o tamanho de uma fila de entrada de requisições. A chamada tem o seguinte formato:

```
listen(descriptor, qlength)
```

O argumento *descriptor* indica o descritor de um socket que deve estar preparado para uso por um servidor, e o argumento *qlength* especifica o tamanho da fila de requisição para esse socket. Após a chamada, o sistema enfileirará até *qlength* requisições para conexões. Se a fila estiver cheia quando uma requisição chegar, o sistema operacional recusará a conexão, descartando a requisição. *Listen* se aplica apenas a sockets que selecionaram o serviço de entrega de fluxo confiável.

21.7 Como um servidor aceita conexões

Como visto anteriormente, o processo servidor utiliza as funções *socket*, *bind* e *listen* para criar um socket, vinculá-lo a um protocolo de porta bem conhecido e especificar o tamanho da fila de requisições de conexão. Note que a chamada *bind* associa o socket a um protocolo bem conhecido, mas o socket não está conectado a um destino específico. De fato, o destino precisa ser especificado com um *curinga*, permitindo que o socket receba requisições de conexão de um cliente arbitrário.

Uma vez estabelecido o socket, o servidor necessita aguardar uma conexão. Para isso, o utiliza a função *accept*. Uma chamada à função *accept* causa um bloqueio até que chegue uma requisição de conexão. A chamada apresenta o seguinte formato:

```
newsock = accept(descriptor, addr, addrlen)
```

O argumento *descriptor* especifica o descritor que se aguarda. O argumento *addr* é o ponteiro para uma estrutura do tipo *sockaddr*, e *addrlen* é um ponteiro para um inteiro.

Quando uma requisição chega, o sistema preenche argumento *addr* com a informação do ponto de extremidade do cliente que fez a requisição coloca em *addrlen* o tamanho da estrutura do ponto de extremidade. Por

fim, o sistema cria um novo socket que tem seu destino conectado com o cliente que efetuou a requisição e retorna um novo descritor socket para quem o invocou. O socket original continua com seu destino remoto curinga e continua aberto (open). Dessa forma, o servidor original pode continuar a aceitar (accept) requisições adicionais no socket original.

Quando uma requisição de conexão chega, ocorre o retorno da chamada *accept*. O servidor pode tanto tratar a requisição ele mesmo ou usar uma abordagem concorrente. Se ele mesmo tratar a requisição, o servidor envia uma resposta, fecha o novo socket e, em seguida, chama *accept* para obter a próxima requisição de conexão. No enfoque concorrente, após retornar da função *accept*, o servidor cria um novo processo para lidar com a requisição (na terminologia UNIX, ele invoca a system *call fork* para criar um processo filho que tratará a requisição). O processo filho herda uma cópia do novo socket e assim pode proceder ao atendimento da requisição. Quando termina, o filho fecha o socket e conclui. Entretanto, o processo servidor original fecha sua cópia do novo socket (depois de iniciar o filho) e continua a chamar *accept* para obter uma nova conexão de requisição.

O projeto concorrente pode parecer confuso, já que múltiplos processos utilizarão a mesma porta de protocolo. A chave para entender o mecanismo reside na forma como os protocolos básicos tratam as portas de protocolos. Lembre-se de que, no TCP, um par de extremos define uma conexão. Assim, não importa quantos processos utilizam uma dada porta de protocolo local, contanto que se conectem a diferentes destinos. No caso de um servidor concorrente, existe um processo por cliente e um processo adicional que aceita conexões. Quando *accept* retorna ao novo socket, este terá um específico ponto de extremidade remoto. Ao chegar um segmento TCP, o software de protocolo irá enviar o segmento ao socket que já está conectado à origem do segmento. Se não existir tal socket, o segmento será enviado para o socket que tem o destino remoto curinga. Como o socket com o destino remoto curinga não tem uma conexão aberta, ele somente irá honrar os segmentos TCP que requisitarem uma nova conexão (por exemplo, um segmento SYN); todos os outros serão descartados.

21.8 Servidores que tratam múltiplos serviços

A API socket fornece outra possibilidade interessante para projeto de servidores porque permite a um processo aguardar conexões em múltiplos sockets. O sistema de chamada que torna esse projeto possível chama-se

select e é aplicado a E/S em geral, não apenas em sockets. A *system call select** apresenta o seguinte formato:

```
nready = select(ndesc, indesc, outdesc, excdesc, timeout)
```

Em geral, uma chamada ao *select* causa um bloqueio esperando que um descritor, considerando um conjunto de descritores, fique pronto. O argumento *ndesc* especifica quantos descritores devem ser examinados (os descritores verificados são sempre de *0* até *ndesc*-1). O argumento *indesc* é um ponteiro para uma máscara de bits que especifica os descritores de arquivos para verificar se houve uma entrada (leitura); o argumento *outdesc* é um ponteiro para uma máscara de bits que especifica os descritores de arquivo para verificar uma saída (escrita), e o argumento *excdesc* é um ponteiro para uma máscara de bits que especifica os descritores de arquivos para verificar se ocorreu uma condição excepcional. Por fim, se o argumento *timeout* for diferente de zero, ele será o endereço de um inteiro que especifica quanto tempo se deve esperar uma conexão antes de retornar para quem invocou a chamada. Um *timeout* com valor igual a zero força a chamada a bloquear até que o descritor esteja pronto. Uma vez que o argumento *timeout* contenha o endereço de um inteiro vinculado ao timeout, e não um inteiro propriamente, o processo pode requisitar um retardo igual a zero passando o endereço de um inteiro que contém zero (um processo pode circular para verificar se a E/S está pronta).

Uma chamada para *select* retorna o número dos descritores do conjunto especificado que estão prontos para E/S. Também altera a máscara de bits especificada por *indesc*, *outdesc* e *excdesc* para informar ao aplicativo quais dos descritores selecionados estão prontos. Assim, antes de chamar *select*, é preciso ligar os bits correspondentes ao descritor a ser verificado. No curso da chamada, todos os bits que estiverem em *1* correspondem a um descritor de arquivo no estado pronto.

Para se comunicar sobre mais de um socket, um processo primeiro cria todos os sockets de que necessita e, então, utiliza *select* para determinar qual deles se tornou pronto para E/S primeiro. Uma vez encontrado o socket que está pronto, o processo utiliza as funções de entrada ou saída definidas para se comunicar.

21.9 Obtendo e definindo o nome de host

Embora o IP utilize um endereço de destino quando entrega datagramas, usuários e programas aplicativos usam um *nome* para se referir a um computador. Para os computadores na Internet, seus nomes são derivados do Sistema de nome de domínio (Domain Name System), descrito no Capítulo 23. A função *gethostname* permite que um aplicativo obtenha o nome do computador local. A função relacionada, *sethostname*, permite a um gerente definir o nome do host para uma determinada string. *Gethostname* tem o seguinte formato:

gethostname(name, length)

O argumento *name* fornece o endereço de uma matriz (array) de bytes onde o nome está armazenado, e o argumento *length* é um inteiro que especifica o tamanho máximo do nome (por exemplo, o tamanho do array name). A função retorna o tamanho do nome que é recuperado. Para registrar o tamanho do host, um privilegiado aplicativo chama

sethostname(name, length),

no qual o argumento *name* fornece o endereço de um array onde um nome é armazenado, e o argumento *length* é um inteiro que dá o tamanho do nome.

21.10 Funções de biblioteca relacionadas a sockets

Além das funções descritas anteriormente, o socket API oferece um grande conjunto de funções que executam tarefas úteis relacionadas com a rede. Como elas não interagem diretamente com o software de protocolo, muitas das funções adicionais de socket são implementadas como rotinas de biblioteca. A Figura 21.3 ilustra a diferença entra chamadas de sistemas (system calls) e rotinas de biblioteca.

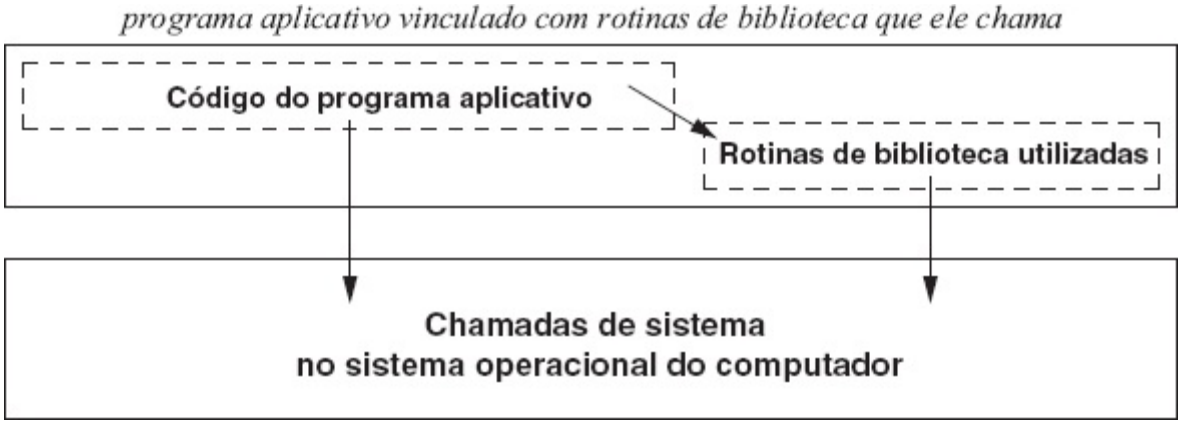


Figura 21.3 Ilustração da diferença entre as rotinas de biblioteca, as quais são vinculadas à aplicação, e chamadas de sistemas, que são parte do sistema operacional.

Conforme a figura mostra, uma chamada de sistema passa o controle diretamente para o sistema operacional do computador. Chamar uma função da biblioteca passa o controle para uma cópia da função que foi incorporada no aplicativo. Um aplicativo pode fazer uma chamada de sistema diretamente ou ainda chamar funções de biblioteca que (geralmente) fazem uma chamada de sistema.

Muitas das rotinas de bibliotecas socket fornecem serviços de banco de dados que permitem que um processo determine o nome de máquinas e serviços de redes, portas de protocolos e outras informações relacionadas. Por exemplo, as rotinas de biblioteca fornecem acesso ao banco de dados de serviços de redes. Pensamos em entradas no banco de dados de serviços como tuplas ternárias:

(service_name, protocol, protocol_port_number)

Onde *service_name* é uma string que dá o nome do serviço, *protocol* é um protocolo de transporte usado para acessar o serviço, e *protocol_port_number* é o número da porta para usar. Por exemplo, o serviço de eco UDP descrito no Capítulo 20 tem a seguinte entrada:

(“echo”, “udp”, 7)

As próximas seções examinam grupos de rotinas de biblioteca, explicando seus propósitos e fornecendo informação sobre como elas podem ser utilizadas. Veremos que conjuntos de funções de biblioteca socket muitas vezes seguem o mesmo padrão. Cada conjunto de rotinas

permite à aplicação: estabelecer comunicação com o banco de dados (o qual pode ser um arquivo no computador local ou no servidor remoto), obter entradas uma por vez e terminar o uso. As rotinas utilizadas para as três operações são chamadas *setXent*, *getXent* e *endXent*, onde *X* é o nome do banco de dados. Por exemplo, as rotinas para banco de dados de host são nomeadas *sethostent*, *gethostent* e *endhostent*. As seções que descrevem as rotinas resumem as chamadas sem repetir os detalhes sobre seu uso.

21.11 Network Byte Order e rotinas de conversão

Lembre-se de que as máquinas diferem entre si quanto à forma como armazenam quantidades inteiras e que os protocolos TCP/IP definem um padrão ordem de bytes de rede (Network Byte Order) que é independente de qualquer computador. A API socket provê quatro macros para conversão de ordem de byte em máquina local e a ordem de byte padrão da rede. Para tornar os programas portáveis, eles precisam ser escritos de forma a chamar as rotinas de conversão toda vez que copiarem um valor inteiro de uma máquina local para um pacote de rede, ou quando copiarem um valor de um pacote para a máquina local.

Todas as quatro rotinas de conversão são funções que pegam um valor como argumento e retornam um novo valor com os bytes rearrumados. Por exemplo, para converter um inteiro curto (2 bytes) da ordem de byte de rede para a ordem de byte do host local, o programador efetua a chamada *ntohs* (network to host short). O formato é:

```
localshort = ntohs(netshort)
```

O argumento *netshort* é um inteiro de 2 bytes (l6 bits) na ordem de byte padrão da rede, e o resultado, *localshort*, é o mesmo inteiro na ordem de byte do host.

As chamadas da linguagem de programação C são um inteiro de 4 bytes (32 bits). A função *ntohl* (network to host long) converte 4-bytes da ordem de byte padrão da rede para a ordem de byte do host local. Os programas invocam a *ntohl* como uma função, fornecendo um inteiro longo em ordem de byte de rede como um argumento:

```
locallong = ntohl(netlong)
```

Duas funções análogas permitem ao programador converter de ordem de

byte de host local para ordem de byte de rede. A função *htons* converte um inteiro curto 2 bytes na ordem de byte do host local para um inteiro de 2 bytes na ordem de byte padrão de rede. Programas invocam *htons* como uma função:

```
netshort = htons(localshort)
```

A última rotina de conversão, *htonl*, converte 4 bytes inteiros para a ordem de byte padrão da rede. Como as outras, *htonl* é uma função:

```
netlong = htonl(locallong)
```

21.12 Rotinas de manipulação do endereço IP

Uma vez que muitos programas efetuam a tradução entre um endereço IP de 32 bits e o correspondente em notação decimal com ponto ou entre os endereços IPv6 de 128 bits e notação hexadecimal com dois pontos, a biblioteca socket inclui rotinas utilitárias que efetuam a tradução. Por exemplo, a função *inet_aton* traduz do formato decimal com ponto para um endereço IPv4 de 32 bits na ordem byte da rede. Uma chamada tem o formato

```
error_code = inet_aton(string, address),
```

no qual o argumento *string* é uma string ASCII que contém o endereço IPv4 expresso no formato decimal com ponto e *address* é um ponteiro para um inteiro de 32 bits em que um valor binário é colocado. O formato decimal com ponto pode ter de um a quatro segmentos de dígitos separados por períodos (pontos; dots). Se todos os quatro aparecem, cada um corresponde a um único octeto resultante do número inteiro de 32 bits. Se menos do que quatro aparecem, o último segmento se expande para preencher os octetos restantes do endereço.

A função *inet_ntoa* efetua o inverso do *inet_aton* mapeando um endereço IPv4 de 32 bits em uma string ASCII no formato decimal com ponto. O formato é o seguinte:

```
str = inet_ntoa(internetaddr)
```

no qual o argumento *internetaddr* é um endereço IPv4 de 32 bits na

ordem de byte de rede, e *str* é o endereço da string ASCII resultante.

21.13 Acessando o Domain Name System

Um conjunto de cinco procedimentos de bibliotecas constitui a interface do *Domain Name System* (DNS).* Os programas aplicativos que chamam as funções de biblioteca tornam-se clientes do Domain Name System. Ou seja, quando um aplicativo chama a função de biblioteca para obter informações, esta constrói uma consulta (query), a envia para um servidor de nome de domínio e aguarda uma resposta. Assim que a resposta chega, a função de biblioteca retorna as informações para o aplicativo que fez a chamada. Como existem muitas opções, a função de biblioteca dispõe de apenas alguns argumentos básicos. Eles contam com uma estrutura global, *res*, para manter argumentos adicionais. Por exemplo, um campo em *res* habilita a depuração de mensagens, enquanto outros campos controlam se o aplicativo especifica utilizar o UDP ou TCP para consultas. Muitos dos campos na estrutura *res* começam com valores defaults, razoáveis, então as funções do socket de biblioteca muitas vezes podem ser utilizadas sem alterar *res*.

Um programa chama *res_init* antes de usar outras funções. A chamada não tem argumentos:

```
res_init()
```

Res_init armazena o nome de um servidor de nome de domínio na estrutura *res* global, tornando o sistema pronto para contatar o servidor.

A Função *res_mkquery* forma uma consulta de nome de domínio e coloca a consulta em um buffer na memória. A forma da chamada é:

```
res_mkquery(op, dname, class, type, data, datalen, newrr, buffer, buflen)
```

Os primeiros sete argumentos correspondem diretamente aos campos de consulta do nome de domínio. O argumento *op* especifica as operações requisitadas; *dname* fornece o endereço de um array de caracteres que contém o nome do domínio; *class* é um inteiro que fornece a classe da consulta; *type* é um inteiro que fornece o tipo da consulta; *data* fornece o

endereço de um array de dados a serem incluídos na consulta; *datalen* é um inteiro que fornece o tamanho do dado. Além das funções de biblioteca, a API socket oferece aos programas aplicativos definições de constantes simbólicas de valores importantes. Assim, os programadores podem usar o sistema de domínio de nomes sem precisar entender os detalhes do protocolo. Os últimos dois, *buffer* e *buflen*, especificam, respectivamente, o endereço da área em que a consulta deve ser colocada e o tamanho (um inteiro) da área de buffer. Na implementação corrente, o argumento *newrr* não é utilizado.

Uma vez que um programa tenha formado uma consulta, ele chama *res_send* para enviá-la a um servidor de nomes e obter a resposta. O formato é:

```
res_send(buffer, buflen, answer, anslen)
```

O argumento *buffer* é um ponteiro para a memória que contém a mensagem a ser enviada (presumivelmente, a aplicação chamou o procedimento *res_mkquery* para formar a mensagem). O argumento *buflen* é um inteiro que especifica o tamanho, o argumento *answer* fornece o endereço na memória em que a resposta deve ser escrita, o argumento inteiro *anslen* especifica o tamanho da área de resposta.

Além das funções que criam e enviam consultas, a biblioteca socket contém duas funções que traduzem nomes de domínio entre a forma convencional (ASCII) e o formato comprimido utilizado nas consultas. A função *dn_expand* expande um nome de domínio comprimido para sua versão completa em ASCII. O formato é o seguinte:

```
dn_expand(msg, eom, compressed, full, fullen)
```

O argumento *msg* fornece o endereço de uma mensagem de nome de domínio que contém o nome a ser expandido, com *eom* especificando o limite fim da mensagem que a expansão não pode ultrapassar; o argumento *full* é um ponteiro para um array em que o nome expandido deve ser escrito; o argumento *fullen* é um inteiro que especifica o tamanho do array.

Gerar um nome comprimido é mais complexo do que expandi-lo, posto que a compressão envolve eliminar sufixos comuns. Quando comprimimos

os nomes, o cliente precisa manter o registro de sufixos que apareceram previamente. A função *dn_comp* comprime um nome de domínio completo comparando-o com uma lista de sufixos utilizados previamente e eliminando os possíveis sufixos de maior tamanho. A chamada tem o seguinte formato:

```
dn_comp(full, compressed, cmprlen, prevptrs, lastptr)
```

O argumento *full* fornece o endereço do nome de domínio completo. O argumento *compressed* aponta para um array de bytes que conterá o nome comprimido com o argumento *cmprlen* especificando o tamanho do array. O argumento *prevptrs* é o endereço de um array de ponteiros para sufixos previamente comprimidos da mensagem corrente, com *lastptr* apontando para o final do array. Normalmente, *dn_comp* comprime o nome e atualiza *prevptrs* caso um novo sufixo tenha sido utilizado.

A função *dn_comp* pode ser também utilizada para traduzir um nome de domínio em ASCII para um formato interno, sem compressão (ou seja, sem remover os sufixos). Para tal, o processo invoca *dn_comp* com o argumento *prevptrs* inicializado em NULL (por exemplo, zero).

21.14 Obtendo informações sobre hosts

Existem funções de biblioteca que permitem que um processo recupere informações sobre um dado host e seu nome de domínio, ou seu endereço IP. As funções de biblioteca fazem com que o aplicativo seja um cliente do sistema de nome de domínio: elas enviam uma requisição para servidor de nome de domínio e esperam uma resposta. Por exemplo, a função *gethostbyname* recebe um nome de domínio e retorna um ponteiro para uma estrutura de informação sobre um host especificado. A chamada tem o seguinte formato:

```
ptr = gethostbyname(namestr)
```

O argumento *namestr* é um ponteiro para uma string de caracteres que contém um nome de domínio desse host. O valor retornado, *ptr*, aponta para uma estrutura que contém as seguintes informações: o nome oficial do

host, a lista de apelidos (aliases) que foram registrados para esse host, o tipo de endereço do host (por exemplo, IPv4, IPv6, ou algum outro tipo), o tamanho de um endereço, e uma lista de um ou mais endereços para o host. Mais detalhes podem ser encontrados no manual do programador UNIX.

A função *gethostbyaddr* produz a mesma informação que a *gethostbyname*. A diferença entre as duas funções é que a *gethostbyaddr* aceita um endereço de host como argumento:

```
ptr = gethostbyaddr(addr, len, type)
```

O argumento *addr* é um ponteiro para uma sequência de bytes que contém um endereço de host. O argumento *len* é um inteiro que fornece o tamanho do endereço, e o argumento *type* é um inteiro que especifica o tipo do endereço (por exemplo, que é um endereço IPv6).

Como mencionado anteriormente, as funções *sethostent*, *gethostent* e *endhostent* fornecem acesso sequencial ao banco de dados – um aplicativo pode abrir um banco de dados, extrair entradas sequencialmente e, em seguida, fechar o banco de dados.

21.15 Obtendo informações sobre redes

A biblioteca socket também inclui funções que permitem a um aplicativo acessar o banco de dados da rede. A função *getnetbyname* obtém e formata o conteúdo de uma entrada a partir do banco de dados dado no nome de domínio de uma rede. Uma chamada tem o seguinte formato:

```
ptr = getnetbyname(name)
```

no qual o argumento *name* é um ponteiro para uma string que contém o nome da rede para a qual a informação é desejada. O valor retornado é um ponteiro para uma estrutura que contém campos para o nome oficial da rede, uma lista de apelidos (aliases) registrados, um tipo de endereço inteiro (por exemplo, IPv4, IPv6, ou algum outro tipo) e o endereço de prefixo usado com a rede (ou seja, a parte da rede de um endereço IP com a parte do host definido para zero).

21.16 Obtendo informações sobre protocolos

Cinco rotinas de biblioteca oferecem acesso ao banco de dados de protocolos disponíveis em uma máquina. Cada protocolo tem um nome oficial, apelidos (aliases) registrados e um número de protocolo oficial. A função *getprotobyname* permite que quem chama obtenha informações sobre um protocolo dando seu nome:

```
ptr = getprotobyname(name)
```

O argumento *name* é um ponteiro para uma string ASCII que contém o nome do protocolo para o qual a informação é desejada. A função retorna um ponteiro para uma estrutura que possui campos para o nome de protocolo oficial, uma lista de aliases e um valor inteiro exclusivo atribuído ao protocolo.

A função *getprotobynumber* permite que um processo procure informações de protocolo usando o número de protocolo como uma chave:

```
ptr = getprotobynumber(number)
```

Finalmente, as funções *getprotoent*, *setprotoent* e *endprotoent* oferecem acesso sequencial ao banco de dados de protocolo.

21.17 Obtendo informações sobre serviços de rede

Lembre-se (Capítulos 10 e 11) de que alguns números de protocolo UDP e TCP são reservados para serviços bem conhecidos. Por exemplo, a porta TCP 37 é reservada para o protocolo de tempo descrito no capítulo anterior. A entrada no banco de dados de serviços especifica o nome do serviço, tempo, um protocolo (por exemplo, TCP) e o número de porta do protocolo 37. Existem cinco rotinas de biblioteca que obtêm informações sobre os serviços e as portas de protocolo que eles utilizam.

A função *getservbyname* é a mais importante porque mapeia um serviço chamado para um número de porta:

```
ptr = getservbyname(name, proto)
```

O argumento *name* especifica o endereço de uma string que contém o nome do serviço desejado, e o argumento *proto* é uma string que dá o nome do protocolo com o qual o serviço deve ser usado. Normalmente, os protocolos são limitados a TCP e UDP. O valor retornado é um ponteiro para uma estrutura que contém campos para o nome do serviço, uma lista

de apelidos (aliases), uma identificação do protocolo com o qual o serviço é usado e um número de porta de protocolo inteiro, atribuído para esse serviço.

A função *getservbyport* permite que quem chama obtenha uma entrada a partir do banco de dados de serviços dado o número de porta atribuído. Uma chamada tem o formato:

```
ptr = getservbyport(port, proto)
```

O argumento *port* é o número de porta de protocolo inteiro atribuído ao serviço, e o argumento *proto* especifica o protocolo para o qual o serviço é desejado. Assim como outros bancos de dados, um aplicativo pode acessar o banco de dados de serviços sequencialmente usando *setservent*, *getservent* e *endservent*.

21.18 Um exemplo de cliente

O programa exemplo em C a seguir ilustra como um programa utiliza a API socket para acessar protocolos TCP/IP. O cliente faz uma conexão TCP com um servidor, envia as linhas de texto que um usuário entra e exibe a resposta do servidor para cada um.

```
/*****************************************************************************/
/*                                                                           */
/* Programa:  Cliente testa o exemplo servidor de eco.                       */
/*                                                                           */
/* Método:      Formar uma conexão TCP com o servidor de eco e                 */
               repetidamente ler uma linha de texto, enviar o texto        */
               para o servidor e receber o mesmo texto de volta a          */
               partir do servidor.                                         */
/*                                                                           */
/* Use:        cliente [-p port]host                                          */
/*                                                                           */
/*          Onde port (porta) é o número de porta TCP ou nome, e host        */
             é o nome ou endereço IP do host do servidor.                  */
/*                                                                           */
/* Autor:    Barry Shein, bxs@TheWorld.com, 3/1/2013                         */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>
#include <stdarg.h>
#include <sys/types.h>
#include <errno.h>
#include <fcntl.h>
#include <time.h>
#ifdef USE_READLINE
#include <readline/readline.h>
#include <readline/history.h>
```

```
#endif /* USE_READLINE */

#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
static char  *prog;        /* ptr para nome do programa (para mensagens) */
#define DEFAULT_PORT"9000" /* deve combinar com porta padrão do servidor */

/* Define códigos de saida do processo */

#define EX_OK 0            /* finalização Normal  */
#define EX_ARGFAIL 1       /* argumentos Incorretos */
#define EX_SYSERR  2       /* Erro no sistema de chamada  */

/* Log - exibe uma mensagem de erro ou informacional para o usuário */

static void Log(char *fmt,...) {
    va_list ap;

    va_start(ap,fmt);
    (void)vfprintf(stderr,fmt,ap);
    va_end(ap);
}

/* Fatal - exibe mensagem de erro fatal para o usuário e sai */

static void Fatal(int exval,char *fmt,...) {
    va_list ap;

    va_start(ap,fmt);

    void)vfprintf(stderr,fmt,ap);
    va_end(ap);
    exit(exval);
}

/* getLine - pega uma linha de entrada do keyboard */

static char *getLine(char *prompt) {
#ifdef USE_READLINE
    return(readline(prompt));
#else /* !USE_READLINE */

    (void)fputs(prompt,stdout); /* exibe o prompt */
    fflush(stdout);

    /* Lê uma linha do keyboard retorna NULL */
    if(fgets(buf,sizeof(buf),stdin) == NULL)
         return(NULL);
    else {
         char *p;

         /* emulate readline() and strip NEWLINE */
         if((p = strrchr(buf,'\n')) != NULL)
              *p = '\0';
         return(strdup(buf)); /* readline retorna allocated buffer */
    }
#endif /* !USE_READLINE */
}

/* initClient - inicializa e cria uma conexão com o servidor */
```

```
static int initClient(char *host,char *port) {
    struct addrinfo hints;
    struct addrinfo *result, *rp;
    int s;

    memset(&hints,0,sizeof(hints));
    hints.ai_family = AF_UNSPEC;      /* use IPv4 or IPv6     */
    hints.ai_socktype = SOCK_STREAM;  /* stream socket (TCP) */

    /* Get address of server host */
    if((s = getaddrinfo(host,port,&hints,&result)) != 0)
         Fatal(EX_SYSERR,"%s: getaddrinfo: %s\n",prog,gai_strerror(s));

    /* tentar cada endereço correspondente a nome */
    for(rp=result; rp != NULL; rp = rp->ai_next) {
         int sock, ret;        /* descriptor de socket e retorna value  */
    char hostnum[NI_MAXHOST];  /* nome do host                          */

    /* Pega endereço numérico do host para mensagem */
    if((ret = getnameinfo(rp->ai_addr,rp->ai_addrlen,hostnum,
                          sizeof(hostnum), NULL,0,NI_NUMERICHOST)) != 0) {
         Log("%s: getnameinfo: %s\n",prog,gai_strerror(ret));
    } else {
         (void)printf("Trying %s...",hostnum);
         fflush(stdout);
    }

    /* Pega um novo socket */

    if((sock =
         socket(rp->ai_family,rp->ai_socktype,rp->ai_protocol)) < 0) {
         if((rp->ai_family == AF_INET6) && (errno == EAFNOSUPPORT))
         Log("\nsocket: no IPv6 support on this host\n");
    else
         Log("\nsocket: %s\n",strerror(errno));
    continue;
    }

    /* tenta conectar o novo socket ao servidor */

    if(connect(sock,rp->ai_addr,rp->ai_addrlen) < 0)
         { Log("connect:%s\n",strerror(errno));
         (void)shutdown(sock,SHUT_RDWR);
         continue;
    } else { /* success */
         (void)printf("connected to %s\n",host);
         return(sock);
         break;
      }
    }
    Fatal(EX_ARGFAIL,"%s: não é possivel conectar com o host %s\n",prog,host);
    return(-1); /* nunca alcançado, mas isso suprime alerta */
    }

    /* runClient - ler a partir do keyboard, enviar para o servidor, resposta eco */

    static void runClient(int sock)
         { FILE *sfp;
         char *input;

    /* criar uma buffered stream para o socket */
```

```
    if((sfp = fdopen(sock,"r+")) == NULL) { (void)shutdown(sock,SHUT_RDWR);
         Fatal(EX_SYSERR,"%s: não foi possivel criar buffered sock.\n",prog);
    }
    setlinebuf(sfp);

    (void)printf("\nWelcome to %s: period newline exits\n\n",prog);

    /* read keyboard... */
    while(((input=getLine("> ")) != NULL) && (strcmp(input,".") != 0)) {
         char buf[BUFSIZ];

         (void)fprintf(sfp,"%s\n",input);/* escrever para o socket */
    free(input);
    if(fgets(buf,sizeof(buf),sfp) == NULL) { /* ter resposta */
         Log("%s: lost connection\n",prog);
         break;
    } else
         (void)printf("response: %s",buf); /* servidor de echo resp.*/
    }
}

/* doneClient - termina: fecha socket */

static void doneClient(int sock) {
    if(sock >= 0)
         if(shutdown(sock,SHUT_RDWR) != 0)
              Log("%s: shutdown error: %s\n",strerror(errno));
         Log("client connection closed\n");
}

/* Usage - comando útil linha de mensagem */

static void Usage(void) {
    (void)printf("Usage: %s [-p port] host\n",prog);
    exit(EX_OK);
}

/* main - análise da linha de comando e inicia cliente */

int main(int argc,char **argv) {
    int c;
    char *host = NULL;

    char *port = DEFAULT_PORT;
    int sock;
    prog = strrchr(*argv,'/') ? strrchr(*argv,'/')+1 : *argv;

    while((c = getopt(argc,argv,"hp:")) != EOF)
    switch(c) {
         case 'p':
         port = optarg;
         break;
    case 'h':
         default:
         Usage();
       }
    if(optind < argc) {
    host = argv[optind++];
    if (optind != argc) {
         Log("%s: quantidade muito grande de args na linha de comando\n",prog);
         Usage();
```

```
        }
    } else {
        Log("%s: missing host arg\n",prog);
        Usage();
    }
    sock = initClient(host,port); /* chamada vai se encerrar se houver erro ou falha */
    runClient(sock);
    doneClient(sock);
    exit(EX_OK);
}
```

21.19 Um exemplo de servidor

O código de exemplo de servidor é apenas um pouco mais complexo do que o código do cliente. A operação em geral é simples: o servidor é iterativo; começa especificando a porta a ser utilizada e aguarda conexões. Ele aceita uma conexão TCP de entrada, executa um serviço e espera a próxima conexão. O serviço utilizado é o de eco trivial: o servidor lê linhas de entrada de texto e envia, inalterada, cada linha de volta para o cliente. Este deve encerrar a conexão.

O servidor irá permitir a um cliente que use o IPv4 ou IPv6 (assumindo que o IPv6 está disponível). Mesmo em sistemas onde a pilha de protocolo não está configurada para IPv6, o código supõe que a *inclusão* de arquivos está disponível para utilização dos programas.

```
/*************************************************************************/
/*                                                                       */
/* Program:   Server that offers a text echo service via TCP on          */
/*            IPv4 or IPv6                                               */
/*                                                                       */
/* Method:    Repeatedly accept a TCP connection, echo lines of text     */
/*            until the client closes the connection, and go on to       */
/*            wait for the next connection.                              */
/*                                                                       */
/* Use:       server [-p port]                                           */
/*                                                                       */
/*            where port is a TCP port number or name                    */
/*                                                                       */
/* Author:    Barry Shein, bxs@TheWorld.com, 3/1/2013                  */
/*                                                                       */
/*************************************************************************/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>
#include <stdarg.h>
#include <sys/types.h>
#include <errno.h>
#include <signal.h>
#include <setjmp.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>

static char *prog;             /* ptr para nome do programa (para mensagens) */

/* Isto é arbitrário mas deve ser não privilegiado (>1024) */
#define DEFAULT_PORT    "9000"   /* deve coincidir com a porta padrão do cliente */

/* Define process exit codes */
```

```
#define EX_OK        0    /* Terminação Normal                                */
#define EX_ARGFAIL   1    /* Argumentos Incorretos                            */
#define EX_SYSERR    2    /* Erro no sistema de chamada chamadaystem call     */
#define EX_NOMEM     3    /* Não pode alocar memória                          */

/* Estrutura do Servidor usada para passar informação internamente */

typedef struct {

    int      sock;         /* descriptor de socket   */
    char     *port_name;   /* ptr para nomear a porta em uso */
    int      port_number;  /* valor inteiro para porta*/
    FILE     *ferr;        /* stdio lidar com mensagens de erro */
} _Server, *Server;

/* Log - exibe uma mensagem de erro ou informacional para o usuário */

static void Log(Server srv, char *fmt,...) {
    va_list ap;

    va_start(ap,fmt);
    void)vfprintf(srv->ferr,fmt,ap);
    va_end(ap);
}

/* Fatal - exibe uma mensagem de erro fatal para o usuário e então sai */

static void Fatal(Server srv,int exval,char *fmt,...) {
    va_list ap;
    va_start(ap,fmt);
    (void)vfprintf(srv->ferr,fmt,ap);
    va_end(ap);
    exit(exval);
}

/* newServer - Cria um novo servidor objeto */

static Server newServer(void)
    { Server srv;

    /* Aloca memória para um novo servidor, sair se der erro */

    if((srv = (Server)calloc(1,sizeof(*srv))) == NULL) {
        (void)fprintf(stderr,"%s",strerror(errno));
        exit(EX_NOMEM);
    } else {
        srv->ferr = stderr; /* initialize log output */
        return(srv);
}
}

/* freeServer - memória livre associada com instância de um servidor struct */

static void freeServer(Server srv) {


    if(srv->port_name != NULL)
        free(srv->port_name);
    free(srv);
}
```

```
/* initServer - Inicializar instância de um servidor struct */

static Server initServer(char *port) {
    Server srv;
    char *protocol = "tcp";
    struct protoent *pp;
    struct servent *sport;
    char *ep;
    extern const struct in6_addr in6addr_any;
    struct sockaddr_storage sa;
    int sopt = 0;
    extern int errno;

    srv = newServer();                  /* sair se falhar  */
    srv->port_name = strdup(port);      /* salvar nome porta onde passaram  */

    /* pesquisa número de protocolo para "tcp" */

    if((pp = getprotobyname(protocol)) == NULL)
       Fatal(srv,EX_ARGFAIL,"initServer: %s\n",strerror(errno));

    /* Primeiro verificar se o número da porta é uma string de digito, tal como "9000", */
    /* e então verifique se é um nome tal como "echo" (see /etc/services) */

    if(((srv->port_number=strtol(srv->port_name,&ep,0))>0) && (*ep=='\0'))
       srv->port_number = htons(srv->port_number);
    else if((sport = getservbyname(srv->port_name,protocol)) == NULL)
       Fatal(srv,EX_ARGFAIL,"initServer: bad port '%s'\n",srv->port_name);

    else
       srv->port_number = sport->s_port; /* Success */

    /* Pega um novo socket IPv4 ou IPv6 e o prepara para vinculo() */

    (void)memset(&sa,0,sizeof(sa));
    if((srv->sock = socket(AF_INET6,SOCK_STREAM,pp->p_proto)) < 0) {
       if(errno == EAFNOSUPPORT) { /* Não há IPv6 neste sistema; use IPv4 */
          se((srv->sock = socket(AF_INET,SOCK_STREAM,pp->p_proto)) < 0)
             Fatal(srv,EX_SYSERR,"initServer: socket: %s\n",
                                                 strerror(errno));
          else {

             struct sockaddr_in *sa4 = (struct sockaddr_in *)&sa;
             sa4->sin_family = AF_INET;
             sa4->sin_port = srv->port_number;
             sa4->sin_addr.s_addr = INADDR_ANY;
          }
       }
    } else { /* IPv6 supported */
       struct sockaddr_in6 *sa6 = (struct sockaddr_in6 *)&sa;
       /* Defina a opção de socket IPV6_V6ONLY como zero (off) então */
       /* escutará conexões de entrada para ambos IPv6 e IPv4. */
       if(setsockopt(srv->sock,IPPROTO_IPV6,IPV6_V6ONLY,&sopt,
                                                 sizeof(sopt)) < 0)
          Fatal(srv,EX_SYSERR,"initServer: setsockopt: %s\n",
                                                  strerror(errno));
       sa6->sin6_family = AF_INET6;
       sa6->sin6_port = srv->port_number;
       sa6->sin6_addr = in6addr_any; /* Listen to any iface & addr */
    }
```

```
    /* Vincula o novo socket ao serviço */

    if(bind(srv->sock,(const struct sockaddr *)&sa,sizeof(sa)) < 0)
        Fatal(srv,EX_SYSERR,"initServer: bind: %s\n",strerror(errno));
    /* Defina o número máximo de conexões de entrada em espera */
    if(listen(srv->sock,SOMAXCONN) < 0)
        Fatal(srv,EX_SYSERR,"initServer: listen: %s\n",strerror(errno));
    return(srv);
}

/* runServer - Execute o servidor & aceite conexões de entrada iterativamente */

static void runServer(Server srv) {
    while(1) { /* Interaja sempre (até o usuário abortar o processo)*/
        int s;

        /* sockaddr_storage é grande o suficiente para manter informação tanto */
        /*   do socket IPv6 ou IPv4, como definido pelo sistema. */

      struct sockaddr_storage addr;
      socklen_t addrlen = sizeof(addr);
      struct sockaddr *sap = (struct sockaddr *)&addr;

      /* aceitar vai bloquear esperando por uma nova conexão de entrada */
      memset(&addr,0,sizeof(addr));
      if((s = accept(srv->sock,sap,&addrlen)) >= 0) {


      char host[NI_MAXHOST];
      char service[NI_MAXSERV];
      FILE *sfp;

      /* Pegue informação sobre o novo cliente */

/*NOTUSED  if(getpeername(s,sap,&addrlen) != 0)
              {Log(srv,"getpeername: %s\n",strerror(errno));
              (void)shutdown(s,SHUT_RDWR);
              continue;
      } else END_NOTUSED*/ if(getnameinfo(sap,addrlen, host,
                 sizeof(host), service,sizeof(service),0) != 0) {
              Log(srv,"getnameinfo: %s\n",strerror(errno));
              (void)shutdown(s,SHUT_RDWR);
              continue;
      }
      Log(srv,"accept: host=%s port=%s\n",host,service);

      /* crie um buffered stream para o novo socket */

      if((sfp = fdopen(s,"r+")) == NULL) {
           Log(srv,"fdopen: erro criando buffered stream?\n");
           (void)shutdown(s,SHUT_RDWR);
           continue;
      } else { /* Uma conexão válida foi aceita */
          char buf[BUFSIZ];

          /* loop, lendo entrada e respondendo com char count */
          setlinebuf(sfp);
          while(fgets(buf,sizeof(buf),sfp) != NULL) {
              Log(srv,"client: %s",buf);
              (void)fprintf(sfp,"got %zd chars\n",strlen(buf));
      }
```

```
      Log(srv,"cliente fechou a conexão\n");
      if(shutdown(s,SHUT_RDWR) != 0)
            Log(srv,"%s: shutdown error: %s\n",strerror(errno));
            (void)fclose(sfp); /* libere qualquer memória associada */
                               /* with stdio file pointer sfp */
      }
    }
  }
}

/* doneServer - usuário abortou o processo, então feche o servidor socket e Log */

static void doneServer(Server srv) {
    if(shutdown(srv->sock,SHUT_RDWR) != 0)
       Log(srv,"%s: shutdown error: %s\n",strerror(errno));
    freeServer(srv);
    Log(srv,"\n%s: shut down\n\n",prog);
}

/* Cuide do desligamento do servidor quando vários sinais ocorrem */

static jmp_buf sigenv;
static void onSignal(int signo) {
    longjmp(sigenv,signo); /* envie de volta sinal num se alguém responder*/
}

/* Usage - Imprima uma mensagem informando ao usuário sobre args, e então saia */

static void Usage(void) {
    (void)printf("Usage: %s [-p tcp_port]\n",prog);
    exit(EX_OK);
}

/* main - main program: analise os argumentos e então inicie o servidor */

int main(int argc,char **argv) {
    Server srv;
    char *port = DEFAULT_PORT; /* porta protocolo padrão a usar */
    int c;

    prog = strrchr(*argv,'/') ? strrchr(*argv,'/')+1 : *argv;

    /* Parse argument */

    while((c = getopt(argc,argv,"hp:")) != EOF)
       switch(c) {
       case 'p': /* nome ou número da porta da linha de comando line */
          port = optarg;
          break;
       case 'h': /* help, falls through... */
          default: /* comando arg não reconhecido */
          Usage();
       }

    srv = initServer(port); /* esta chamada termina ao erro */

    if(setjmp(sigenv) > 0) {

          doneServer(srv); /* para cá ao sinal */
          exit(EX_OK);
       } else {
```

```
        signal(SIGHUP,onSignal);
        signal(SIGINT,onSignal);
        signal(SIGTERM,onSignal);
    }

    Log(srv,"\n%s: Inicializado, aguardando por conexões de entrada\n\n",prog);
    runServer(srv);
    return(EX_OK); /* suprime aviso de compilamento */
}
```

21.20 Resumo

Embora os padrões TCP/IP não definam a interface exata entre um programa aplicativo e protocolos TCP/IP, o socket API tornou-se um padrão de fato utilizado por fabricantes como Microsoft e Apple, bem como em Linux. Os sockets adotaram o paradigma UNIX abrir-fechar-ler-escrever (open-close-read-write) e adicionaram muitas novas funções. Um aplicativo de servidor deve criar um socket, ligar endereços para ele, aceitar conexões de entrada ou mensagens e enviar respostas. Um cliente deve criar um socket, conectá-lo a um ponto de extremidade remota e, em seguida, comunicar. Quando um aplicativo termina usando um socket, o aplicativo precisa fechá-lo. Além do sistema de chamadas de socket, o socket API inclui muitas rotinas de biblioteca que ajudam os programadores a criarem e manipularem endereços IP, converter inteiros entre o formato da máquina local e a ordem de bytes padrão da rede e procurar informações tais como endereços de host.

Examinamos exemplo de código para um cliente e servidor o qual ilustra o uso da API socket para um serviço básico de eco textual. Além de muitos detalhes relacionados com o uso de sockets, o código de exemplo é complicado porque está escrito para usar ou IPv4 ou IPv6, sendo dada preferência ao IPv6.

EXERCÍCIOS

21.1 Faça o download do exemplo cliente e servidor de *comerbooks.com* e execute-o em seu sistema local.

21.2 Monte um servidor simples, que aceite várias conexões concorrentes TCP. Para testá-lo, faça com que o processo que trata de uma conexão imprima uma mensagem curta, espere um tempo qualquer, imprima outra mensagem e saia.

21.3 Quando a chamada *listen* é importante?

21.4 Que funções o seu sistema local oferece para acessar o Domain Name

System?

21.5 Elabore um servidor que use um único processo do Linux (ou seja, um único thread de execução), mas que trate de várias conexões TCP simultâneas. (Dica: pense em *select*.)

21.6 Leia sobre as alternativas à interface socket, como *Transport Library Interface* (TLI), e compare-as com sockets. Quais são as principais diferenças conceituais?

21.7 Cada sistema operacional limita o número de sockets que determinado programa pode usar a qualquer momento. Quantos sockets um programa pode criar no seu sistema local?

21.8 Podem o mecanismo descritor de socket/arquivo e as operações *read* e *write* associadas serem considerados uma forma de projeto orientado a objeto? Sim ou não? Por quê?

21.9 Considere um projeto API alternativo, que oferece uma interface para cada camada software do protocolo (por exemplo, o API permite que um programa aplicativo envie e receba pacotes brutos sem usar IP, ou envie e receba datagramas IP sem usar UDP ou TCP). Quais são as vantagens de ter essa interface? E as desvantagens?

21.10 Um cliente e servidor podem ser executados no mesmo computador e usar um socket TCP para se comunicarem. Explique como é possível montar um cliente e um servidor que possam se comunicar em uma única máquina sem descobrir o endereço IP do host.

21.11 Experimente o servidor de exemplo deste capítulo para ver se pode gerar conexões TCP de modo suficientemente rápido para exceder o backlog que o servidor especifica. Você acha que as requisições de conexão que chegam ultrapassam o backlog mais rápido se o servidor operar em um computador que tem um centro do que em um computador que tem quatro centros? Explique.

21.12 Algumas das funções do soquete API original agora são irrelevantes. Faça uma lista das funções de socket que não são mais úteis.

21.13 Leia mais sobre escopo de endereço IPv6. Se um servidor vincula um socket a um endereço com escopo de link local, que computadores podem contatar o servidor?

21.14 Se um programador quer criar um servidor que pode ser alcançado tanto através de IPv4 quanto de IPv6, que funções socket pode o programador usar e como devem ser especificados os endereços?

* Os programadores frequentemente usam o termo WINSOCK para se referir ao *Windows Sockets*.

** O termo *descritor de arquivo* surge porque, no UNIX, os dispositivos são mapeados no sistema de arquivos.

* Termos em letras maiúsculas utilizados ao longo dos exemplos são os nomes de constantes simbólicas que os programadores usam com o socket API.

** Se um cliente não chama *bind*, o sistema operacional atribui um número de porta automaticamente; em geral, os números de porta são atribuídos em sequência.

* A versão da chamada *select* no Windows Sockets é aplicável somente aos descritores do tipo socket.

* O Capítulo 23 considera o sistema de domínio de nomes em detalhes.

CAPÍTULO 22

Bootstrap e autoconfiguração (DHCP, NDP, IPv6-ND)

CONTEÚDOS DO CAPÍTULO

22.1 Introdução

Os capítulos anteriores explicam como os protocolos TCP/IP operam no estado estacionário. Esses capítulos assumem que hosts e roteadores estão funcionando e que o software de protocolo foi configurado e inicializado. Este capítulo analisa a inicialização do sistema e discute as etapas para inicializar a pilha de protocolos. Curiosamente, o capítulo explica que muitos sistemas utilizam o paradigma cliente-servidor como parte de seu processo de inicialização (bootstrap) e, em particular, considera um computador host conectado a uma internet TCP/IP. Ele explica como o computador pode obter um endereço IPv4 ou IPv6 e as informações associadas, incluindo uma máscara de endereço, um prefixo de rede e os endereços de um roteador-padrão e um servidor de nome. O capítulo, ainda, descreve os protocolos que um host pode usar para obter as informações necessárias. Essa inicialização automática é importante porque permite que um usuário conecte um computador à Internet sem compreender os detalhes de endereços, máscaras, roteadores ou como configurar o software de protocolo. O fechamento do capítulo consiste em uma discussão sobre descoberta de vizinho IPv6, que lida com tarefas tais como o endereço de ligação, além de configuração.

Os procedimentos de boostrapping descritos aqui são surpreendentes porque usam o IP para transferir mensagens. Pode parecer impossível usar IP antes que o computador aprenda seu próprio endereço IP. Veremos, no entanto, que os endereços IP especiais descritos anteriormente tornam essa comunicação possível.

22.2 História do bootstrapping IPv4

Como o Capítulo 6 descreve, o protocolo RARP foi inicialmente desenvolvido para permitir que um computador obtenha um endereço IPv4. Posteriormente um protocolo mais geral chamado BOOTstrap Protocol (BOOTP) substituiu o RARP. Finalmente, o *Dynamic Host Configuration Protocol* (*DHCP*) foi desenvolvido como uma extensão do BOOTP.* Como o DHCP foi derivado do BOOTP, nossa descrição do básico geralmente se aplica a ambos. Para simplificar o texto, no entanto, iremos nos concentrar principalmente no DHCP.

Como usa o UDP e o IP, o DHCP pode ser implementado por um programa aplicativo. Assim como outros protocolos de aplicação, o DHCP segue o modelo cliente-servidor. No caso mais simples, exige apenas uma única troca de pacotes em que um computador host envia um pacote para requisitar informações de boostrap e um servidor responde enviando um único pacote que especifica itens necessários na partida, incluindo o endereço IPv4 do computador, o endereço IPv4 de um roteador-padrão e o endereço IPv4 de um servidor de nome de domínio. O DHCP também inclui na resposta uma *opção* de fornecedor específico que permite ao fornecedor enviar informações adicionais usadas apenas para seus computadores.**

22.3 Usando IP para determinar um endereço IP

Dissemos que o DHCP utiliza UDP para transportar mensagens, e que mensagens UDP são encapsuladas em datagramas IP para entrega. Para entender como um computador pode enviar DHCP em um datagrama IP antes de aprender seu endereço IP, lembre-se do Capítulo 5, de que existem vários endereços IP de caso especial. Particularmente, quando usado como um endereço de destino, o endereço IPv4 consistindo de todos os bits *1* (255.255.255.255) especifica broadcast limitado. O software IP pode aceitar e difundir datagramas que especificam o endereço de broadcast limitado mesmo antes de ter descoberto suas informações de endereço IP locais. A questão importante é descrita a seguir.

Um programa aplicativo pode usar o endereço IPv4 de broadcast limitado para obrigar o software IP a difundir um datagrama na rede local antes que o software IP no host tenha descoberto seu endereço IP.

Suponha que uma máquina cliente *A* queira usar o DHCP para encontrar informações de boostrap (incluindo seu endereço IPv4) e também que *B* seja o servidor na mesma rede física que responderá à requisição. Como *A* não sabe o endereço IPv4 de *B* ou o prefixo IP da rede, precisa difundir sua requisição DHCP inicial usando o endereço IPv4 de broadcast limitado. *B* pode enviar uma resposta direcionada? Não, embora ele conheça o endereço IPv4 de *A*. Para ver o porquê, considere o que acontece se um programa aplicativo em *B* tentar enviar um datagrama usando o endereço IP de *A*. Após rotear o datagrama, o software IP em *B* passará o datagrama para o software de interface de rede. Este precisa mapear o endereço IPv4 do próximo salto para um endereço de hardware correspondente. Se a interface de rede usar ARP como descrito no Capítulo 6, o ARP irá falhar – *A* ainda não recebeu resposta do DHPC, então não reconhece seu endereço IP. Portanto, *A* não pode responder à requisição ARP de *B*. Como consequência, *B* tem apenas duas alternativas: pode difundir a resposta de volta para *A* ou pode extrair o endereço MAC de *A* do frame que levou a requisição e usá-lo para enviar uma resposta direcionada. A maioria das pilhas de protocolo não permite que um aplicativo crie e envie um frame de Camada 2 arbitrário. Assim, uma técnica consiste em extrair o endereço MAC de *A* do pacote de requisição e adicionar a entrada ao cache ARP local de *A*. Uma vez que a entrada tenha sido colocada no cache do ARP, pacotes de saída serão enviados para *A* (até que a entrada expire).

22.4 Retransmissão e randomização DHCP

O DHCP coloca toda a responsabilidade pela segurança da comunicação no cliente. Sabemos que, como o UDP usa IP para entrega, as mensagens podem ser atrasadas, perdidas, entregues fora de ordem ou duplicadas.

Além disso, como o IP não fornece um checksum para dados, o datagrama UDP poderia chegar com alguns bits danificados. Para se prevenir de danos, o DHCP exige que o UDP tenha um checksum ativo. O padrão DHCP também especifica que requisições e respostas devem ser enviadas com o bit *não fragmentar* definido para acomodar clientes que têm muito pouca memória para reconstituir datagramas. Finalmente, para lidar com duplicação, O DHCP é construído para permitir múltiplas respostas; o protocolo só aceita e processa a primeira resposta.*

Para lidar com perda de datagrama, o DHCP usa a técnica convencional do *timeout e retransmissão*. Quando o cliente transmite uma requisição, ele inicia um timer. Se nenhuma resposta chegar antes que o timer expire, a requisição precisa ser retransmitida. É claro que, após uma interrupção de energia, todas as máquinas em uma rede reiniciarão simultaneamente, possivelmente sobrecarregando o servidor ou servidores DHCP com requisições simultâneas. Da mesma forma, se todos os clientes usarem exatamente o mesmo timeout de retransmissão, muitos ou todos eles tentarão retransmitir simultaneamente. Para evitar ações simultâneas, a especificação DHPC recomenda adicionar um atraso aleatório. Além disso, a especificação recomenda iniciar com um valor de timeout aleatório entre *0* e *4* segundos, e dobrar o timer após cada retransmissão. Após o timer atingir um valor alto, *60* segundos, o cliente não o aumenta, mas continua a usar a aleatoriedade. Dobrar o timeout após cada retransmissão evita que o DHCP acrescente tráfego excessivo a uma rede congestionada; a aleatoriedade ajuda a evitar transmissões simultâneas.

22.5 Formato de mensagem DHCP

Para manter uma implementação o mais simples possível, as mensagens DHCP possuem campos de tamanho fixo, e as respostas têm o mesmo formato das requisições. Embora tenhamos dito que clientes e servidores são programas, o protocolo DHCP usa os termos livremente, referindo-se à máquina que envia a requisição DHCP como o *cliente* e a qualquer máquina que envia uma resposta como *servidor*. A Figura 22.1 mostra o formato de mensagem DHCP.

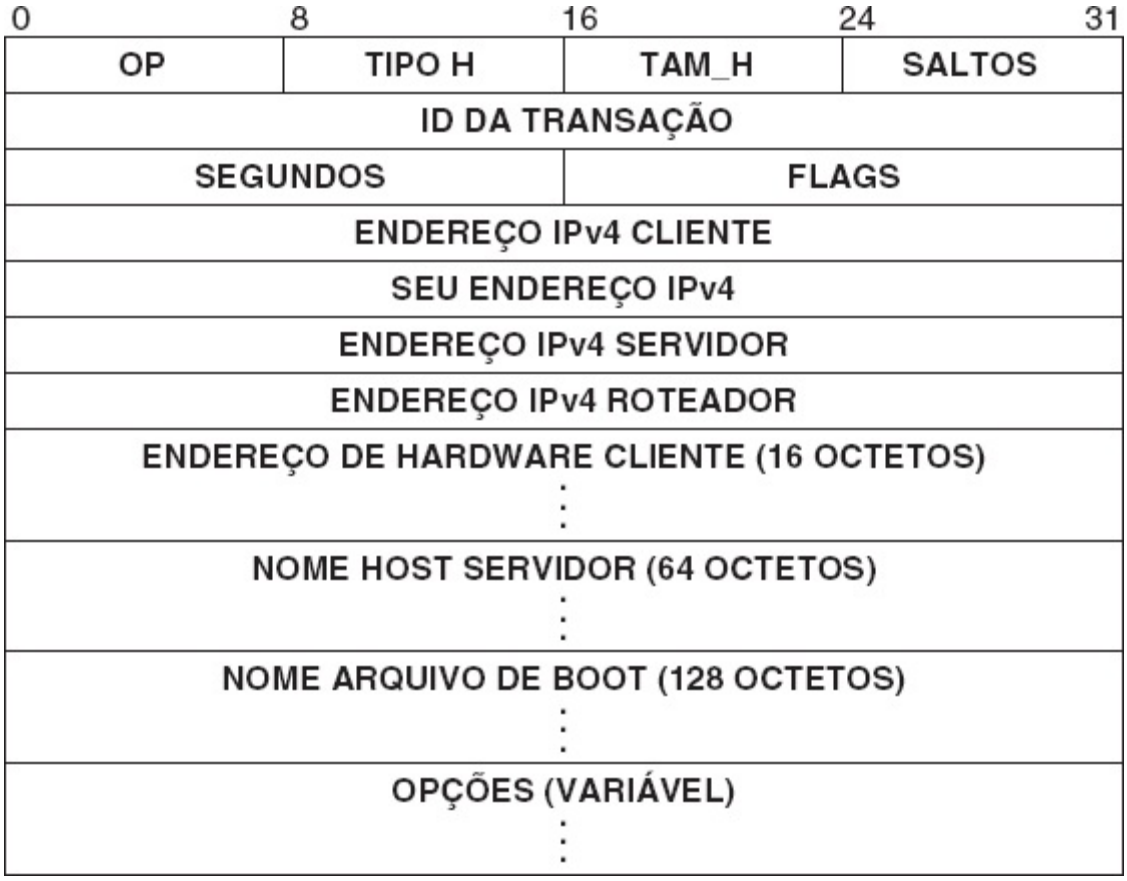


Figura 22.1 O formato de uma mensagem DHCP. O protocolo utiliza campos fixos para manter o software DHCP pequeno o suficiente para caber na ROM.

O campo OP especifica se a mensagem é uma requisição (*1*) ou uma resposta (*2*). Como no ARP, os campos TIPO_H e TAM_H especificam o tipo de hardware de rede e o tamanho do endereço de hardware (por exemplo, Ethernet tem tipo *1* e o tamanho de endereço *6*).* O cliente coloca *0* no campo SALTOS. Se receber a requisição e decidir passá-la para outra máquina (por exemplo, para permitir o boostrapping através de múltiplos roteadores), o servidor DHCP incrementa a contagem de SALTOS. O campo ID DA TRANSAÇÃO contém um inteiro que clientes usam para adequar as respostas às requisições. O campo SEGUNDOS informa o número de segundos desde que o cliente inicializou.

O campo ENDEREÇO IPv4 CLIENTE e todos os campos que o seguem contêm as informações mais importantes. Para permitir maior flexibilidade, os clientes preenchem o máximo de informações que eles conhecem e deixam os campos restantes definidos em zero. Por exemplo, se um cliente

sabe o nome ou endereço de um servidor específico do qual queira informações, ele pode preencher os campos ENDEREÇO IPv4 SERVIDOR ou NOME HOST SERVIDOR. Se esses campos forem não zero, apenas o servidor com nome/endereço correspondente responderá à requisição; se eles forem zero, qualquer servidor que receba a requisição responderá.

O DHCP pode ser usado através de um cliente que já conheça seu endereço IPv4 (ou seja, para obter outras informações). Um cliente que conhece o endereço IP o coloca no campo ENDEREÇO IPv4 CLIENTE; outros clientes usam zero. Se o endereço IP do cliente for zero na requisição, um servidor retorna o endereço IP do cliente no campo SEU ENDEREÇO IP.

O campo FLAG de 16 bits permite o controle da requisição e resposta. Como a Figura 22.2 mostra, é atribuído um significado apenas ao bit de ordem superior do campo FLAGS.

Figura 22.2 Formato do campo FLAGS de 16 bits em uma mensagem DHCP. O bit mais à esquerda é interpretado como uma requisição de broadcast; todos os outros bits precisam ser definidos em zero.

Um cliente usa o bit de ordem superior no campo *FLAGS* para controlar se o servidor envia a resposta via unicast ou broadcast. Para entender por que um cliente poderia escolher uma resposta de broadcast,, lembre-se de que, enquanto se comunica com um servidor DHCP, um cliente ainda não tem um endereço IP, o que significa que não pode responder a consultas ARP. Portanto, para garantir que possa receber mensagens enviadas por um servidor DHCP, um cliente pode requisitar que o servidor envie respostas usando broadcast IP, que corresponde ao broadcast de hardware. As regras para o processamento de datagrama permitem que o IP descarte qualquer datagrama que chegue via unicast de hardware se o endereço de destino não corresponder ao endereço do computador. Entretanto, o IP precisa aceitar e tratar qualquer datagrama enviado para o endereço de broadcast IP.

Interessantemente, o DHCP não provê espaço na mensagem para baixar uma imagem de memória específica para um sistema incorporado. Em vez disso, o DHCP fornece um campo *NOME ARQUIVO DE BOOT* que um

pequeno sistema sem disco pode usar. O cliente pode usar o campo para fornecer um nome genérico como "unix", que significa "quero inicializar o sistema operacional UNIX para esta máquina". O servidor DHCP consulta seu banco de dados de configuração a fim de mapear o nome genérico para um nome de arquivo específico que contém a imagem de memória apropriada para o hardware de cliente e retorna o nome de arquivo totalmente qualificado em sua resposta. É claro, o banco de dados de configuração também permite boostrapping completamente automático, em que o cliente coloca zeros no campo *NOME ARQUIVO DE BOOT* e o DHCP seleciona uma imagem de memória para a máquina. O cliente, então, usa um protocolo de transferência de arquivo padrão, como o TFTP, para obter a imagem. A vantagem do método é que um cliente sem disco pode usar um nome genérico sem codificar um arquivo específico e o gerente de rede pode mudar o local de uma imagem de inicialização sem mudar a ROM nos sistemas incorporados.

Todos os itens na área OPÇÕES usam uma codificação de estilo *Tipo-Tamanho-Valor* (*TTV*) – cada item contém um octeto de *tipo*, um octeto de *tamanho* e termina com um *valor* do tamanho especificado. Duas opções são especialmente significativas: uma máscara sub-rede IPv4 para a rede local e um endereço IPv4 de um roteador-padrão.

22.6 A necessidade de configuração dinâmica

Os primeiros protocolos de bootstrap operavam em um ambiente relativamente estático em que cada host tinha uma conexão de rede permanente. Um gerente criava um arquivo de configuração que especificava um conjunto de parâmetros para cada host, incluindo um endereço IP. O arquivo não mudava frequentemente porque a configuração normalmente permanecia estável. Em geral, uma configuração permanecia inalterada por semanas.

Na Internet moderna, porém, os provedores de serviço possuem um conjunto de clientes que muda continuamente, e os computadores laptop portáteis com conexões sem fio possibilitam mover um computador de um local para outro de maneira rápida e fácil. Para tratar a atribuição de endereço automatizada, o DHCP permite que um computador obtenha um endereço IP rápida e dinamicamente. Ou seja, ao configurar um servidor DHCP, um gerente fornece um conjunto de endereços IPv4. Sempre que um novo computador se conecta a uma rede, ele contata o servidor e requisita um endereço. O servidor escolhe um dos endereços do conjunto que o

gerente especificou e o aloca ao computador.

Para ser completamente geral, o DHCP permite três tipos de atribuição de endereço descritos a seguir.

- Estático.
- Automático.
- Dinâmico.

E um gerente escolhe como o DHCP responderá para cada rede ou cada host. Como seu predecessor, BOOTP, o DHCP permite configuração *estática* em que um gerente configura manualmente um endereço específico para um dado computador. O DHCP também permite uma forma *automática* de configuração de endereço em que um gerente permite a um servidor DHCP atribuir um endereço permanente assim que um computador se conecta a uma rede. Finalmente, o DHCP permite a configuração *dinâmica* de endereço na qual um servidor "aluga" um endereço para um computador por um tempo limitado. A atribuição dinâmica de endereço é o aspecto mais poderoso e novo do DHCP.

Um DHCP usa a identidade de um cliente e um arquivo de configuração para decidir como proceder. Quando um cliente contata um servidor DHCP, envia um identificador, normalmente o endereço de hardware do cliente. O servidor usa o identificador do cliente (e a rede através da qual a requisição chegou) para determinar como atribuir-lhe um endereço IP. Portanto, um gerente possui controle completo sobre como os endereços são atribuídos. Um servidor pode ser configurado para atribuir estatisticamente um endereço IPv4 para um computador, enquanto permite que outros computadores obtenham endereços automaticamente ou dinamicamente. Podemos fazer o resumo a seguir.

O DHCP permite que um computador obtenha todas as informações necessárias para se comunicar em uma determinada rede (incluindo endereço IPv4, máscara de sub-rede e endereço de um roteador-padrão) quando o computador inicia.

22.7 Locação DHCP e atribuição de endereço dinâmico

A atribuição de endereço dinâmico de DHCP é temporária. Dizemos que

um servidor DHCP *aluga* um endereço para um cliente por um período de tempo finito. O servidor especifica o período de locação, quando se atribui o endereço. Durante o período de locação, o servidor não irá alugar o mesmo endereço para outro cliente. No final do período de locação, o cliente deve renovar o contrato ou parar de usar o endereço.

Quanto tempo deve durar uma concessão de DHCP? O tempo ideal para uma locação depende da rede em particular e das necessidades de um determinado host. Por exemplo, para garantir que os endereços possam ser reciclados rapidamente, os computadores em uma rede usados por estudantes em um laboratório de universidade podem ter um período de locação curto (por exemplo, uma hora). Por outro lado, uma rede corporativa pode usar um período de locação de um dia ou uma semana. Um ISP pode fazer a duração de um contrato de arrendamento depender do contrato de um cliente. Para acomodar todos os ambientes possíveis, o DHCP não especifica um período de locação fixo constante. Em vez disso, o protocolo permite que um cliente solicite um período de locação determinado, e permite que um servidor informe ao cliente do período de locação que concede. Assim, um gerente pode decidir por quanto tempo cada servidor deve atribuir um endereço a um cliente. Em casos extremos, o DHCP reserva um valor ao *infinito* para permitir a um contrato de arrendamento uma duração arbitrariamente longa (ou seja, para fazer uma atribuição de endereço permanente).

22.8 Endereços múltiplos e relays

Um computador multirresidente se conecta a mais de uma rede. Quando esse computador inicializa, pode precisar obter informações de configuração para cada uma de suas interfaces. Como vimos, uma mensagem DHCP só fornece ao computador um endereço IPv4 e informações sobre uma interface (por exemplo, máscara de sub-rede) para uma rede. O projeto DHCP significa um computador com interfaces múltiplas deve tratar de cada interface separadamente. Assim, embora descrevamos o DHCP como se um computador precisasse apenas de um endereço, o leitor precisa lembrar que cada interface de um computador multirresidente precisa de seu próprio endereço. Se um host multirresidente escolhe enviar requisições para múltiplas interfaces, o software cliente do DHCP para cada interface pode estar em um ponto diferente no protocolo.

O DHCP usa a noção de *agente de relay* para permitir que um

computador contate um servidor em uma rede não local. Quando um agente de relay, normalmente um roteador, recebe uma requisição de broadcast de um cliente, ele encaminha a requisição para um servidor DHCP e, depois, retorna a resposta do servidor para o host. Os agentes de relay podem complicar a configuração multirresidente porque um servidor pode receber múltiplas requisições do mesmo computador. Embora o DHCP use o termo *identificador do cliente*, consideramos que um cliente multirresidente envia um identificador diferente para cada interface (por exemplo, um endereço de hardware único para cada interface). Portanto, um servidor sempre será capaz de distinguir entre requisições de um host multirresidente, mesmo quando o servidor recebe essas requisições através de um agente de relay.

22.9 Estados de aquisição de endereço DHCP

Ao usar o DHCP para obter um endereço IPv4, um cliente está em um de seis estados. O diagrama de transição de estado na Figura 22.3 mostra eventos e mensagens que fazem um cliente mudar de estado.

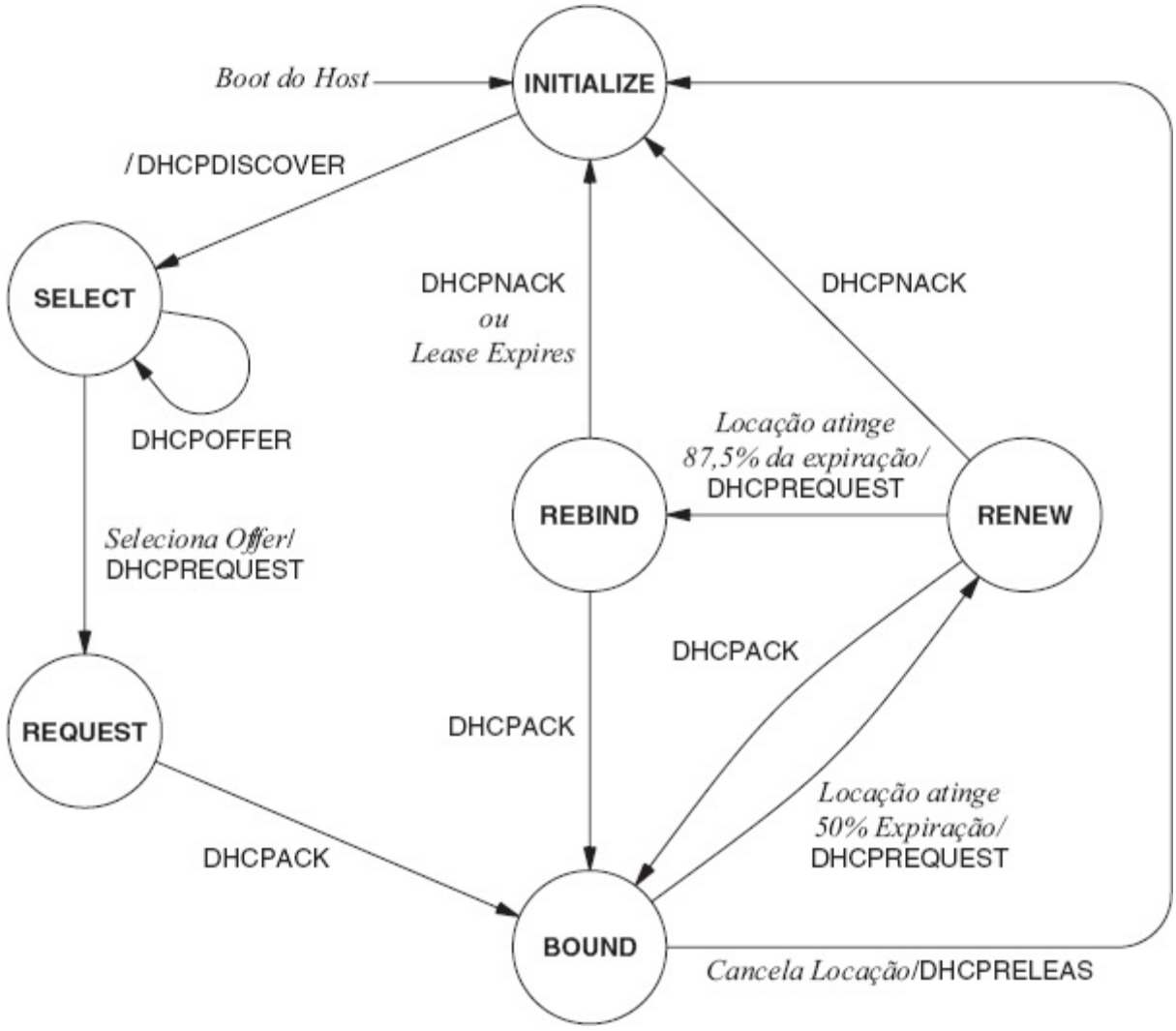


Figura 22.3 Os seis principais estados de um cliente DHCP e as transições entre eles. Cada rótulo em uma transição lista a mensagem ou evento de entrada que causa a transmissão, seguido de uma barra e a mensagem que o cliente envia.

Quando faz o boot inicial, um cliente entra no estado INITIALIZE. Para começar a adquirir um endereço IPv4, o cliente primeiro contata todos os servidores DHCP na rede local. Para isso, o cliente difunde uma mensagem DHCPDISCOVER e muda para o estado rotulado como SELECT. Como o protocolo é uma extensão do BOOTP, o cliente envia a mensagem DHCPDISCOVER em um datagrama UDP com a porta de destino definida para a porta BOOTP (ou seja, porta 67). Todos os servidores DHCP na rede local recebem a mensagem, e aqueles servidores que foram programados para responder a um determinado cliente enviam uma mensagem DHCPOFFER. Assim, um cliente pode receber zero ou mais respostas.

Enquanto está no estado SELECT, o cliente coleta respostas DHCPOFFER dos servidores DHCP. Cada oferta contém informações de configuração para o cliente junto com um endereço IPv4 que o servidor está oferecendo para alugar ao cliente. O cliente precisa escolher uma das respostas (por exemplo, a primeira a chegar) e negociar um aluguel com o servidor. Para isso, o cliente envia ao servidor uma mensagem DHCPREQUEST e entra no estado REQUEST. Para reconhecer o recebimento da requisição e iniciar o aluguel, o servidor responde enviando um DHCPACK. A chegada de um reconhecimento faz com que o cliente mude para o estado BOUND, quando o cliente passa a usar o endereço. Podemos fazer o resumo a seguir.

Para usar o DHCP, um host se torna um cliente pela difusão de uma mensagem para todos os servidores na rede local. O host, então, coleta ofertas dos servidores, seleciona uma das ofertas e verifica a aceitação com o servidor.

22.10 Término precoce do aluguel

Pensamos no estado BOUND como o estado normal da operação; um cliente normalmente permanece no estado BOUND enquanto usa o endereço IP que adquiriu. Se um cliente possui armazenamento secundário (por exemplo, um disco local), o cliente pode armazenar o endereço IPv4 ao qual foi atribuído e requisitar o mesmo endereço quando reiniciar novamente. Em alguns casos, entretanto, um cliente no estado BOUND pode descobrir que não precisa mais de um endereço IP. Por exemplo, suponha que um usuário conecte um laptop a uma rede, use DHCP para adquirir um endereço IP e, então, use o computador para ler e-mail. O protocolo especifica que um aluguel precisa durar no mínimo uma hora, o que pode ser mais tempo do que as necessidades do usuário.

Quando um endereço não é mais necessário, o DHCP permite que um cliente termine o aluguel prematuramente. O término precoce é especialmente importante se o número de endereços IP que um servidor tem disponível for muito menor que o número de computadores que se conectam à rede. Se cada cliente terminar seu aluguel assim que o endereço IP não for mais necessário, o servidor será capaz de atribuir o endereço a outro cliente.

Para terminar um aluguel precocemente, um cliente envia uma mensagem DHCPRELEASE para o servidor. Liberar um endereço é uma

ação final que evita que o cliente use-o por mais tempo. Portanto, após transmitir a mensagem de liberação, o cliente não pode enviar mais datagrama algum que use o endereço. Em termos do dispositivo de transição de estado da Figura 22.3, um host que envia um DHCPRELEASE deixa o estado BOUND e precisa iniciar no estado INITIALIZE novamente antes de poder usar o IP.

22.11 Estados de renovação de aluguel

Dissemos que quando adquire um endereço, um cliente DHCP muda para o estado BOUND. Ao entrar no estado BOUND, o cliente define três timers que controlam a renovação, a reassociação e a expiração do aluguel. Um servidor DHCP pode especificar valores explícitos dos timers quando aloca um endereço para o cliente; se o servidor não especificar valores de timer, o cliente usa padrões. O valor-padrão para o primeiro timer é metade do tempo de aluguel total. Quando o primeiro timer expira, o cliente precisa tentar renovar seu aluguel. Para requisitar uma renovação, o cliente envia uma mensagem DHCPREQUEST ao servidor do qual o aluguel foi obtido. Então, passa para o estado RENEW a fim de esperar uma resposta. O DHCPREQUEST contém o endereço IP que o cliente está usando atualmente e pede ao servidor para estender o aluguel no endereço. Como na negociação de aluguel inicial, um cliente pode requisitar um período para a extensão, mas o servidor, no final, controla a renovação. Um servidor pode responder à requisição de renovação de um cliente de duas maneiras: ele pode instruir o cliente a parar de usar o endereço ou pode aprovar o uso continuado. Se aprovar, envia um DHCPACK, que faz com que o cliente retorne ao estado BOUND e continue a usar o endereço. O DHCPACK também pode conter novos valores para os timers do cliente. Se um servidor desaprovar o uso continuado, ele envia um DHCPNACK (reconhecimento negativo), o que faz com que o cliente pare de usar o endereço imediatamente e retorne ao estado INITIALIZE.

Após enviar uma mensagem DHCPREQUEST que requisita uma extensão em seu aluguel, um cliente permanece no estado RENEW esperando uma resposta. Se nenhuma resposta chegar, o servidor que concedeu o aluguel está desativado ou inalcançável. Para tratar dessa situação, o DHCP utiliza um segundo timer, que foi definido quando o cliente entrou no estado BOUND. O segundo timer expira após *87,5%* do período de aluguel e faz com que o cliente mude do estado RENEW para o estado REBIND. Ao fazer a transição, o cliente considera que o antigo servidor DHCP está indisponível e começa a difundir uma mensagem

DHCPREQUEST para qualquer servidor na rede local. Qualquer servidor configurado para fornecer serviço ao cliente pode responder positivamente (isto é, estender o aluguel) ou negativamente (isto é, negar mais uso do endereço IP). Se receber uma resposta positiva, o cliente retorna para o estado BOUND e reinicia os dois timers. Se receber uma resposta negativa, o cliente precisa mudar para o estado INITIALIZE, parar imediatamente de usar o endereço IP e adquirir um novo endereço IP antes de continuar a usar o IP.

Após mudar para o estado REBIND, um cliente terá pedido uma extensão de aluguel ao servidor original mais todos os servidores na rede local. Na remota possibilidade de que o cliente não receba uma resposta de qualquer servidor antes que seu terceiro timer expire, o aluguel expira. O cliente precisa parar de usar o endereço IP, voltar ao estado INITIALIZE e começar a adquirir um novo endereço.

22.12 Opções DHCP e tipos de mensagem

Surpreendentemente, o DHCP não aloca campos fixos no cabeçalho da mensagem para a mensagem de tipo ou informações de aluguel. Em vez disso, mantém o formato BOOTP da mensagem e usa o campo OPÇÕES para identificar a mensagem como DHCP. A Figura 22.4 ilustra a opção *tipo de mensagem DHCP* usada para especificar que mensagem DHCP está sendo enviada.

CAMPO TIPO	DHCP correspondente TIpo de mensagem
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNACK
7	DHCPRELEASE
8	DHCPINFORM

Figura 22.4 O formato de uma opção DHCP usada para especificar o tipo de mensagem DHCP com uma lista dos valores possíveis para o terceiro octeto.

Mais de 200 OPÇÕES foram definidas para uso em uma resposta DHCP;

cada uma tem um campo tipo e tamanho que juntos determinam o tamanho da opção. As atribuições são um tanto aleatórias, pois os fornecedores usaram valores que eram inicialmente reservados. Como se atribui códigos, o IETF decidiu evitar conflitos, evitando códigos que os fornecedores estavam usando. A Figura 22.5 lista algumas das opções possíveis.

Tipo de Item	Código do Item	Tamanho Octeto	Conteúdos do Valo
Subnet mask	1	4	Máscara de sub-rec a usar
Routers	3	N	Endereços IPv4 de N/4 roteadores
DNS Servers	6	N	Endereços IPv4 de N/4 servidores
Hostname	12	N	N bytes do nome de host do cliente
Boot Size	13	2	Tamanho inteiro de octetos do arquivo inicialização
Default IP TTL	23	1	Valor para datagrama TTL
NTP Servers (time)	42	N	Endereços IPv4 de N/4 servidores
Mail Servers (SMTP)	69	N	Endereços IPv4 de N/4 servidores
Web Servers	72	N	Endereços IPv4 de N/4 servidores

Figura 22.5 Exemplos de OPTIONS que podem estar presentes em uma resposta DHCP IPv4.*

22.13 Sobrecarga de opção DHCP

Os campos NOME HOST SERVIDOR e NOME ARQUIVO INICIALIZAÇÃO no cabeçalho de mensagem DHCP ocupam, cada um, muitos octetos. Se uma determinada mensagem não contiver informações em nenhum desses campos, o espaço é desperdiçado. Para permitir que um servidor DHCP use os dois campos para outras opções, o DHCP define uma opção *Sobrecarga de Opção*. Quando presente, a opção de sobrecarga diz a um receptor para ignorar o significado usual dos campos NOME HOST SERVIDOR e NOME ARQUIVO INICIALIZAÇÃO e procurar opções nos campos.

22.14 DHCP e nomes de domínio

Embora possa alocar um endereço IP para um computador sob demanda, o DHCP não automatiza completamente todos os procedimentos necessários para conectar um host permanente a uma internet. Em particular, o protocolo DHCP não especifica qualquer interação com o *Sistema de Nome de Domínio (DNS).** Portanto, a menos que um mecanismo adicional seja usado, a vinculação entre um nome de host e o endereço IP que o DHCP atribui ao host permanecerá independente.

Apesar da falta de um padrão, alguns servidores DHCP, sem dúvida, interagem com o DNS quando atribuem um endereço. Por exemplo, os sistemas UNIX, tal como o Linux ou BSD, se organizam para coordenar o DHCP com o software de DNS, o que é conhecido como *named bind*** ou simplesmente *bind*. Da mesma forma, o software Microsoft DHCP se coordena com o software Microsoft DNS para garantir que um host ao qual é atribuído um endereço DHCP também tenha um nome de domínio. Os mecanismos de coordenação também funcionam ao contrário para garantir que, quando um aluguel DHCP for revogado, um servidor DHCP notifique o DNS para revogar o nome correspondente.

22.15 Configuração gerenciada e não gerenciada

Existem duas grandes abordagens para a configuração de dispositivos de rede que têm consequências tanto para a infraestrutura de rede como para os protocolos de configuração:

- gerenciada;
- não gerenciada.

Gerenciada. Um sistema *gerenciado* requer que os operadores de rede instalem e configurem servidores. Quando um computador se une a uma rede, contacta um servidor de configuração para obter informação sobre endereçamento, roteamento e outros serviços. Embora seja difícil prever serviços no abstrato, nossa discussão de DHCP torna o conceito claro, pois o DHCP é geralmente usado como um exemplo canônico de configuração gerenciada.

Não gerenciada. Um sistema *não gerenciado* não requer que um gerente de rede atribua endereços nem requer servidores de configuração.

Em vez disso, quando um computador se junta à rede, gera um endereço único, e então o usa para se comunicar.

O protocolo *AppleTalk* original ilustra um sistema não gerenciado: quando se juntou a uma rede, um computador usou um gerador de números aleatórios para escolher um endereço e, em seguida, transmitiu uma mensagem para verificar se o endereço já não estava em uso. Se outro computador já estava usando o endereço, um novo valor aleatório foi selecionado até que um endereço único foi encontrado. Não foi necessária nenhuma outra configuração, porque os serviços foram alcançados por requisições broadcasting.

Cada abordagem à configuração apresenta vantagens e desvantagens. Uma rede não gerenciada tem a vantagem de não exigir que humanos configurem e operem um conjunto de servidores. Assim, computadores e outros dispositivos (por exemplo, impressoras) podem se conectar e comunicar automaticamente. Infelizmente, a abordagem não gerenciada também tem desvantagens. A atribuição de endereço aleatório pode levar a conflitos se um computador estiver temporariamente desconectado ou ocupado quando um novo computador ingressar e escolher o mesmo endereço. Além disso, como o tamanho da rede aumenta, o uso de broadcast torna-se um problema – uma abordagem não gerenciada pode funcionar em uma única rede, mas não através da Internet global.

Uma abordagem gerenciada tem a principal vantagem de dar a cada proprietário de rede o controle total sobre os computadores e dispositivos conectados. Os gerentes de rede geralmente preferem a abordagem gerenciada porque é necessária uma equipe experiente para outras tarefas, e um servidor de configuração pode ser executado em hardware com outros servidores.

22.16 Configuração gerenciada e não gerenciada para o IPv6

Quando o IPv6 foi inicialmente concebido, os desenvolvedores pensaram em um caso especial: dois hosts IPv6 que se conectam sem quaisquer servidores em sua rede. Por exemplo, considere dois dispositivos móveis IPv6 que têm capacidade de Wi-Fi. Os desenvolvedores pensaram que deveria ser possível para os dispositivos comunicarem-se diretamente sem a necessidade de uma estação base e sem a necessidade de um servidor para fornecer endereços. Consequentemente, os desenvolvedores adotaram uma abordagem não gerenciada em que a atribuição de endereços é automatizada. Eles usam o termo IPv6 stateless autoconfiguration para

descrever o esquema de atribuição de endereços IPv6. Sempre que um host se junta a uma rede não gerenciada, ele emprega stateless autoconfiguration para gerar um endereço IPv6 e começar a comunicação. Assim, stateless autoconfiguration significa que hosts podem se comunicar sem a necessidade de um servidor para distribuir endereços.

Muitos gerentes opuseram-se ao stateless autoconfiguration. Os operadores de rede que gerenciam grandes redes de ISP comerciais ficaram especialmente desapontados. Uma vez que gerenciam serviços com fins lucrativos, que cobram os clientes por conexões de rede, os operadores queriam controlar quais hosts conectar à sua rede (ou seja, para excluir os não clientes). Em particular, eles queriam um serviço gerenciado que lhes desse o controle sobre a atribuição de endereços.

Em termos de serviços de atribuição de endereços gerenciados, o DHCP é amplamente aceito como o padrão da indústria. Os operadores de rede gostam do DHCP porque este dá um controle preciso sobre a forma como os endereços são atribuídos. Um gerente pode escolher a política de atribuição em uma base host-por-host pela pré-atribuição de um endereço IP fixo para um determinado host ou permitir que ele obtenha um endereço automaticamente a partir de um pool.

22.17 Opções de configuração do IPv6 e conflitos potenciais

Para satisfazer os operadores de rede que querem uma solução gerenciada e indivíduos que querem ser capazes de criar redes ad-hoc, o IETF decidiu endossar duas abordagens para a configuração de endereço IPv6, as quais são:

- gerenciada via DHCPv6;
- não gerenciada via stateless autoconfiguration.

Gerenciada via DHCPv6. Uma nova versão do DHCP foi criada para IPv6. Nomeada *DHCPv6*, é conceitualmente semelhante ao DHCP original. Como a original, por exemplo, a nova versão requer um servidor que esteja disponível para cada rede e um host que se contate ao servidor para obter um endereço IP. No entanto, como o IPv6 não suporta broadcast, um host não pode usar broadcast para chegar a um servidor DHCPv6 da mesma forma que um host IPv4 transmite uma solicitação DHCP. Em vez disso, o IPv6 permite que um host gere um endereço link-local e use multicast link-local, que é efetivamente o mesmo que um broadcast limitado IPv4.

Infelizmente, o DHCPv6 é substancialmente mais complexo do que o DHCP. Como a maior parte do IPv6, o DHCPv6 tenta acomodar todas as possibilidades, muda completamente o formato das mensagens e adiciona vários tipos novos de mensagens que dão funcionalidade adicional. Por exemplo, a especificação permite *delegação de prefixo* em que um servidor delega um conjunto de prefixos para outro servidor (por exemplo, a um roteador em casa para atribuição a aparelhos da casa). Parte da maior complexidade decorre da provisão IPv6 que permite que um host use vários prefixos de rede em uma determinada interface. Outras complexidades surgem porque o DHCPv6 permite autenticação. O resultado é que o RFC que define o DHCPv6 é mais de duas vezes o tamanho da RFC que define o DHCP.*

Não gerenciada via stateless autoconfiguration. Dissemos que, no IPv6, a configuração stateless autoconfiguration refere-se ao método de criação de endereço para um link não gerenciado. Stateless autoconfiguration depende do IPv6 *Neighbor Discovery Protocol* (NDP), descrito na próxima seção. Veremos que a NDP fornece muito mais funcionalidades do que a configuração de endereço gerenciado. No entanto, quando se compara NDP com DHCPv6 só precisamos considerar o básico: sem o uso de um servidor de configuração, um host pode gerar um endereço IPv6 e verificar se o endereço é único (ou seja, nenhum outro nó na rede está usando o mesmo endereço).

O uso de duas abordagens para a configuração IPv6 leva a perguntas. É uma abordagem preferida em detrimento de outra? Pode um determinado host utilizar ambas as abordagens? Se ambas forem utilizadas e os endereços IPv6 resultantes diferirem, pode o host manter os dois endereços IPv6 ou um endereço deve ser descontinuado? As normas não especificam preferências ou como lidar com conflitos de endereços. Em vez disso, apenas fornecem duas tecnologias alternativas. Podemos fazer o resumo a seguir.

Padrões IPv6 incluem esquemas de atribuição de endereço gerenciados e não gerenciados. As normas não especificam qual é o preferido ou como lidar com situações em que surgem conflitos.

22.18 Protocolo de descoberta de vizinho IPv6 (NDP)

O *protocolo de descoberta de vizinho IPv6* (*Neighbor Discovery Protocol – NDP –* ou *IPv6-ND*) inclui funcionalidade de baixo nível, tal como a resolução de endereços da Camada 2 e mensagens de redirecionamento de host. Portanto, pode parecer que o NDP pertence aos primeiros capítulos do texto. No entanto, a discussão foi adiada para este capítulo porque o NDP também inclui a funcionalidade de protocolos de camadas mais altas. Especificamente, o NDP fornece um mecanismo para a configuração de endereço.

O NDP opera na Camada 3 usando mensagens ICMPv6. A seguir, são listadas as principais funções que o NDP fornece.

- *Descoberta de roteador*: um host pode identificar o conjunto de roteadores em um dado link.
- *Rotas próximo salto*: um host pode encontrar o próximo salto para um determinado destino.
- *Descoberta de vizinho*: um nó pode identificar o conjunto de nós em um dado link.
- *Detecção de vizinho inalcançável (NUD):* um nó monitora vizinhos continuamente para saber quando um vizinho se torna inalcançável.
- *Descoberta de prefixo de endereço:* um host pode aprender o prefixo de rede(s) que está sendo usado em um link.
- *Descoberta de parâmetro de configuração:* um host pode determinar parâmetros, tal como o MTU utilizado em um dado link.
- *Stateless autoconfiguration:* um host pode gerar um endereço para uso em um link.
- *Detecção de Endereço Duplicado (DAD):* um nó pode determinar se um endereço que ele gera já está em uso.
- *Resolução de endereço:* um nó pode mapear um endereço IPv6 para um endereço MAC equivalente.
- *Descoberta de servidor* DNS: um nó pode encontrar o conjunto de servidores DNS em um link.
- *Redirecionar:* um roteador pode informar um nó sobre um primeiro roteador de salto preferido.

Para alcançar o que foi dito acima, o NDP define cinco tipos de mensagem ICMPv6, a saber:

- solicitação de roteador;
- anúncio de roteador;
- solicitação de vizinho;
- anúncio de vizinho;
- redirecionar.

Em vez de definir um tipo de mensagem único para cada uma das funções descritas anteriormente, o NDP usa uma combinação dos cinco tipos de mensagens ICMPv6 para alcançar cada função. As seções seguintes discutem cada um dos cinco tipos de mensagens.

22.19 Mensagem de solicitação de roteador ICMPv6

Um host envia uma mensagem de *Solicitação de Roteador* para requisitar resposta dos roteadores. A Figura 22.6 ilustra o formato de uma solicitação de roteador.

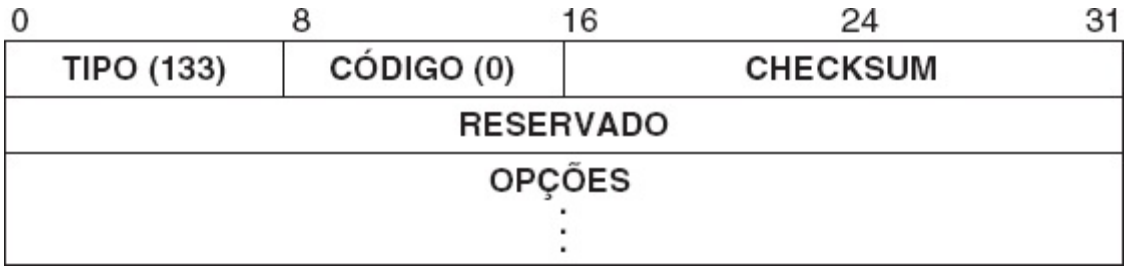


Figura 22.6 O formato de uma mensagem de *Solicitação de Roteador* ICMPv6.

Se um nó já conhece o seu endereço IP, o campo de OPÇÕES contém o endereço MAC do nó (*chamado de endereço da camada de link no IPv6*).

22.20 Mensagem de anúncio de roteador ICMPv6

Um roteador envia periodicamente uma mensagem *Anúncio de Roteador*, ou quando solicitado por uma solicitação de roteador. A mensagem permite que um roteador anuncie sua presença na rede e sua disponibilidade como um nó através do qual o tráfego off-link pode ser encaminhado. A Figura 22.7 ilustra o formato de um anúncio de roteador.

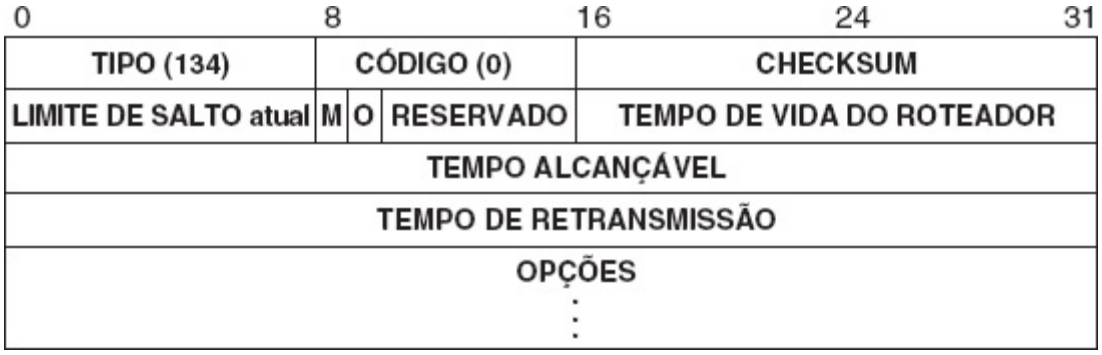


Figura 22.7 Formato de uma mensagem de Anúncio de Roteador ICMPv6.

O LIMITE DE SALTO ATUAL (CUR. HOP LIMIT) especifica um valor que deve ser utilizado como o LIMITE DE SALTO em cada datagrama de saída, o bit *M* especifica se a rede está usando a atribuição de endereços gerenciados (isto é, DHCPv6), e o bit *0* especifica se outras informações de configuração estão disponíveis via DHCPv6. Se o roteador pode ser utilizado como um roteador-padrão, o campo ROTEADOR TEMPO DE VIDA dá a quantidade de tempo que o roteador pode ser utilizado em segundos. O campo TEMPO ALCANÇÁVEL especifica por quanto tempo (em milissegundos) um vizinho permanece alcançável após ter respondido, e o campo TEMPO DE RETRANSMISSÃO (RETRANSMIT TIME) especifica com que frequência retransmitir mensagens de solicitação de vizinho. Opções possíveis incluem os endereços MAC de origem, a MTU usada no link e uma lista de um ou mais prefixos IPv6 usados no link.

22.21 Mensagem de solicitação de vizinho ICMPv6

Um nó envia uma mensagem de *Solicitação De Vizinho* por duas razões: para obter o endereço MAC de um vizinho (o equivalente IPv6 da ARP) e para testar se um vizinho ainda está alcançável. A Figura 22.8 ilustra o formato de uma solicitação de vizinho.

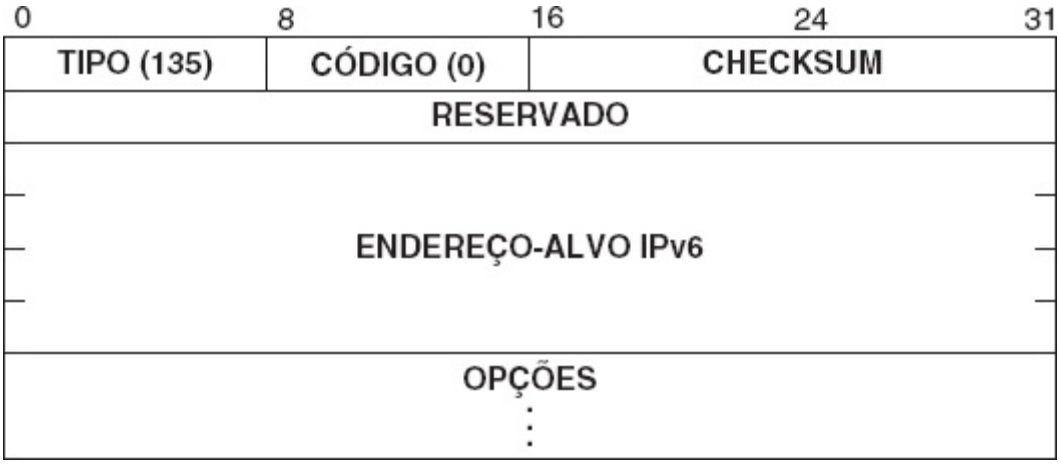


Figura 22.8 Formato de uma mensagem de Solicitação de Vizinho ICMPv6.

O campo ENDEREÇO -ALVO IPv6 fornece o endereço IP de um vizinho para o qual é necessário um endereço MAC. Se o remetente já tem um endereço de IP, o campo OPÇÕES inclui o endereço MAC do remetente para que o receptor saiba o vínculo de endereço IP para MAC do remetente.

22.22 Mensagem de anúncio de vizinho ICMPv6

Um nó envia uma mensagem de *Anúncio de Vizinho* (*Neighbor Advertisement*) em resposta a uma mensagem de *Solicitação de Vizinho* ou para propagar alcançabilidade. A Figura 22.9 ilustra o formato de um anúncio de vizinho.

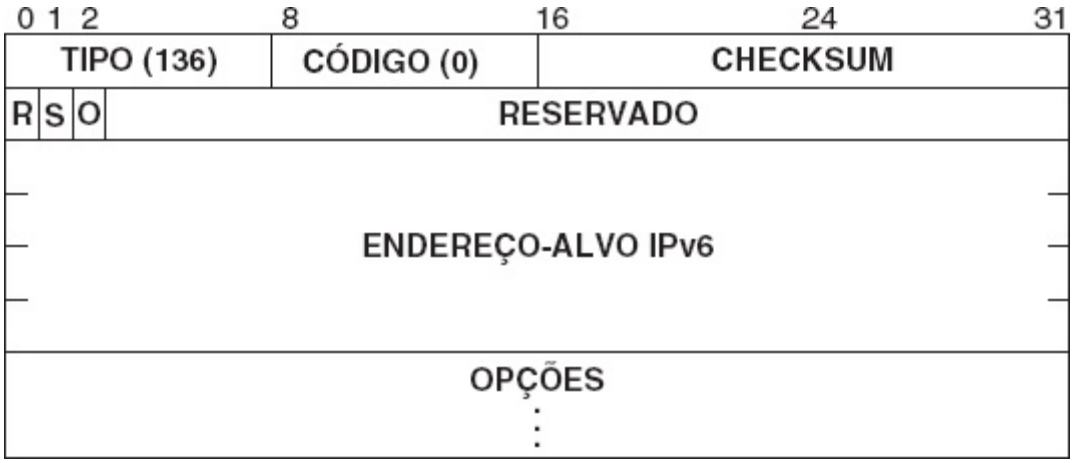


Figura 22.9 Formato de uma mensagem de Anúncio de Vizinho ICMPv6.

O bit *R* indica que o remetente é um roteador, o bit *S* indica que o

anúncio é uma resposta a uma mensagem solicitação de vizinho e o bit *0* indica que a informação na mensagem deve substituir qualquer informação que o receptor tenha previamente armazenada em cache. Apesar do nome, o campo ENDEREÇO-ALVO IPV6 (TARGET IPv6 ADDRESS) fornece o endereço IP do remetente (o remetente era o alvo da mensagem solicitação de vizinho que originou o anúncio). O campo de OPÇÕES dá o endereço MAC do remetente.

22.23 Mensagem redirecionar ICMPv6

Um roteador envia uma mensagem *Redirecionar* exatamente pela mesma razão que um roteador IPv4 envia um redirecionar ICMP: para solicitar que um host mude seu primeiro salto para um destino específico. A Figura 22.10 ilustra o formato de uma mensagem de redirecionar.

Como esperado, uma mensagem de redirecionar especifica dois endereços IPv6: um destino e o endereço de um primeiro salto a usar. Normalmente, uma mensagem redirecionar é solicitada quando um roteador recebe um datagrama de um host em um link diretamente conectado e descobre que o destino é alcançado através de outro roteador no mesmo link. Quando um host recebe um redirecionar, deve mudar sua tabela de encaminhamento para usar o ENDEREÇO DE PRIMEIRO SALTO IPV6 especificado para futuros datagramas enviados para o ENDEREÇO DE DESTINO IPV6.

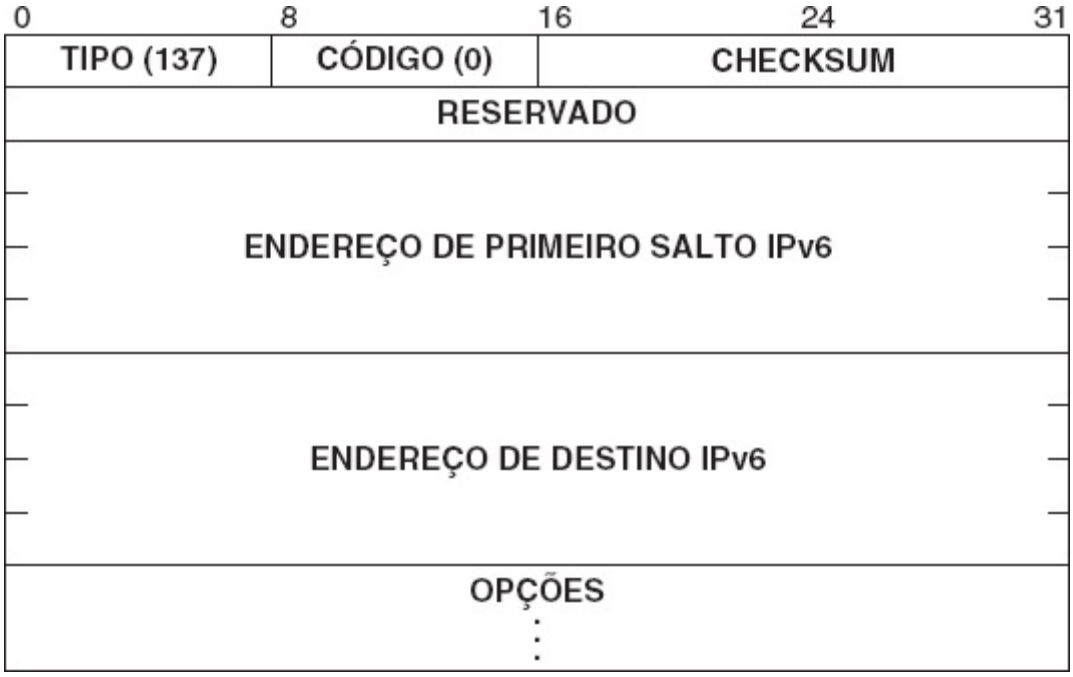


Figura 22.10 Formato de uma mensagem Redirecionar ICMPv6.

22.24 Resumo

O Protocolo de Configuração de Host Dinâmico (Dynamic Host Configuration Protocol – DHCP) permite que um computador obtenha informações na inicialização, incluindo um endereço IP, o endereço de um roteador-padrão e o endereço de um servidor de nome de domínio. O DHCP permite que um servidor aloque endereços IP automática ou dinamicamente. A alocação dinâmica é necessária para ambientes como uma rede sem fio, em que computadores podem se conectar e desconectar rapidamente.

Para usar o DHCP, um computador se torna um cliente. O computador difunde uma requisição para servidores DHCP, seleciona uma das ofertas que recebe e troca mensagens com o servidor a fim de obter um aluguel no endereço IP anunciado. Um agente de relay pode encaminhar requisições DHCP em nome do cliente, o que significa que um site pode ter um único servidor DHCP tratando atribuição de endereço para várias sub-redes.

Quando um cliente obtém um endereço IPv4, ele inicia três timers. Após o primeiro timer expirar, o cliente tenta renovar seu aluguel. Se um segundo timer expirar antes que a renovação se complete, o cliente tenta reassociar seu endereço através de qualquer servidor DHCP. Se o último timer expirar antes que o aluguel tenha sido renovado, o cliente para de usar o endereço IP e retorna para o estado inicial a fim de adquirir um novo endereço. Uma máquina de estado finito explica a aquisição e renovação de aluguel. Uma máquina de estado finito especifica a aquisição de locação e renovação.

Dizemos que DHCP fornece atribuição de endereço gerenciado; a alternativa é um sistema não gerenciado em que cada computador escolhe um endereço e verifica se este é único. O IPv6 oferece tanto atribuição gerenciada como não gerenciada. A abordagem IPv6 gerenciada usa o DHCPv6, e a não gerenciada usa endereço stateless autoconfiguration.

Stateless autoconfiguration é tratada pelo Protocolo de Descoberta de Vizinho IPv6 (NDP ou IPv6-ND), que também lida com a resolução de endereços e alcançabilidade de vizinho. NDP define cinco mensagens ICMPv6: duas para solicitação e anúncio de roteador, duas para solicitação e anúncio de vizinho e uma para redirecionar primeiro salto.

EXERCÍCIOS

22.1 O DHCP não contém um campo explícito para retornar a hora do dia do servidor ao cliente, mas torna isso parte das informações (opcionais) específicas do fornecedor. A hora deve ser incluída nos campos necessários? Sim ou não? Por quê?

22.2 Argumente que a separação entre a configuração e o armazenamento das imagens de memória *não* é desejável. (Veja a RFC 951 para obter dicas.)

22.3 O formato de mensagem é inconsistente porque possui dois campos para o endereço IP do cliente e um para o nome da imagem de inicialização. Se o cliente deixar seu campo de endereço IP vazio, o servidor retorna o seu endereço IP no segundo campo. Se o cliente deixar o campo de nome de arquivo de boot vazio, o servidor o *substitui* pelo nome explícito. Por quê?

22.4 Leia o padrão para descobrir como os clientes e servidores usam o campo HOPS em uma mensagem DHCP.

22.5 Quando um cliente DHCP recebe uma resposta através de broadcast de hardware, como ele sabe se a resposta é destinada a outro cliente DHCP na mesma rede física?

22.6 Quando uma máquina obtém sua máscara de sub-rede com o DHCP em vez do ICMP, ela coloca menos carga sobre *outros* computadores host. Explique.

22.7 Leia o padrão para descobrir como um cliente e um servidor DHCP podem concordar na duração de um aluguel sem ter relógios sincronizados.

22.8 Considere um host que possui um disco e usa DHCP para obter um endereço IP. Se o host armazenar seu endereço no disco juntamente com a data em que o aluguel expira e, depois, reiniciar dentro do período de aluguel, ele pode usar o endereço? Justifique sua resposta.

22.9 O DHCP exige um aluguel de endereço mínimo de uma hora. Você pode imaginar uma situação em que o aluguel mínimo do DHCP causa inconveniência? Explique.

22.10 Leia a RFC para descobrir como o DHCP especifica timers de renovação e reassociação. Um servidor deve, alguma vez, definir um sem o outro? Por quê?

22.11 O diagrama de transição de estado não mostra retransmissão. Leia o padrão para descobrir quantas vezes um cliente deve retransmitir uma requisição.

22.12 O DHCP pode garantir que um cliente não esteja "forjando" (ou seja, o DHCP pode garantir que não enviará informações de configuração sobre o host *A* para o host *B*)? A resposta em relação ao BOOTP é diferente?

22.13 O DHCP especifica que um cliente precisa estar preparado para tratar pelo menos *312* octetos de opções. De onde surgiu o número *312*?

22.14 Um computador que usa DHCP para obter um endereço IP pode operar um servidor? Em caso afirmativo, como um cliente pode alcançar o servidor?

22.15 Suponha que um computador IPv6 se conecte a uma rede que não tem nenhum roteador. Como o nó 1Pv6 sabe que ele deve usar stateless autoconfiguration para obter um endereço IPv6?

22.16 Estenda a pergunta anterior: se um nó IPv6 se conecta a uma rede que tem um roteador, como é que o nó sabe se usa stateless autoconfiguration?

22.17 Se um nó IPv6 usa stateless autoconfiguration, pode executar um servidor? Explique.

---

* Definir DHPC como uma extensão do BOOTP habilitou o DHCP a ser implantado sem substituir agentes de retransmissão BOOTP existentes.

** Como veremos, o termo "opções" é um tanto enganoso, pois o DHPC usa um campo de opções para levar a maior parte da informação de bootstrap.

* Embora o padrão permita que um cliente espere por respostas de vários servidores, a maioria das implementações aceita e processa a primeira resposta.

* Valores para o campo TIPO_H são atribuídos pela IETF.

* Como cada endereço IPv4 ocupa 4 octetos, um campo de N octetos guarda N / 4 endereços IPv4.

* O Capítulo23 considera em detalhes o Domain Name System.

** O termo *named* vem de *name daemon*.

* A comparação direta do RFC para DHCP e DHCPv6 é um pouco injusta, pois a especificação do DHCPv6 inclui algumas das opções.

CAPÍTULO 23

O Domain Name System (DNS)

CONTEÚDOS DO CAPÍTULO

23.25 Resumo

23.1 Introdução

Os protocolos descritos em capítulos anteriores usam valores binários chamados endereços *Internet Protocol* (endereços IP) para identificar hosts e roteadores. Embora esses endereços ofereçam uma representação conveniente e compacta para especificar a origem e o destino nos datagramas enviados por uma internet, os usuários preferem atribuir às máquinas nomes pronunciáveis, facilmente lembrados.

Este capítulo considera um esquema para atribuir nomes de domínio de alto nível significativos a um grande conjunto de máquinas, e discute um mecanismo que mapeia entre nomes de máquina de alto nível e endereços IP binários. Ele considera a tradução dos nomes de alto nível para endereços IP e a tradução de endereços IP para nomes de máquina de alto nível. O esquema de nomes é interessante por dois motivos. Primeiro, ele foi usado para atribuir nomes de máquina pela Internet. Segundo, por usar um conjunto de servidores geograficamente distribuídos para mapear nomes a endereços, a implementação do mecanismo de mapeamento de nome oferece um exemplo em grande escala do paradigma cliente-servidor descrito no Capítulo 20.

23.2 Nomes para computadores

Os sistemas de computador mais antigos forçavam os usuários a entenderem os endereços numéricos para objetos como tabelas do sistema e dispositivos periféricos. Os sistemas de tempo compartilhado avançaram a computação, permitindo que os usuários inventem nomes simbólicos significativos para objetos físicos (por exemplo, dispositivos periféricos) e objetos abstratos (por exemplo, arquivos). Um padrão semelhante surgiu nas redes de computador. Os primeiros sistemas admitiam conexões ponto a ponto entre computadores e usavam endereços de MAC de baixo nível para especificar os computadores. A interligação de redes introduziu o endereçamento universal e também o software de protocolo para mapear endereços universais em endereços de MAC de baixo nível. Uma vez que a Internet contém milhões de máquinas, os usuários precisam de nomes significativos, simbólicos, para especificar os computadores que eles usam.

Os primeiros nomes de computadores refletiam o ambiente pequeno em que eram escolhidos. Era bastante comum que um site com um punhado de máquinas escolhesse nomes com base nas finalidades delas. Por exemplo, as máquinas normalmente tinham nomes como *contabilidade*, *desenvolvimento* e *produção*. Os usuários preferem esses nomes a endereços de hardware complicados.

Embora a distinção entre *endereço* e *nome* seja intuitivamente atraente, ela é artificial. Qualquer *nome* é simplesmente um identificador que consiste em uma sequência de caracteres escolhida de um alfabeto finito. Os nomes só são úteis se o sistema puder mapeá-los eficientemente para o objeto que representam. Assim, pensamos em um endereço IP como um *nome de baixo nível* e dizemos que os usuários preferem *nomes de alto nível* para os computadores.

O formato dos nomes de alto nível é importante porque determina como os nomes são traduzidos para nomes de baixo nível ou vinculados a objetos, além de como as atribuições de nome são autorizadas. Com apenas algumas máquinas, a escolha de nomes de alto nível é fácil – cada administrador pode escolher um nome qualquer e verificar se este não está em uso no ambiente local. Por exemplo, quando seu computador departamental principal foi conectado à Internet, em 1980, o Computer Science Department, na Purdue University, escolheu o nome *purdue* para identificar a máquina conectada. Na época, na lista de conflitos em potencial, havia apenas algumas dezenas de nomes. Por volta de 1986, a lista oficial de host na Internet continha 3.100 nomes oficialmente

registrados e 6.500 apelidos (aliases). Embora a lista estivesse crescendo rapidamente na década de 1980, a maioria dos sites tinha máquinas adicionais (por exemplo, tipicamente computadores pessoais) que não estavam registrados. Na Internet atual, com centenas de milhões de máquinas, a escolha de nomes simbólicos é muito mais difícil.

23.3 Espaço de nomes plano

O conjunto original de nomes de máquina usados pela Internet formou um *espaço de nomes plano*, em que cada nome consistia em uma sequência de caracteres sem qualquer estrutura adicional. No esquema original, um site central, o *Network Information Center* (*NIC*), administrava o namespace e determinava se um novo nome era apropriado (ou seja, ele proibia nomes obscenos ou nomes novos que entravam em conflito com os existentes).

A principal vantagem de um espaço de nomes plano é que os nomes são convenientes e curtos; a principal desvantagem é que um espaço de nomes plano não pode generalizar para conjuntos grandes de máquinas por motivos técnicos e administrativos. Primeiro, como os nomes são tirados de um único conjunto de identificadores, o potencial para conflito aumenta à medida que o número de sites cresce. Segundo, como a autoridade para acrescentar novos nomes precisa estar em um único site, a carga de trabalho administrativa nesse site central também aumenta com o número de sites. Para entender a seriedade do problema, imagine uma autoridade central tentando lidar com a Internet de hoje, em que um novo computador aparece aproximadamente dez vezes por segundo. Terceiro, como os vínculos entre nome e endereço mudam com frequência, o custo de manter cópias corretas da lista inteira em cada site é alto e aumenta à medida que o número de sites cresce. Como alternativa, se o banco de dados de nomes residir em um único site, o tráfego da rede para esse site aumenta com o número de sites.

23.4 Nomes hierárquicos

Como um sistema de nomeação pode acomodar um conjunto de nomes grande e em expansão rápida sem exigir que um site central o administre? A resposta está na descentralização do mecanismo de nomes, delegando autoridade para partes do espaço de nomes e distribuindo a responsabilidade pelo mapeamento entre nomes e endereços. A Internet utiliza esse esquema. Antes de examinar os detalhes, vamos considerar a motivação e a intuição por trás disso.

O particionamento de um espaço de nomes precisa ser definido de modo

que admita o mapeamento eficiente destes e garanta o controle autônomo da atribuição de nomes. A otimização apenas para mapeamento eficiente pode levar a soluções que retêm um espaço de nomes plano e reduzem o tráfego dividindo os nomes entre várias máquinas de mapeamento. A otimização apenas por facilidade administrativa pode levar a soluções que tornam a delegação de autoridade fácil, mas o mapeamento de nomes dispendioso ou complexo.

Para entender como o espaço de nomes deve ser dividido, considere a estrutura interna de grandes organizações. No topo, um diretor executivo tem responsabilidade geral. Como o diretor não consegue supervisionar tudo, a organização pode ser particionada em divisões, com um executivo encarregado de cada uma. O diretor executivo concede a cada divisão autonomia dentro de limites especificados. Mais diretamente ao ponto, o executivo encarregado de determinada divisão pode contratar ou demitir funcionários, atribuir cargos e delegar autoridade, sem pedir autorização direta do diretor executivo.

Além de facilitar a delegação de autoridade, a hierarquia de uma grande organização apresenta operação autônoma. Por exemplo, quando um funcionário precisa de informações como o número de telefone de um novo empregado, ele começa perguntando aos funcionários administrativos locais (que podem contatar funcionários administrativos em outras divisões). O ponto é que, embora a autoridade sempre passe para baixo na hierarquia corporativa, a informação pode fluir pela hierarquia de um cargo para outro.

23.5 Delegação de autoridade para nomes

Um esquema de nomes hierárquico funciona como a administração de uma grande organização. O espaço de nomes é *particionado* no nível superior, e a autoridade para nomes em subdivisões é passada para agentes designados. Por exemplo, alguém poderia decidir particionar o espaço de nomes com base no *nome do site* e delegar a cada site responsabilidade por manter nomes dentro de sua partição. O nível superior da hierarquia divide o espaço de nomes e delega autoridade para cada divisão; ele não precisa se preocupar com mudanças dentro de uma divisão.

Em geral, a sintaxe dos nomes atribuídos hierarquicamente reflete a delegação hierárquica da autoridade usada para atribuí-los. Como exemplo, considere um espaço de nomes no formato:

local . site

onde o *site* é o nome do site autorizado pela autoridade central, *local* é a parte de um nome controlada pelo site e o caractere* de ponto é um delimitador usado para separá-los. Quando a autoridade superior aprova a inclusão de um novo site, *X*, ela acrescenta *X* à lista de sites válidos e delega ao site *X* autoridade para todos os nomes que terminam com "*.X*".

23.6 Autoridade de subconjunto

Em um espaço de nomes hierárquico, a autoridade pode ser subdividida ainda mais em cada nível. Em nosso exemplo de partição por sites, o próprio site pode consistir em vários grupos administrativos, e a autoridade dele pode escolher subdividir seu espaço de nomes entre os grupos. A ideia é manter a subdivisão do espaço de nomes até que cada subdivisão seja pequena o suficiente para ser administrável.

Sintaticamente, a subdivisão do espaço de nomes introduz outra partição do nome. Por exemplo, a inclusão de uma subdivisão de *grupo* para nomes já particionados por site produz a seguinte sintaxe de nome:

local . group . site

Como o nível mais alto delega autoridade, os nomes de grupo não precisam combinar entre todos os sites. Um site de universidade poderia escolher nomes de grupo como *engenharia*, *ciência* e *artes*, enquanto um site corporativo poderia escolher nomes de grupo como *produção*, *contabilidade* e *pessoal*.

O sistema telefônico dos Estados Unidos oferece outro exemplo de sintaxe de nomes hierárquica. Os dez dígitos de um número de telefone foram particionados em um *código de área* de três dígitos, uma *central* de três dígitos e um *número de assinante* de quatro dígitos dentro da central. Cada central tem autoridade para atribuir números de assinante dentro de sua parte do espaço de nomes. Embora seja possível agrupar quaisquer assinantes em centrais e agrupar quaisquer centrais em códigos de área, a atribuição de números de telefone não é caprichosa; os números são cuidadosamente escolhidos para facilitar o roteamento de chamadas pela rede telefônica.

O exemplo do telefone é importante porque ilustra uma distinção básica entre o esquema de nomes hierárquico usado em uma internet TCPIIP e outras hierarquias: o particionamento do conjunto de máquinas possuídas

por uma organização por linhas de autoridade não necessariamente implica o particionamento do local físico. Por exemplo, pode ser que, em alguma universidade, um único prédio acomode o departamento de matemática e também o departamento de ciência da computação. Pode acontecer até mesmo que, embora as máquinas desses dois grupos fiquem sob domínios administrativos completamente separados, elas se conectem à mesma rede física. Também pode acontecer que um único grupo possua máquinas em várias redes físicas. Por esses motivos, o esquema de nomes do TCPIIP permite a delegação arbitrária de autoridade para o espaço de nomes hierárquico sem considerar as conexões físicas. O conceito pode ser resumido da forma a seguir.

> *Na Internet, os nomes de máquina hierárquicos são atribuídos de acordo com a estrutura das organizações que obtêm autoridade para partes do espaço de nomes, não necessariamente de acordo com a estrutura das interconexões físicas da rede.*

Naturalmente, em muitos sites, a hierarquia organizacional corresponde à estrutura das interconexões físicas da rede. Por exemplo, suponha que os computadores em determinado departamento se conectem à mesma rede. Se o departamento também receber parte da hierarquia de nomes, todas as máquinas com nomes nessa parte da hierarquia também se conectarão a uma única rede física.

23.7 Nomes de domínios da Internet

O *Domain Name System* (*DNS*) é o sistema que fornece mapeamento de nome para endereço para a Internet. O DNS possui dois aspectos conceitualmente independentes. O primeiro é abstrato: ele especifica a sintaxe de nome e as regras para delegar autoridade sobre os nomes. O segundo é concreto: ele especifica a implementação de um sistema de computação distribuído que eficientemente mapeia nomes a endereços. Esta seção considera a sintaxe de nome, e outras seções examinam a implementação.

O Domain Name System utiliza um esquema de nomes hierárquico conhecido como *nomes de domínio*. Assim como em nossos exemplos anteriores, um nome de domínio consiste em uma sequência de subnomes separados por um caractere delimitador, o ponto (dot). Em nossos

exemplos, dissemos que as seções individuais do nome poderiam representar sites ou grupos, mas o sistema de nome de domínio simplesmente chama cada seção de *label*. Assim, o nome de domínio

cs . purdue . edu

contém três labels: *cs*, *purdue* e *edu*. Qualquer sufixo de um rótulo em um nome de domínio também é chamado de domínio. Nesse exemplo, o domínio de nível mais baixo é *cs.purdue.edu* (o nome de domínio para o Computer Science Department na Purdue University), o domínio de segundo nível é *purdue.edu* (o nome de domínio para a Purdue University), e o domínio de alto nível é *edu* (o nome de domínio para instituições educacionais). Como o exemplo mostra, os nomes de domínio são escritos com o rótulo local primeiro e o domínio superior por último. Conforme veremos, sua escrita na ordem possibilita a compactação de mensagens que contêm vários nomes de domínio.

23.8 Domínios de nível superior

A Figura 23.1 lista exemplos de nomes globais de domínio de nível superior em uso atualmente.

A *Internet Corporation for Assigned Names and Numbers* (*ICANN*), que atribui nomes, tem lutado com a questão de quantos domínios de nível superior são necessários e que nomes devem ser permitidos. O esquema de código de país de duas letras, que se pensava permanente, está sujeito a mudanças políticas. Por exemplo, quando a Alemanha foi reunificada, o domínio *dd* de nível superior que havia sido designado para a Alemanha Oriental ficou obsoleto. Um mecanismo de internacionalização foi inventado para permitir nomes em outros conjuntos de caracteres. Assim, a figura só dá um instantâneo que pode mudar à medida que novos nomes de nível superior são aprovados e tornam-se ativos.

Nome do domínio	Significado
aero	**Setor de transporte aéreo**
arpa	**Domínio de infraestrutura**
asia	**Domínio regional para a Ásia**
biz	**Negócios**
cat	**Língua catalã e cultura**
com	**Organização comercial**
coop	**Associações cooperativas**
edu	**Instituição educacional (4 anos)**
gov	**Governo**
info	**Informação**
int	**Organizações de tratado internacional**
jobs	**Gestão de recursos humanos**
mil	**Militares dos Estados Unidos**
museum	**Museus**
name	**Indivíduos**
net	**Principais centros de suporte de rede**
org	**Organizações diferentes das citadas**
pro	**Profissionais credenciados**
travel	**Indústria turística**
xxx	**Pornografia da Internet**
código país	**Cada país (esquema geográfico: br = Brasil; il = Israel; pt = Portugal)**

Figura 23.1 Os domínios de Internet de nível superior e seus significados. Embora os labels sejam mostrados em minúsculas, as comparações de nome de domínio não diferenciam maiúsculas de minúsculas, de modo que *COM* é equivalente a *com*.

Conceitualmente, os nomes de nível superior permitem duas hierarquias de nomes completamente diferentes: geográfica e organizacional. O esquema geográfico divide o universo de máquinas por país. As máquinas

nos Estados Unidos ficam sob o domínio de nível superior *us*; quando outro país deseja registrar máquinas no sistema de nome de domínio, a autoridade central atribui ao país um novo domínio de nível superior com o identificador de padrão internacional do país com duas letras como seu label. A autoridade para o domínio US escolheu dividi-lo em um domínio de segundo nível por estado. Por exemplo, o domínio para o estado de Virginia é:

va . us

Como alternativa à hierarquia geográfica, os domínios de alto nível também permitem que as organizações sejam agrupadas por tipo organizacional. Quando uma organização deseja participar do Domain Name System, ela escolhe como deseja ser registrada e requisita aprovação. Um *registrador de nome de domínio* revê a aplicação e atribui um subdomínio* à organização sob um dos domínios existentes de nível superior. O proprietário de determinado domínio de alto nível pode decidir o que permitir e como particionar o espaço de nomes ainda mais. Por exemplo, no Reino Unido, que possui o código de país de duas letras *uk*, as universidades e outras instituições acadêmicas são registradas sob o domínio *ac.uk*.

Um exemplo poderá ajudar a esclarecer o relacionamento entre a hierarquia de nomes e a autoridade para nomes. Uma máquina chamada *xinu*, no Computer Science Department da Purdue University, tem o nome de domínio oficial:

xinu . cs . purdue . edu

O nome da máquina foi aprovado e registrado pelo administrador da rede local no Computer Science Department. O administrador do departamento, anteriormente, obteve autoridade para o subdomínio *cs.purdue.edu* por uma autoridade de rede da universidade, que obteve permissão para administrar o subdomínio *purdue.edu* da autoridade da Internet. Esta retém controle do domínio *edu*, de modo que novas universidades só podem ser acrescentadas com sua permissão. De maneira semelhante, o gerente de rede da universidade, na Purdue University, retém autoridade para o subdomínio *purdue.edu*, de modo que novos domínios de terceiro nível só podem ser acrescentados com a permissão do

administrador.

A Figura 23.2 ilustra uma pequena parte da hierarquia de nome de domínio da Internet. Como mostra a figura, a IBM Corporation, uma organização comercial, registrou-se como *ibm.com*, a Purdue University registrou-se como *purdue.edu* e a National Science Foundation, uma agência do governo, registrou-se como *nsf.gov*. Por outro lado, a Corporation for National Research Initiatives decidiu registrar-se sob a hierarquia geográfica como *cnri.reston.va.us*.

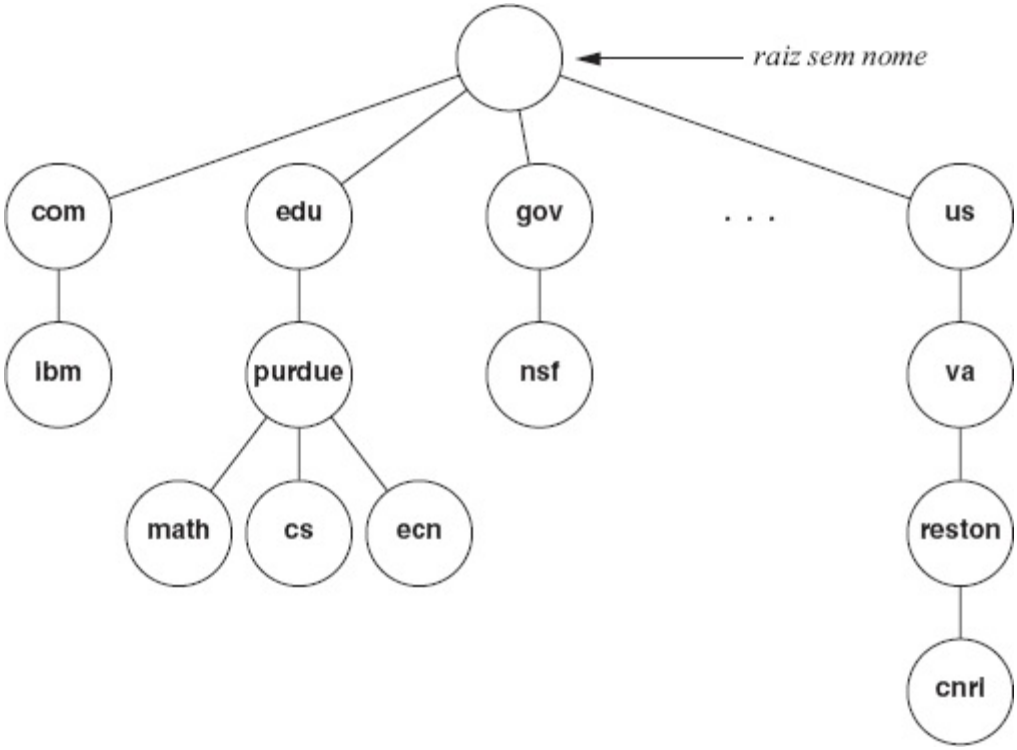


Figura 23.2 Uma pequena parte da hierarquia de nome de domínio (árvore) da Internet. Na prática, a árvore é larga e chata; a maioria das entradas de host aparece no quinto nível.

23.9 Sintaxe e tipo de nome

O sistema de nome de domínio é bastante genérico, pois permite que várias hierarquias de nomes sejam incorporadas. Além disso, no sistema pode haver vários tipos de mapeamento. Por exemplo, um determinado nome pode ser o nome de um computador host que tem um endereço IPv4, um computador host que tem um endereço IPv6, um servidor de e-mail e assim por diante. De modo curioso, a sintaxe do nome não indica o tipo.

Para permitir aos clientes distinguirem entre vários tipos de entradas,

cada item nomeado armazenado no sistema recebe um *tipo* que especifica se ele é o endereço de uma máquina, uma caixa de correio, um usuário e assim sucessivamente. Assim que um cliente pede ao sistema de domínio para traduzir um nome, ele precisa especificar o tipo de resposta desejada. Por exemplo, quando um aplicativo de correio eletrônico usa o sistema de domínio para traduzir um nome, ele especifica que a resposta deve ser o endereço de uma *central de correio*. Quando um browser resolve um nome de domínio para um site, deve especificar que procura o endereço IP do computador do servidor. Curiosamente, um determinado nome pode mapear vários itens. Ao resolver um nome, a resposta recebida depende do tipo especificado na consulta. Desse modo, se um usuário envia um e-mail para alguém em *x. com* e tipos de *x. com* em um browser, as duas ações podem resultar em contato de dois computadores completamente diferentes. Nós podemos, a seguir, resumir o ponto-chave.

Um determinado nome pode mapear mais de um item no sistema de domínio. O cliente especifica o tipo de objeto desejado ao traduzir um nome, e o servidor retorna objetos desse tipo.

Além de especificar o tipo de resposta procurada, o sistema de domínio permite que o cliente especifique a família de protocolo a usar. O sistema de domínio particiona o conjunto inteiro de nomes por *classe*, permitindo que um único banco de dados armazene mapeamentos para vários conjuntos de protocolos.*

A sintaxe de um nome não determina a classe de protocolo ou o tipo de objeto a que o nome se refere. Em particular, o número de labels em um nome não determina se este se refere a um objeto individual (máquina) ou um domínio. Assim, em nosso exemplo, é possível ter uma máquina

gwen . purdue . edu

embora

cs . purdue . edu

nomeie um subdomínio. Podemos resumir este ponto importante da maneira a seguir.

Não se podem distinguir os nomes dos subdomínios dos nomes dos objetos individuais ou do tipo de um objeto usando apenas a sintaxe do nome de domínio.

23.10 Mapeando nomes de domínio para endereços

Além das regras para a sintaxe de nome e delegação de autoridade, o esquema de nome de domínio inclui um sistema distribuído eficiente, confiável e de uso geral para mapear nomes a endereços. O sistema é distribuído no sentido técnico, significando que um conjunto de servidores operando em vários sites soluciona cooperativamente o problema de mapeamento. Ele é eficiente no sentido de que a maioria dos nomes pode ser mapeada localmente; somente alguns exigem tráfego de internet. Ele é de uso geral porque não está restrito a nomes de computadores (mas usaremos esse exemplo por enquanto). Finalmente, ele é confiável porque nenhuma falha de um único servidor impedirá que o sistema opere corretamente.

O mecanismo de domínio para o mapeamento de nomes a endereços consiste em sistemas independentes, cooperativos, chamados *servidores de nome*. Um servidor de nome é um programa servidor que fornece tradução de nome para endereço, mapeando nomes de domínio para endereços IP. Normalmente, o software servidor é executado em um dedicado processador, e a própria máquina é chamada servidor de nome. O software cliente, chamado *tradutor de nome*, utiliza um ou mais servidores de nome ao traduzir um nome.

O modo mais fácil de entender como os servidores de domínio funcionam é imaginá-los arrumados em uma estrutura de árvore que corresponde à hierarquia de nomeação, como ilustra a Figura 23.3. A raiz da árvore é um servidor que reconhece os domínios de alto nível e sabe qual servidor traduz cada um. Dado um nome para traduzir, a raiz pode escolher o servidor correto para esse nome. No próximo nível, um conjunto de servidores de nome oferece respostas para um domínio de alto nível (por exemplo, *edu*). Um servidor nesse nível sabe quais servidores podem traduzir cada um dos subdomínios sob seu domínio. No terceiro nível da árvore, os servidores de nomes oferecem respostas para subdomínios (por exemplo, *purdue* sob *edu*). A árvore conceitual continua com um servidor em cada nível para o qual um subdomínio foi definido.

Os links na árvore conceitual não indicam conexões físicas de rede. Em vez disso, eles mostram quais outros servidores de nomes determinado servidor conhece e contata. Os próprios servidores podem estar localizados em quaisquer locais em uma internet. Assim, a árvore de servidores é uma abstração que usa uma internet para comunicação.

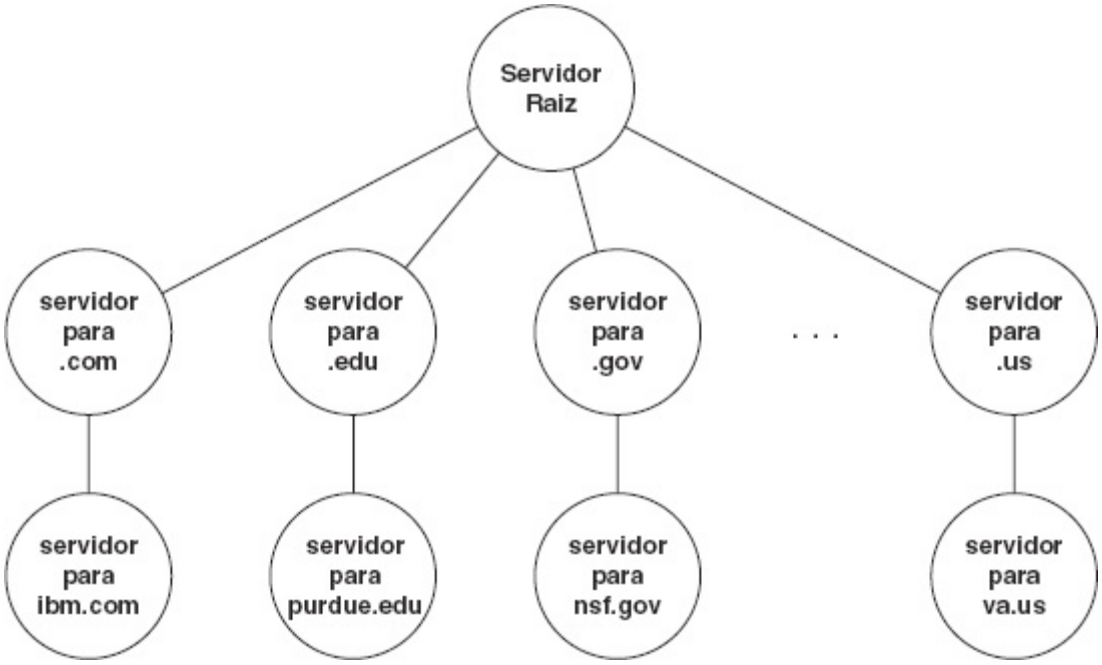


Figura 23.3 A organização conceitual dos servidores de nome de domínio em uma árvore que corresponde à hierarquia de nomes. Em teoria, cada servidor conhece os endereços de todos os servidores de nível inferior para todos os subdomínios dentro do domínio que ele trata.

Se os servidores no sistema de domínio funcionassem exatamente como nosso modelo simplista sugere, o relacionamento entre conectividade e autorização seria muito simples. Quando a autoridade fosse concedida a um subdomínio, a organização que a solicitou teria de estabelecer um servidor de nome de domínio para esse subdomínio e vinculá-lo à árvore.

Na prática, o relacionamento entre a hierarquia de nomes e a árvore de servidores não é tão simples quanto nosso modelo mostra. A árvore de servidores possui poucos níveis, pois um único servidor físico pode conter toda a informação para grandes partes da hierarquia de nomes. Em particular, as organizações normalmente coletam informações de todos os seus subdomínios para um único servidor. A Figura 23.4 mostra uma organização mais realista dos servidores para a hierarquia de nomes da Figura 23.2.

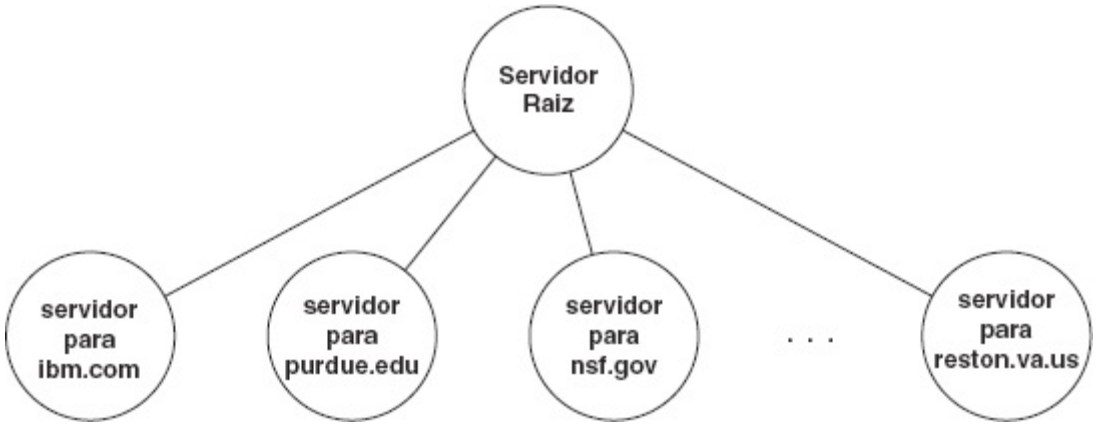


Figura 23.4 Uma organização realista de servidores para a hierarquia de nomes da Figura 23.2. Como a árvore é larga e chata, poucos servidores precisam ser contatados ao traduzir um nome.

Um servidor raiz contém informações sobre a raiz e domínios de alto nível, e cada organização utiliza um único servidor para seus nomes. Como a árvore de servidores é superficial, no máximo dois servidores precisam ser contatados para traduzir um nome como *xinu.cs.purdue.edu*: o servidor raiz e o servidor para o domínio *purdue.edu* (ou seja, o servidor raiz sabe qual servidor trata de *purdue.edu*, e a informação de domínio inteira para Purdue University reside em um servidor).

23.11 Tradução de nome de domínio

Embora a árvore conceitual facilite a compreensão do relacionamento entre os servidores, ela esconde diversos detalhes sutis. O exame do algoritmo de tradução de nomes ajudará a explicá-los. Conceitualmente, a tradução de nome de domínio prossegue de cima para baixo, começando com o servidor de nomes raiz e prosseguindo para servidores localizados nas folhas da árvore. Existem duas maneiras de usar o Domain Name System: contatando servidores de nomes um de cada vez ou pedindo ao sistema de servidor de nomes para realizar a tradução completa. De qualquer forma, o software cliente forma uma consulta de nome de domínio que contém o nome a ser traduzido, uma declaração da classe do nome, o tipo da resposta desejada e um código que especifica se o servidor de nomes deve traduzir o nome completamente. Ele envia a consulta a um servidor de nomes para tradução.

Quando um servidor de nomes de domínio recebe uma consulta, ele verifica se o nome se encontra no subdomínio para o qual é uma autoridade. Nesse caso, ele traduz o nome para um endereço, de acordo

com seu banco de dados, e anexa uma resposta à consulta antes de enviá-la de volta ao cliente. Se o servidor de nomes não puder traduzir o nome completamente, ele procura ver que tipo de interação o cliente especificou. Se o cliente solicitou tradução completa (*tradução recursiva*, na terminologia de nome de domínio), o servidor contata um servidor de nomes de domínio que pode traduzir o nome e retorna a resposta ao cliente. Se o cliente solicitou tradução não recursiva (*tradução iterativa*), o servidor de nomes não pode fornecer uma resposta. Ele gera uma resposta que especifica o servidor de nomes que o cliente deve contatar em seguida para traduzir o nome.

Como um cliente encontra um servidor de nomes em que começará a busca? Como um servidor de nomes encontra outros servidores de nomes que possam responder a perguntas quando ele não puder? As respostas são simples. Um cliente precisa saber como contatar pelo menos um servidor de nomes. Para garantir que um servidor de nome de domínio possa alcançar outros, o sistema de domínio exige que cada servidor saiba o endereço de pelo menos um servidor raiz.* Além disso, um servidor pode saber o endereço de outro para o domínio imediatamente acima dele (chamado de *pai*).

Os servidores de nome de domínio utilizam uma porta de protocolo bem conhecida para toda a comunicação, de modo que os clientes saibam como se comunicar com um servidor uma vez que conheçam o endereço IP da máquina em que o servidor é executado. Como um cliente sabe o endereço de um servidor de nomes? Muitos sistemas obtêm o endereço de um servidor de domínio automaticamente como parte do processo bootstrap.* Por exemplo, os protocolos de bootstrap, tais como o DHCP do IPv4 e IPv6 do NDP ou DHCPv6, podem fornecer um endereço de servidor de nomes. Naturalmente, outras abordagens são possíveis. Por exemplo, o endereço de um servidor de nomes pode ser vinculado a programas de aplicação em tempo de compilação. De modo alternativo, o endereço pode ser armazenado num arquivo no armazenamento secundário.

23.12 Tradução eficiente

Embora possa parecer natural resolver consultas descendo pela árvore de servidores de nomes, isso pode causar ineficiências por três motivos. Primeiro, como a maior parte da tradução de nome se refere a nomes locais, traçar um caminho através da hierarquia para contatar com a autoridade local seria ineficiente. Segundo, se cada tradução de nome sempre iniciasse

contatando o nível mais alto da hierarquia, a máquina nesse ponto ficaria sobrecarregada. Terceiro, a falha das máquinas nos níveis mais altos da hierarquia impediria a tradução de nomes, mesmo que a autoridade local pudesse traduzi-los. A hierarquia de nomes de telefone mencionada anteriormente ajuda a explicar. Embora os números de telefone sejam atribuídos hierarquicamente, eles são traduzidos de baixo para cima. Como a maioria das ligações telefônicas é local, elas podem ser traduzidas pela central local sem pesquisar a hierarquia. Além do mais, as chamadas dentro de determinado código de área podem ser traduzidas sem contatar sites fora do código de área. Quando aplicadas a nomes de domínio, essas ideias levam a um mecanismo de tradução de nome de duas etapas, que preserva a hierarquia administrativa, mas permite a tradução eficiente.

No processo de tradução de nomes em duas etapas, a tradução começa com o servidor de nomes local. Se o servidor local não puder traduzir um nome, a consulta precisa então ser enviada a outro servidor no sistema de domínio. A ideia-chave é descrita a seguir.

> *Um cliente sempre contata um servidor de nome de domínio local primeiro.*

23.13 Caching: a chave para a eficiência

Se os tradutores enviarem cada consulta ao servidor raiz, o custo de pesquisa para nomes não locais pode ser extremamente alto. Mesmo que as consultas pudessem ir diretamente para o servidor que tem autoridade para o nome, a pesquisa de nome pode apresentar uma carga intensa para a Internet. Assim, para melhorar o desempenho geral de um sistema servidor de nomes, é necessário reduzir o custo de pesquisa para nomes não locais.

Os servidores de nomes da Internet utilizam o caching para resolução eficiente. Cada servidor mantém um cache de respostas a pesquisas recentes, bem como um registro de onde a resposta foi obtida. Quando um cliente pede ao servidor para traduzir um nome, o servidor primeiro verifica se tem autoridade para o nome de acordo com o procedimento-padrão. Se não, o servidor verifica seu cache para ver se o nome foi traduzido recentemente. Os servidores dão informações do cache aos clientes, mas as marcam como um vínculo *não autorizado*, e dão o nome de domínio do servidor, *S*, do qual obtiveram o vínculo. O servidor local também envia informações adicionais que dizem ao cliente o vínculo entre

S e um endereço IP. Portanto, os clientes recebem respostas rapidamente, mas a informação pode estar desatualizada. Como a eficiência é importante, o cliente aceitará a resposta não autorizada e prosseguirá.

O caching funciona bem no Domain Name System porque os vínculos entre nome e endereço mudam com pouca frequência. No entanto, eles mudam. Se os servidores colocassem informações em cache na primeira vez que ela fosse solicitada e nunca a atualizassem, as entradas no cache ficariam *passadas* (ou seja, incorretas). Para manter o cache correto, os servidores só salvam as informações em cache enquanto elas são válidas – uma vez que um item se torna obsoleto, um servidor o remove do cache. Depois de uma entrada ser removido do seu cache, um servidor deve voltar para a origem autorizada e obter o vínculo para satisfazer os pedidos subsequentes.

A chave do sucesso DNS justifica-se porque um servidor não aplica um único tempo (timeout) limite fixo para todas as entradas. Isto é, sempre que uma autoridade responde a uma requisição, ela inclui um valor *Time To Live* (TTL) na resposta, que especifica por quanto tempo ele garante que o vínculo permanecerá válido. Assim, as autoridades podem reduzir o overhead de rede especificando timeouts longos para entradas que esperam permanecer inalteradas, enquanto especificam timeouts curtos para entradas que esperam mudar.

O caching é importante nos hosts e também em servidores de nome de domínio locais. A maioria dos softwares tradutores coloca entradas de DNS em cache no host. Assim, se um usuário pesquisar o mesmo nome repetidamente, pesquisas subsequentes podem ser traduzidas do cache local sem usar a rede.

23.14 Formato de mensagem do Domain Name System

O exame dos detalhes das mensagens trocadas entre clientes e os servidores de nome de domínio ajudará a esclarecer como o sistema opera do ponto de vista de um programa aplicativo típico. Consideramos que um usuário invoca um programa aplicativo e fornece o nome de uma máquina com a qual a aplicação precisa se comunicar. Antes que possa usar protocolos como TCP ou UDP para se comunicar com a máquina especificada, o programa aplicativo precisa encontrar o endereço IP da máquina. Ele passa o nome de domínio a um tradutor local e requisita um endereço IP. O tradutor local verifica seu cache e retorna a resposta, se houver uma presente. Se o tradutor local não tiver uma resposta, ele formatará uma

mensagem e a enviará para o servidor (ou seja, o local resolve tornar-se um cliente). Embora nosso exemplo só envolva um nome, o formato da mensagem permite que um cliente faça várias perguntas em uma única mensagem. Cada pergunta consiste em um nome de domínio para o qual o cliente busca um endereço IP, uma especificação da classe da consulta (ou seja, *internet*) e o tipo de objeto desejado (por exemplo, *endereço*). O servidor responde retornando uma mensagem semelhante, que contém respostas às perguntas para as quais o servidor possui vínculos. Se o servidor não puder responder a todas as perguntas, a resposta terá informações sobre outros servidores de nomes que o cliente pode contatar para obter as respostas.

As respostas também contêm informações sobre os servidores que são autoridades para as respostas e os endereços IP desses servidores. A Figura 23.5 mostra o formato da mensagem.

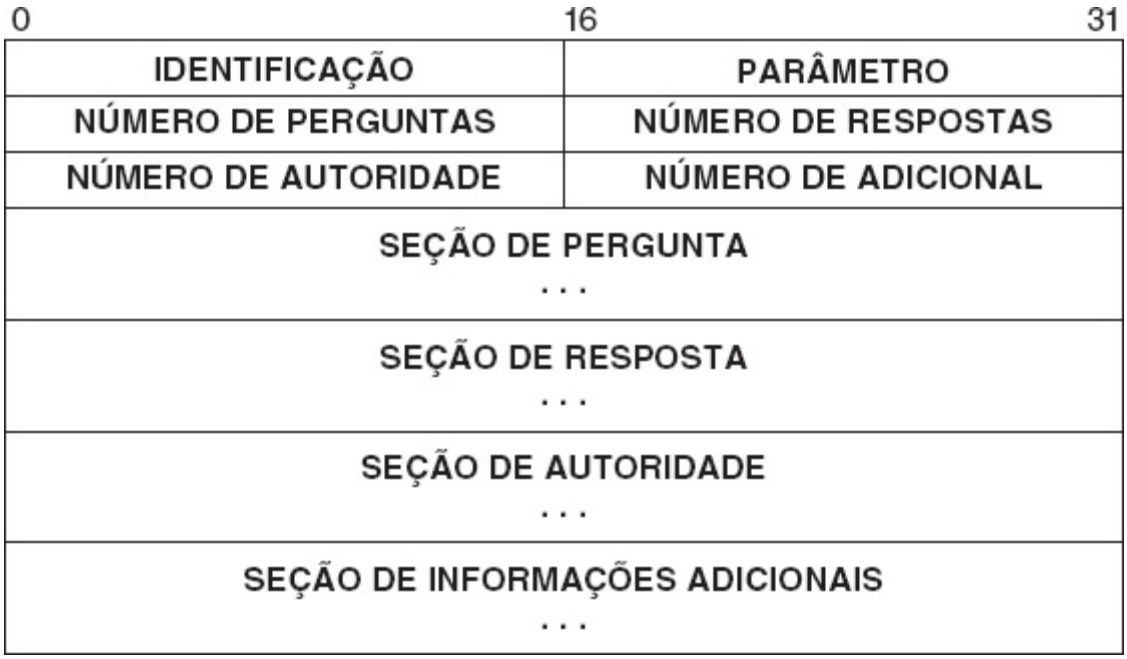


Figura 23.5 Formato de mensagem do servidor de nome de domínio. As seções de PERGUNTA, RESPOSTA, AUTORIDADE e INFORMAÇÕES ADICIONAIS são de tamanho variável.

Como a figura mostra, cada mensagem começa com um cabeçalho fixo. O cabeçalho contém um campo de IDENTIFICAÇÃO exclusivo, que o cliente utiliza para combinar respostas e perguntas, e um campo de PARÂMETRO que especifica a operação solicitada e um código de resposta. A Figura 23.6 indica a interpretação dos bits no campo PARÂMETRO.

Na Figura 23.5, os campos rotulados com NÚMERO DE dão uma contagem de entradas nas seções correspondentes que ocorrem mais adiante na mensagem. Por exemplo, o campo rotulado com NÚMERO DE PERGUNTAS indica a contagem de entradas que aparecem na SEÇÃO DE PERGUNTA da mensagem.

A SEÇÃO DE PERGUNTA contém consultas para as quais são desejadas respostas. O cliente preenche apenas a seção de pergunta; o servidor retorna as perguntas e respostas em sua resposta. Cada pergunta consiste em um NOME DE DOMÍNIO DE CONSULTA seguido pelos campos TIPO DE CONSULTA e CLASSE DE CONSULTA, como mostra a Figura 23.7.

Bit do campo PARÂMETRO	Significado
0	Operação: 0 Consulta 1 Resposta
1-4	Tipo de consulta: 0 Padrão 1 Reverso 2 Requisição de status do servidor 4 Notificar 5 Atualizar
5	Marcado se resposta autorizada
6	Marcado se mensagem truncada
7	Marcado se recursão desejada
8	Marcado se recursão disponível
9	Marcado se dados autenticados
10	Marcado se verificação desativada
11	Reservado
12-15	Tipo de resposta: 0 Sem erro 1 Erro de formato na consulta 2 Falha no servidor 3 Nome não existe 5 Recusado 6 Nome existe, quando não deveria 7 Conjunto RR existe 8 Conjunto RR que não deveria existir 9 Servidor não autorizado para a zona 10 Nome não contido na zona 0

Figura 23.6 O significado dos bits do campo PARÂMETRO em uma mensagem do servidor de nome de domínio. Os bits são numerados da esquerda para a direita, começando com 0.

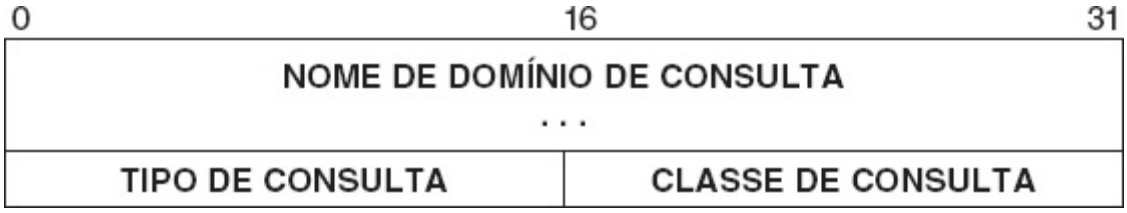


Figura 23.7 O formato das entradas na SEÇÃO DE PERGUNTA de uma mensagem do servidor de nome de domínio. O nome de domínio tem tamanho variável. Os clientes preenchem as perguntas, e os servidores as retornam junto com as respostas.

Embora o campo NOME DE DOMÍNIO DE CONSULTA tenha tamanho variável, veremos na próxima seção que a representação interna dos nomes de domínio torna possível para o receptor conhecer o tamanho exato. O TIPO DE CONSULTA codifica o tipo da consulta (por exemplo, se a pergunta se refere a um nome de máquina ou a um endereço de correio). O campo CLASSE DE CONSULTA permite que os nomes de domínio sejam usados para quaisquer objetos, pois os nomes oficiais da Internet são apenas uma classe possível. Deve-se observar que, embora o diagrama da Figura 23.5 siga nossa convenção de mostrar formatos em múltiplos de 32 bits, o campo NOME DE DOMÍNIO DE CONSULTA pode conter um número qualquer de octetos. Nenhum preenchimento é utilizado. Portanto, as mensagens de ou para servidores de nome de domínio podem conter um número estranho de octetos.

Em uma mensagem do servidor de nome de domínio, cada uma das seções de RESPOSTA, AUTORIDADE e INFORMAÇÕES ADICIONAIS consiste em um conjunto de registros de recurso que descreve os nomes de domínio e mapeamentos. Cada registro de recurso descreve um nome. A Figura 23.8 mostra o formato desse registro.

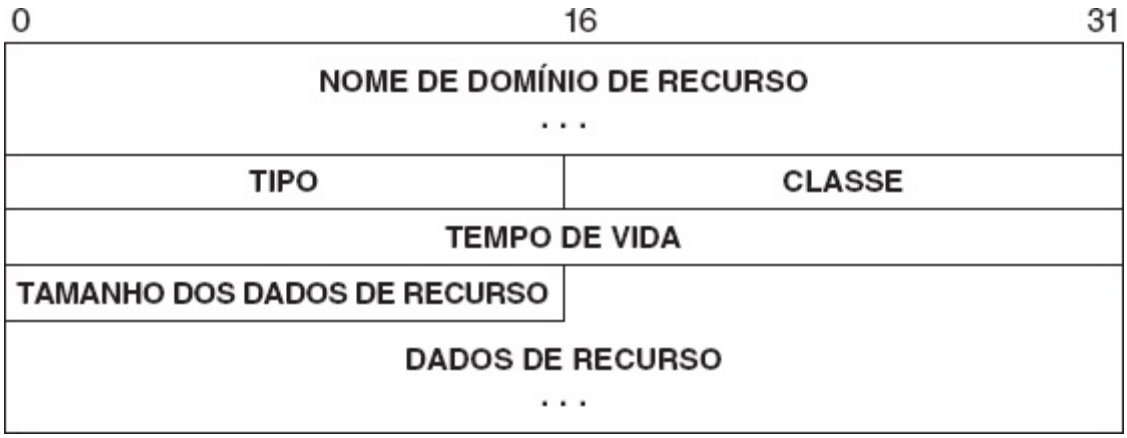


Figura 23.8 O formato dos registros de recurso usados em outras seções das

mensagens retornadas pelos servidores de nome de domínio.

O campo NOME DE DOMÍNIO DE RECURSO contém o nome de domínio ao qual esse registro de recurso se refere. Ele pode ter um tamanho qualquer. O campo TIPO especifica o tipo dos dados incluídos no registro de recurso; o campo CLASSE especifica a classe dos dados. O campo TEMPO DE VIDA contém um inteiro de 32 bits. O inteiro especifica o número de segundos em que a informação neste registro de recurso pode ser colocada em cache. Clientes usam o valor TEMPO DE VIDA para definir um timeout quando eles armazenam o registro do recurso. Os dois últimos campos contêm os resultados do vínculo, com o campo TAMANHO DOS DADOS DE RECURSO especificando a contagem de octetos no campo DADOS DE RECURSO.

23.15 Formato de nome compactado

Quando representados em uma mensagem, os nomes de domínio são armazenados como uma sequência de rótulos. Cada rótulo começa com um octeto que especifica seu tamanho. Assim, o receptor reconstrói um nome de domínio lendo repetidamente um tamanho de 1 octeto, n, e depois lendo um rótulo com n octetos de extensão. Um octeto de tamanho contendo zero marca o final do nome.

Os servidores de nome de domínio normalmente retornam várias respostas a uma consulta e, em muitos casos, os sufixos do domínio se sobrepõem. Para economizar espaço no pacote de resposta, os servidores de nomes compactam os nomes armazenando apenas uma cópia de cada nome de domínio. Ao extrair um nome de domínio de uma mensagem, o software cliente precisa verificar cada segmento do nome para ver se consiste em uma string literal (no formato de um contador de 1 octeto seguido pelos caracteres que compõem o nome) ou em um ponteiro para uma string literal. Ao encontrar um ponteiro, o cliente precisa segui-lo para encontrar o restante do nome.

Os ponteiros sempre ocorrem no início dos segmentos e são codificados no byte contador. Se os dois bits do topo do segmento de contador de segmento de 8 bits forem 1s, então o cliente precisa considerar os próximos 14 bits como um ponteiro inteiro. Se os dois bits do topo forem zero, os próximos 6 bits especificam o número de caracteres no rótulo.

23.16 Abreviação de nomes de domínio

A hierarquia do número de telefone ilustra outro recurso útil da tradução local, a *abreviação de nome*. A abreviação oferece um método para encurtar nomes quando o processo de tradução pode fornecer parte do nome de modo automático. Normalmente, um assinante omite o código de área quando liga para um número de telefone local. Os dígitos resultantes formam um nome abreviado, considerando que se encontra dentro do mesmo código de área do telefone do assinante. A abreviação também funciona bem para nomes de máquina. Dado um nome como *xyz*, o processo de tradução pode considerar que se encontra na mesma autoridade local da máquina em que está sendo traduzido. Assim, o tradutor pode fornecer automaticamente as partes que faltam do nome. Por exemplo, dentro do Computer Science Department, em Purdue, o nome abreviado

Xinu

é equivalente ao seguinte nome de domínio completo:

xinu . cs . purdue . edu

A maior parte do software cliente implementa abreviações com uma *lista de sufixo de nome de domínio*. O gerente de rede local configura uma lista de sufixos possíveis a serem anexados aos nomes durante a pesquisa. Quando um tradutor encontra um nome, ele percorre a lista, anexando cada sufixo e tentando pesquisar o nome resultante. Por exemplo, a lista de sufixo para o Computer Science Department, em Purdue, inclui:

. cs . purdue . edu
. purdue . edu
null

Assim, os tradutores locais primeiro anexam *cs.purdue.edu* ao nome *xinu*. Se a pesquisa falhar, eles anexam *purdue.edu* ao nome e tentam novamente. O último sufixo na lista de exemplo é a string *null*, significando que, se todas as outras pesquisas falharem, o tradutor tentará pesquisar o nome sem sufixo. Os gerentes podem usar a lista de sufixo para tornar a abreviação conveniente ou restringir os programas aplicativos a nomes locais.

Dissemos que o cliente pode ter responsabilidade pela expansão dessas abreviações, mas deve ser enfatizado que elas não fazem parte do próprio

Domain Name System. O sistema de domínio só permite a pesquisa de um nome de domínio totalmente especificado. Como consequência, os programas que dependem de abreviações podem não funcionar corretamente fora do ambiente em que foram montados. Podemos fazer o resumo a seguir.

> *O Domain Name System só mapeia nomes de domínio completos em endereços; as abreviações não fazem parte do próprio DNS, mas são introduzidas pelo software cliente para tornar os nomes locais convenientes para os usuários.*

23.17 Mapeamentos reversos

Dissemos que o Domain Name System pode oferecer mapeamentos diferentes do nome de computador ao endereço IP. As *consultas reversas* (inverse queries) permitem que o cliente peça a um servidor para mapear na direção oposta, apanhando uma resposta e gerando a pergunta que produziria essa resposta. Naturalmente, nem todas as respostas possuem uma pergunta exclusiva. Mesmo quando possuírem, um servidor pode não ser capaz de fornecê-la. Embora as consultas reversas tenham feito parte do sistema de domínio desde que foi especificado inicialmente, elas geralmente não são usadas, pois normalmente não há como descobrir o servidor que pode traduzir a consulta sem pesquisar o conjunto inteiro de servidores.

23.18 Consultas de ponteiro

Uma forma de mapeamento reverso é um mecanismo de autenticação que um servidor utiliza para verificar se um cliente está autorizado a acessar o serviço: o servidor mapeia o endereço IP do cliente a um nome de domínio. Por exemplo, um servidor na corporação *exemplo.com* poderia ser configurado para oferecer somente o serviço aos clientes da mesma corporação. Quando um cliente contata o servidor, este mapeia o endereço IP do cliente a um nome de domínio equivalente e verifica se o nome termina com *exemplo.com* antes de conceder acesso. A pesquisa reversa é tão importante que o sistema de domínio admite um domínio especial e uma forma especial de pergunta, chamada *consulta de ponteiro* (*pointer*

query), para fornecer o serviço. Em uma consulta de ponteiro, a pergunta apresentada a um servidor de nome de domínio especifica um endereço IP codificado como uma string imprimível na forma de um nome de domínio (ou seja, uma representação textual dos dígitos separados por pontos). Uma consulta de ponteiro requisita que o servidor de nomes retorne o nome de domínio correto para a máquina com o endereço IP especificado.

As consultas de ponteiro não são difíceis de gerar. Considere o IPv4. Quando pensamos em um endereço IPv4 escrito no formato decimal pontuado, ele tem o seguinte formato:

aaa . bbb . ccc . ddd

Para formar uma consulta de ponteiro, o cliente reorganiza a representação decimal pontuada do endereço para uma string no formato:

ddd . ccc . bbb . aaa . in-addr . arpa

O novo formato é um nome no domínio especial *in-addr.arpa.**

O IPv6 é mais complexo e resulta em nomes muito longos. Para formar uma consulta de ponteiro, um cliente representa o endereço IPv6 como uma série de nibbles (isto é, quantidades de 4 bites), escreve cada nibble em hexadecimal, inverte a ordem e acrescenta *ip6.arpa*. Por exemplo, o endereço IPv6

2001 : 18e8 : 0808 : 0000 : 0000 : 00d0 : b75d : 19f9

é representado como:

9 . f . 9 . 1 . d . 5 . 7 . b . 0 . d . 0 . 0 . 0 . 0 . 0 . 0 . 0 . 0 . 0 . 0 . 8 . 0 . 8 . 0 . 8 . e . 8 . 1 . 1 . 0 . 0 . 2 . ip6 . arpa

Como o servidor de nome local geralmente não é a autoridade para os domínios *arpa*, *in-addr.arpa*, ou *ip6.arpa*, deverá entrar em contato com outros servidores do mesmo tipo para completar a resolução. Para tornar a tradução de consultas de ponteiro eficiente, os servidores de domínio raiz da Internet mantêm um banco de dados de endereços IP válidos, junto com informações sobre servidores de nome de domínio que podem traduzir cada grupo de endereços.

23.19 Tipos de objeto e conteúdo de registro de recurso

Dissemos que o Domain Name System pode ser usado para traduzir um

nome de domínio para um endereço de central de correio, além de traduzir um nome de host para um endereço IP. O sistema de domínio é muito genérico porque pode ser usado para quaisquer nomes hierárquicos. Por exemplo, alguém poderia decidir armazenar os nomes dos serviços de computação disponíveis junto com um mapeamento de cada nome ao número de telefone a chamar, para descobrir o serviço correspondente. Ou então alguém poderia armazenar nomes de produtos de protocolo junto com um mapeamento com os nomes e endereços de fornecedores que oferecem esses produtos.

Lembre-se de que o sistema acomoda uma série de mapeamentos incluindo um *tipo* em cada registro de recurso. Ao enviar uma solicitação, um cliente precisa especificar o tipo de sua consulta;** os servidores especificam o tipo de dados em todos os registros de recursos que retornam. O tipo determina o conteúdo do registro de recurso de acordo com a tabela na Figura 23.9.

Tipo	Significado	Conteúdo
A	**IPv4 Host Adress**	**Endereço IPv4 de 32 bits**
AAAA	**IPv6 Host Adress**	**Endereço IPv6 de 128 bits**
CNAME	**Canonical Name**	**Nome de domínio canônico para um alias**
HINFO	**CPU & OS**	**Nome da CPU e sistema operacional**
MINFO	**Mailbox info**	**Informação sobre uma caixa de correio ou lista de correio**
MX	**Mail Exchanger**	**Preferência de 16 bits e nome do host que atua como central de correio para o domínio**
NS	**Name Server**	**Nome do servidor autorizado para o domínio**
PTR	**Pointer**	**Nome de domínio (como um link simbólico)**
SOA	**Start of Authority**	**Múltiplos campos que especificam quais partes da hierarquia de nomes um servidor implementa**

Figura 23.9 Alguns exemplos de tipos de registro de recurso do sistema de nome de domínio. Mais de cinquenta tipos foram definidos.

A maioria dos dados é do tipo *A* ou *AAA*, significando que consistem no nome de um host conectado à Internet, junto com o endereço IP do host. O segundo tipo de domínio mais útil, *MX*, é atribuído a nomes usados para centrais de e-mail. Ele permite que um site especifique vários hosts que são capazes de aceitar correio. Ao enviar e-mail, o usuário especifica um endereço na forma *usuário@parte-domínio*. O sistema de correio utiliza o sistema de nome de domínio para traduzir parte-domínio pelo tipo de consulta *MX*. O sistema de domínio retorna um conjunto de registros de recurso que contém um campo de preferência e um nome de domínio do host. O sistema de correio percorre o conjunto de maior preferência para o menor (números menores significam preferência maior). Para cada registro de recurso *MX*, a central extrai o nome de domínio e usa uma consulta tipo *A* ou *AAA* para traduzir esse nome para um endereço IP. Depois, ele tenta contatar o host e entregar o correio. Se o host não estiver disponível, a central continuará experimentando outros hosts na lista.

Para tornar a pesquisa eficiente, um servidor retorna vínculos adicionais que ele conhece na SEÇÃO DE INFORMAÇÕES ADICIONAIS de uma resposta. No caso dos registros *MX*, um servidor de domínio pode usar a SEÇÃO DE INFORMAÇÕES ADICIONAIS a fim de retornar os registros de recurso tipo *A* ou *AAA* para nomes de domínio informados na SEÇÃO DE RESPOSTA. Isso reduz substancialmente o número de consultas que uma central envia ao seu servidor de domínio.

23.20 Obtendo autoridade para um subdomínio

Antes que uma instituição receba autoridade para um domínio oficial de segundo nível, ela precisa combinar em operar um servidor de nome de domínio que atenda aos padrões da Internet. Naturalmente, um servidor de nome de domínio precisa obedecer aos padrões de protocolo que especificam formatos de mensagem e as regras para responder às requisições. O servidor também precisa saber os endereços dos servidores que tratam de cada subdomínio (se houver algum), além do endereço de pelo menos um servidor raiz. Na Internet atual, não é necessário que cada empresa não opere seu próprio servidor de nome. Em vez disso, existem empresas que, por uma taxa anual, executam servidores de nome de domínio em nome dos outros. Na verdade, essas empresas competem no negócio: elas oferecem uma variedade de serviços relacionados, tais como a

verificação de que um nome de domínio está disponível, registrando o nome com os registros regionais e registrando mapeamentos reversos por meio de consultas de ponteiro.

23.21 Aplicação do servidor e replicação

Na prática, o sistema de domínio é muito mais complexo do que descrevemos. Na maioria dos casos, um único servidor físico pode lidar com mais de uma parte da hierarquia de nomes. Por exemplo, um único servidor de nomes na Purdue University trata do domínio de segundo nível *purdue.edu* e também do domínio geográfico *laf.in.us*. Uma subárvore de nomes controlada por determinado servidor de nomes forma uma *zona de autoridade*, e os protocolos oferecem *zona de download* onde um cliente pode obter uma cópia de todo o conjunto de nomes e registros de recursos de um servidor. Outra complicação prática surge porque os servidores precisam ser capazes de lidar com muitas requisições, embora algumas delas levem muito tempo para resolver. Normalmente, os servidores admitem atividade simultânea, permitindo que o trabalho prossiga em outras requisições, enquanto as anteriores estão sendo processadas. O tratamento de requisições de forma simultânea é especialmente importante quando o servidor recebe uma requisição recursiva que o força a enviar a requisição a outro servidor para tradução.

A implementação do servidor também é complicada porque a autoridade da Internet exige que a informação em cada servidor de nome de domínio seja replicada. A informação precisa aparecer em pelo menos dois servidores que não operem no mesmo computador. Na prática, os requisitos são bastante rigorosos: os servidores não podem ter um ponto de falha comum. Evitar pontos de falha comuns significa que os dois servidores de nome não podem se conectar à mesma rede; eles nem sequer podem obter energia elétrica da mesma origem. Assim, para atender aos requisitos, um site precisa encontrar pelo menos um outro site que concorde em operar um servidor de nomes de backup. Naturalmente, a qualquer ponto na árvore de servidores, um servidor precisa saber como localizar os servidores de nome principal e de backup para subdomínios, e precisa direcionar consultas para um servidor de nomes de backup se o servidor primário estiver indisponível.

23.22 Atualização e notificação dinâmicas do DNS

Nossas discussões sobre NAT no Capítulo 19 e DHCP no Capítulo 22

mencionam a necessidade de interação com o DNS. No caso de uma caixa NAT que obtém um endereço dinâmico de um ISP, um servidor só pode ser colocado atrás da caixa NAT se o servidor de nome de domínio e o sistema NAT forem coordenados. No caso do DHCP, quando um host obtém um endereço dinâmico, o servidor DNS para o host precisa ser atualizado com o endereço atual do host. Para lidar com as situações descritas aqui e permitir que várias partes compartilhem a administração (por exemplo, permitir que vários registradores administrem em conjunto um domínio de alto nível), o IETF desenvolveu uma tecnologia conhecida como *Dynamic DNS*.

Existem dois aspectos do Dynamic DNS: *atualização* (update) e *notificação*. Como o nome sugere, a atualização do Dynamic DNS permite que mudanças sejam feitas dinamicamente na informação que um servidor DNS armazena. Assim, ao atribuir um endereço IP a um host, um servidor DHCP pode usar o mecanismo de atualização dinâmica para informar ao servidor DNS a atribuição. As mensagens de notificação solucionam o problema de propagar mudanças. Observe que, como o DNS utiliza servidores de backup, as mudanças feitas no servidor principal precisam ser propagadas a cada backup. Quando ocorre uma mudança dinâmica, o servidor principal envia uma notificação aos servidores de backup, que permite que cada backup solicite uma atualização da informação de zona. Por evitar o envio de cópias desnecessariamente, a notificação ocupa menos largura de banda do que simplesmente usar um pequeno timeout para as atualizações.

23.23 DNS Security Extensions (DNSSEC)

Por estar entre os aspectos mais importantes da infraestrutura da Internet, o Domain Name System normalmente é citado como um mecanismo crítico que deve ser protegido. Em particular, se um host estiver dando respostas incorretas às consultas de DNS, o software de aplicação ou os usuários podem ser enganados a acreditar em sites impostores ou revelar informações confidenciais. Para ajudar a proteger o DNS, o IETF desenvolveu uma tecnologia conhecida como *DNS Security* (*DNSSEC*).

Os principais serviços fornecidos pelo DNSSEC incluem *autenticação* da origem da mensagem e *integridade* dos dados. Ou seja, quando usa o DNSSEC, um host pode verificar se uma mensagem de DNS na realidade é originada em um servidor de DNS autorizado (ou seja, o servidor

responsável pelo nome na consulta) e se os dados na mensagem chegaram sem ser alterados. Além do mais, o DNSSEC pode autenticar respostas negativas – um host pode obter uma mensagem autenticada que indica que um nome de domínio em particular não existe.

Apesar de oferecer autenticação e integridade, o DNSSEC não soluciona todos os problemas. Em particular, o DNSSEC não oferece confidencialidade, nem delimita ataques de negação de serviço. O primeiro significa que, mesmo que um host use DNSSEC, um observador externo arejando uma rede será capaz de saber quais nomes o host pesquisa (ou seja, o observador pode ser capaz de descobrir por que determinada empresa está sendo contatada). A incapacidade de delimitar ataques de negação de serviço significa que, mesmo que um host e um servidor usem DNSSEC, não existe garantia de que as mensagens enviadas entre eles serão recebidas.

Para oferecer autenticação e integridade de dados, o DNSSEC utiliza um mecanismo de assinatura digital – além da informação solicitada, uma resposta de um servidor DNSSEC contém uma assinatura digital que permite que o receptor verifique se o conteúdo da mensagem não foi alterado. Um dos aspectos mais interessantes do DNSSEC surge da maneira como o mecanismo de assinatura digital é administrado. Assim como muitos mecanismos de segurança, o mecanismo DNSSEC utiliza *tecnologia de criptografia por chave pública*. A jogada interessante é que, para distribuir chaves públicas, o DNSSEC utiliza o Domain Name System. Isto é, um novo tipo foi definido que permite a um nome mapear uma chave pública. Cada servidor contém as chaves públicas para zonas mais abaixo na hierarquia (por exemplo, o servidor para *.com* contém a chave pública para exemplo *.com*). A fim de garantir a segurança para o sistema inteiro, a chave pública para o nível superior da hierarquia (ou seja, a chave para um servidor raiz) precisa ser configurada manualmente para um tradutor.

23.24 Multicast DNS e serviço de descoberta

Como o Domain Name System permite que tipos de registros arbitrários sejam adicionados, diversos grupos criaram nomes para outros objetos que não os computadores. Um uso em particular se destaca. Conhecido como *DNS multicast* (*mDNS*), o serviço é destinado a redes que não possuem servidores DNS específicos. Por exemplo, considere um par de smartphones que têm interfaces Wi-Fi.

Em vez de usar um servidor DNS, o mDNS usa IP multicast. Suponha que o host *A* precise saber o endereço IP do host *B*. O host *A* faz sua requisição por multicast. Todos os hosts que participam no mDNS recebem o multicast, e o host B envia resposta por multicast. Além disso, um host que participa do mDNS armazena em caches respostas mDNS, o que significa que o vínculo pode ser satisfeito a partir do cache.

Além de resolução de nomes de domínio, o mDNS foi estendido para lidar com *DNS Service Discovery* (*DNS-SD*). A ideia básica é simples: criar nomes de serviços na hierarquia DNS utilizando o sufixo *.local* e usar mDNS para procurar o nome. Desse modo, um smartphone pode usar DNS-SD para descobrir outros telefones celulares na área, que estão dispostos a participar de um determinado aplicativo. Os telefones só precisam concordar acerca de um nome para o serviço.

A principal desvantagem do uso de mDNS para descoberta de serviços decorre do tráfego gerado. Em vez de dois smartphones, imagine um conjunto de *N* smartphones. Cada um se anuncia como oferecendo um serviço e, em seguida, aguarda para ser sincronizado com uma variedade de aplicações. Agora imagine a situação em que *N* seja amplo e os telefones estejam usando uma rede sem fio aberta flat (ou seja, não roteada) como um hotspot Wi-Fi em um café de uma rua movimentada em uma cidade. Cada telefone que se conecta envia um multicast para os serviços que ele oferece, e outros respondem se conectando. A menos que *N* seja pequeno, o tráfego pode dominar a rede.

23.25 Resumo

Os sistemas de nomes hierárquicos permitem a delegação de autoridade para nomes, possibilitando acomodar um grande conjunto de nomes sem sobrecarregar um site central com tarefas administrativas. Embora a tradução de nome seja separada da delegação de autoridade, é possível criar sistemas de nomes hierárquicos em que a tradução seja um processo eficiente, que inicie no servidor local, ainda que a delegação de autoridade sempre flua do alto da hierarquia para baixo.

Examinamos o Domain Name System (DNS) da Internet e vimos que ele fornece um esquema de nomes hierárquico. O DNS usa a pesquisa distribuída, em que os servidores de nome de domínio mapeiam cada nome de domínio para um endereço IP ou endereço de central de correio. Os clientes começam tentando traduzir nomes localmente. Quando o servidor

local não puder traduzir o nome, o cliente precisa decidir trabalhar na árvore de servidores de nomes iterativamente ou requisitar o servidor de nomes local para fazê-lo recursivamente. Finalmente, vimos que o sistema de nomes de domínio admite uma série de vínculos, incluindo os de endereços IPv4 ou IPv6 para nomes de alto nível.

O DNSSEC oferece um mecanismo que pode ser usado para proteger o DNS; ele autentica respostas e garante a integridade delas. Além disso, utiliza criptografia por chave pública e usa o DNS para distribuir o conjunto de chaves públicas.

O multicast DNS (mDNS) permite que dois hosts em uma rede isolada obtenham o endereço IP de hosts na rede, sem depender de um servidor DNS. Uma extensão para mDNS, DNS-SD, proporciona a descoberta de serviços em geral. Um smartphone pode usar DNS-SD para descobrir outros smartphones nos arredores que estão dispostos a participar de um determinado serviço de aplicativo. A principal desvantagem de mDNS e DNS-SD surge do tráfego gerado quando uma rede contém muitos nós.

EXERCÍCIOS

23.1 O nome de um computador não deve ser vinculado ao sistema operacional no momento da compilação. Explique por quê.

23.2 Você preferiria usar um computador que obtivesse seu nome de um arquivo remoto ou de um servidor de nomes? Por quê?

23.3 Por que cada servidor de nomes conhece o endereço IP de seu pai em vez do nome de domínio de seu pai?

23.4 Crie um esquema de nomes que tolere mudanças na hierarquia de nomes. Como exemplo, considere duas empresas grandes que possuam uma hierarquia de nomes independente, e suponha que as empresas se fundam. Você consegue fazer com que todos os nomes anteriores ainda funcionem corretamente?

23.5 Leia o padrão e descubra como o Domain Name System utiliza registros *SOA*. Qual é a motivação para *SOA*?

23.6 O Domain Name System também pode acomodar nomes de caixa de correio. Descubra como.

23.7 O padrão sugere que, quando um programa precisa encontrar o nome de domínio associado a um endereço IP, ele deve enviar uma consulta reversa para o servidor local primeiro e usar o domínio *in-addr.arpa* ou *ip6.arpa* somente se isso falhar. Por quê?

23.8 Como você acomodaria abreviações em um esquema de nome de domínio? Como exemplo, mostre dois sites que sejam registrados sob *.edu* e um servidor de nível superior. Explique como cada site trataria cada tipo de abreviação.

23.9 Obtenha a descrição oficial do Domain Name System e monte um programa cliente. Pesquise o nome *xinu.cs.purdue.edu*.

23.10 Estenda o exercício anterior para incluir uma consulta de ponteiro. Tente pesquisar o nome de domínio para o endereço *128.10.19.20*.

23.11 Encontre uma cópia do programa *dig* e use-a para pesquisar os nomes nos dois exercícios anteriores.

23.12 Se estendermos a sintaxe do nome de domínio para incluir um ponto após o domínio de nível superior, os nomes e abreviações seriam não ambíguos. Quais são as vantagens e desvantagens da extensão?

23.13 Leia as RFCs no Domain Name System. Quais são os valores máximo e mínimo possíveis que um servidor de DNS pode armazenar no campo TEMPO DE VIDA de um registro de recurso? Qual a motivação para as escolhas?

23.14 O Domain Name System deverá permitir consultas de combinação parcial (ou seja, um curinga como parte de um nome)? Sim ou não? Por quê?

23.15 O Computer Science Department da Purdue University decidiu colocar a seguinte entrada de registro de recurso tipo *A* em seu servidor de nome de domínio:

```
localhost.cs.purdue.edu 127.0.0.1
```

Explique o que acontecerá se um site remoto tentar pingar uma máquina com nome de domínio *localhost.cs.purdue.edu*.

---

* Em nomes de domínio, o ponto delimitador é pronunciado “dot”.

* O padrão não define o termo “subdomínio”. Decidimos usá-lo porque sua analogia com “subconjunto” ajuda a esclarecer o relacionamento entre os domínios.

* Na prática, poucos servidores de domínio utilizam vários conjuntos de protocolos.

* Por confiabilidade, existem vários servidores para cada nó na árvore do servidor

de domínio; o servidor raiz é replicado ainda mais para fornecer balanceamento de carga.

* Veja o Capítulo 22 para uma discussão do protocolo bootstrapping.

* Os octetos do endereço IP precisam ser revertidos ao formar um nome de domínio, pois os endereços IP possuem os octetos mais significativos primeiro, enquanto os nomes de domínio têm os octetos menos significativos primeiro.

** As consultas podem especificar alguns tipos adicionais (por exemplo, existe um tipo de consulta que requisita todos os registros de recurso).

CAPÍTULO 24

Correio eletrônico (SMTP, POP, IMAP, MIME)

CONTEÚDOS DO CAPÍTULO

24.1 Introdução

Este capítulo continua nossa exploração da interligação de redes, considerando o serviço de correio eletrônico e os protocolos que suportam a transferência de correio e o acesso. O capítulo descreve como um sistema de correio é organizado, explica como o software do sistema de correio utiliza o paradigma cliente-servidor para transferir cada mensagem e descreve a representação da mensagem. Veremos, assim, que o e-mail ilustra várias ideias-chave no design do protocolo de aplicação.

24.2 Correio eletrônico

O sistema de *correio eletrônico* (*e-mail*) permite aos usuários transferirem memorando pela Internet. O e-mail é um sistema aplicativo amplamente utilizado, um meio rápido e conveniente de transferir informações que acomoda pequenas notas ou arquivos grandes e permite a comunicação entre indivíduos ou grupos.

O correio eletrônico difere fundamentalmente da maioria dos outros usos das redes, pois um sistema de correio precisa providenciar instâncias quando o destino remoto está temporariamente inalcançável. Para lidar com a entrega adiada, os sistemas de correio eletrônico utilizam uma técnica conhecida como *spooling*. Quando o usuário envia uma mensagem de correio, o sistema local coloca uma cópia em seu armazenamento privado

(chamado spool),* junto com a identificação do emissor, do destinatário, da máquina de destino e tempo de depósito. O sistema, então, inicia a transferência para a máquina remota como uma atividade em segundo plano, permitindo ao emissor prosseguir com outras atividades computacionais. A Figura 24.1 ilustra a concepção.

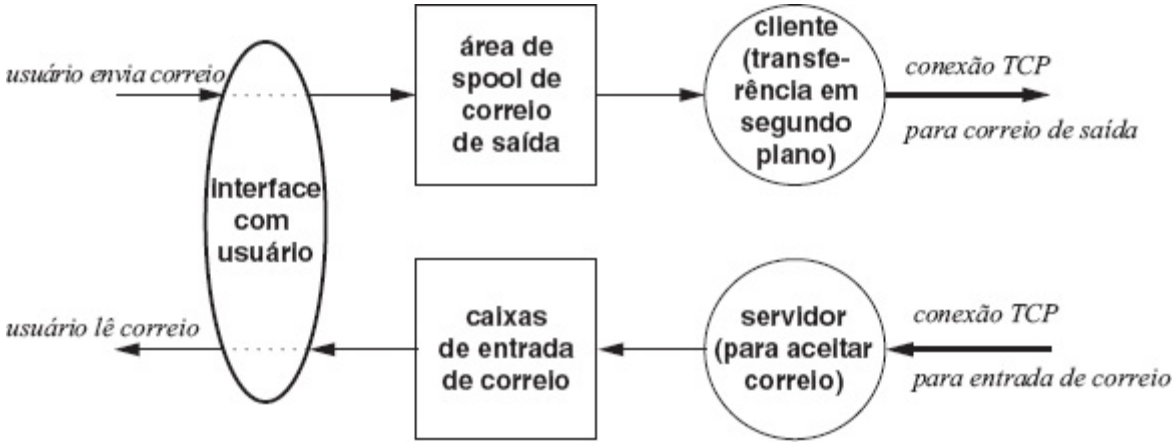


Figura 24.1 Componentes conceituais de um sistema de correio eletrônico. O usuário invoca uma aplicação de interface de correio para depositar ou apanhar correio; toda a transferência ocorre em segundo plano.

O processo de transferência de correio em segundo plano se torna um cliente que usa o sistema de nome de domínio para mapear o nome da máquina de destino para um endereço IP. O cliente, então, tenta formar uma conexão TCP com o servidor de correio da máquina de destino. Se a conexão for bem-sucedida, o cliente transmite uma cópia da mensagem para o servidor remoto, que armazena uma cópia dela temporariamente. Quando o cliente e o servidor concordam que a transferência está completa, ele remove a cópia local e o servidor transfere a cópia para a caixa postal do usuário. Se o cliente não puder formar uma conexão TCP ou se esta falhar, o cliente registra a hora da tentativa e encerra. O sistema de envio de e-mail faz uma varredura pela área de spool periodicamente, em geral uma vez a cada 30 minutos, procurando correio não entregue. Sempre que encontra uma mensagem ou sempre que um usuário deposita novo correio de saída, o processo em segundo plano tenta a entrega. Se descobrir que uma mensagem de correio não pode ser entregue após algumas horas, o software de correio informa ao remetente; depois de um tempo estendido (por exemplo, 3 dias), o software de correio retorna a mensagem ao emissor.

24.3 Nomes e aliases da caixa de correio

Existem três ideias importantes ocultas em nossa descrição simples da entrega de correio. Primeiro, os usuários especificam cada destinatário dando uma string de texto em que há dois itens separados por um sinal de arroba:

usuário@nome-domínio

(user @ domain-name)

O *nome-domínio* designa o nome de domínio de um destino* e *usuário* é o nome da caixa de correio da máquina. Por exemplo, o endereço do correio eletrônico do autor é:

comer@purdue.com.

Segundo, os nomes usados nessas especificações são independentes de outros nomes atribuídos às máquinas. Normalmente, uma caixa de correio é igual ao id de login de um usuário, e o nome de domínio de um computador é usado como destino de correio. Porém, muitos outros projetos são possíveis. Por exemplo, uma caixa de correio pode designar um cargo como *chefe de departamento*. Como o sistema de nome de domínio inclui um tipo de consulta separado para destinos de correio, é possível desacoplar os nomes de destino de correio dos nomes de domínio normais para máquinas. Assim, o correio enviado a um usuário em *exemplo.com* pode ir para uma máquina diferente de uma requisição ping enviada ao mesmo nome. Terceiro, nosso diagrama simplista falha ao considerar o *encaminhamento de correio*, em que algum correio que chega a determinada máquina é encaminhado a outra máquina.

24.4 Expansão de alias e encaminhamento de correio

A maioria de servidores de correio eletrônico fornece um software de *encaminhamento de correio* (*mail forwarder*) que emprega um mecanismo de *expansão de alias de correio*. Um encaminhador permite que cada entrada de mensagem seja enviada para um ou mais destinos. Normalmente, um encaminhador utiliza uma base de dados de aliases de correio para mapear um endereço de entrada do destinatário em um conjunto de endereços *S* e, em seguida, encaminha uma cópia para cada

endereço em *S*. Como eles podem ser muitos-para-um ou um-para-muitos, os mapeamentos de alias aumentam substancialmente a funcionalidade e a conveniência do sistema de correio.

Um único usuário pode ter vários identificadores de correio, ou um grupo pode ter um único alias de correio. No último caso, o conjunto de destinatários associado a um identificador é chamado de *lista de correspondência eletrônica*. A Figura 24.2 ilustra os componentes de um sistema de correio que admite aliases de correio e expansão de lista.

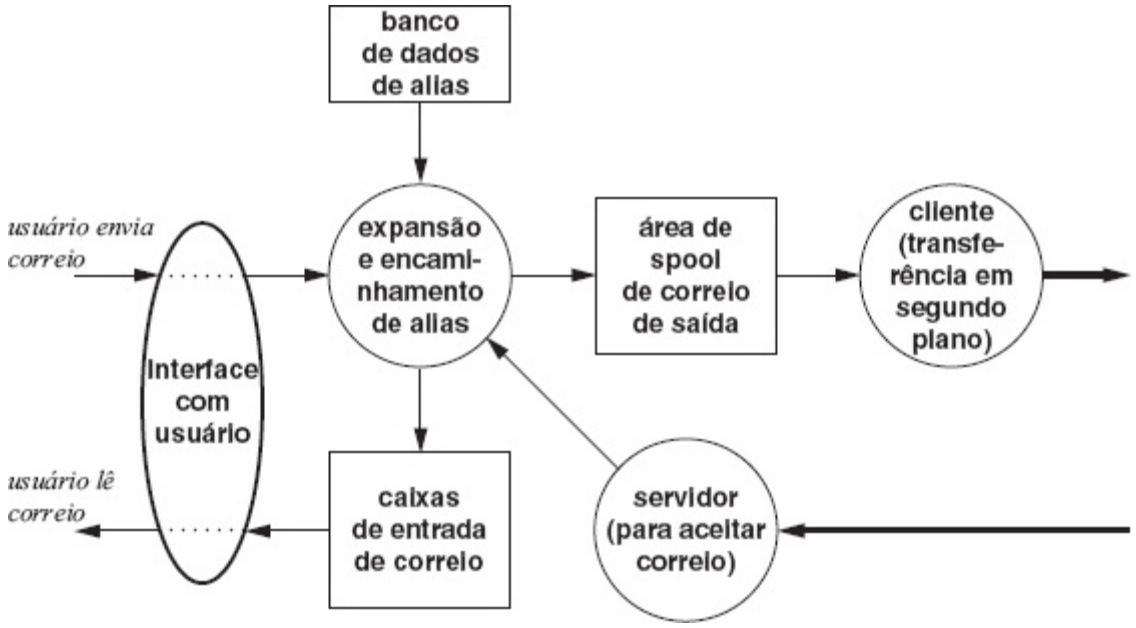


Figura 24.2 Uma extensão do sistema de correio na Figura 24.1, que admite aliases e encaminhamento de correio. O correio que entra e sai passa pelo mecanismo de expansão de alias.

Como mostra a figura, o correio de entrada e saída passa pelo encaminhador de correio que expande aliases. Assim, se o banco de dados de alias especificar que o endereço de correio x é mapeado para o substituto y, a expansão de alias reescreverá o endereço de destino x, trocando por y. O programa de expansão de alias, então, determina se y especifica um endereço local ou remoto de modo que saiba se deve colocar a mensagem na caixa de correio do usuário local ou enviá-la para a fila de correio de saída.

A expansão de alias de correio pode ser perigosa. Suponha que dois sites estabeleçam aliases em conflito. Por exemplo, imagine que o banco de

dados alias do site *A* mapeie o endereço de correio *x* para o endereço de correio *x@B*, e o banco de dados alias do site *B* mapeie o endereço de correio y no endereço *x@A*. Qualquer mensagem de correio enviada para o endereço x do site *A* será encaminhada para o site *B* e, em seguida, de volta para o site *A*, e assim sucessivamente.*

24.5 Padrões do TCP/IP para serviço de correio eletrônico

Lembre-se de que o objetivo do projeto do conjunto do protocolo TCP/IP é providenciar interoperabilidade pela mais vasta gama de sistemas de computador e redes. Visando estender a interoperabilidade do correio eletrônico, o TCP/IP divide seus padrões de correio em dois conjuntos. Um padrão, dado na RFC 2822, especifica o formato sintático usado para mensagens de correio;** o outro padrão especifica os detalhes da troca de correio eletrônico entre dois computadores.

De acordo com a RFC 2822, uma mensagem de correio é representada como texto e dividida em duas partes: um cabeçalho e um corpo, que são separados por uma linha em branco. O padrão para mensagens de correio especifica o formato exato dos cabeçalhos de correio, além da interpretação semântica de cada campo de cabeçalho; ele deixa o formato do corpo a cargo do emissor. Em particular, o padrão especifica que os cabeçalhos contêm texto legível, dividido em linhas que consistem em uma palavra-chave seguida por um sinal de dois pontos e depois por um valor. Algumas palavras-chave são obrigatórias, outras, opcionais, e o restante não é interpretado. Por exemplo, o cabeçalho precisa conter uma linha que especifica o destino. A linha começa com *To:* e contém o endereço de correio eletrônico do destinatário no restante dela. Uma linha que começa com *From:* contém o endereço de correio eletrônico do emissor. Opcionalmente, o emissor pode especificar um endereço para o qual as respostas devem ser enviadas (ou seja, permitir que o emissor especifique que as respostas devem ser enviadas para um endereço diferente da caixa de correio do emissor). Se estiver presente, uma linha que começa com *Reply-to:* especifica o endereço para respostas. Se não houver tal linha, o destinatário usará informações na linha *From:* como endereço de retorno.

O formato da mensagem de correio é escolhido para que se torne fácil processar e transportar dados por máquinas heterogêneas. Manter o formato

do cabeçalho de correio simples permite que ele seja usado em uma grande variedade de sistemas. Restringir as mensagens a texto legível evita os problemas de selecionar uma representação binária padrão e traduzir entre a representação-padrão e a representação da máquina local.

24.6 Simple Mail Transfer Protocol (SMTP)

Além dos formatos de mensagem, o conjunto de protocolos TCP/IP especifica um padrão para a troca de correio entre as máquinas. Ou seja, o padrão especifica o formato exato das mensagens que um cliente em uma máquina utiliza a fim de transferir correio para um servidor em outra máquina. O protocolo de transferência-padrão é conhecido como *Simple Mail Transfer Protocol* (*SMTP*). Como você pode imaginar, o SMTP é mais simples do que o anterior, chamado *Mail Transfer Protocol* (*MTP*). O protocolo SMTP foca especificamente como o sistema de entrega de correio subjacente passa mensagens por uma internet de uma máquina para outra. Ele não especifica como o sistema de correio aceita correio de um usuá rio ou como a interface com o usuário apresenta o correio que chega ao usuário. Além disso, o SMTP não especifica como o correio é armazenado ou com que frequência o sistema de correio tenta enviar mensagens.

O SPMT é surpreendentemente simples. A comunicação segue um projeto predominante em muitos protocolos de camada de aplicação: a comunicação entre um cliente e o servidor consiste em texto ASCII legível. Cada linha começa com um nome de comando, que pode ser um nome abreviado ou, então, um número de três dígitos; o texto restante na linha ou dá argumentos para o comando ou texto que humanos usam para depurar o software de correio. Embora o SMTP defina rigidamente o formato do comando, os humanos podem facilmente ler uma transcrição das interações entre um cliente e o servidor, pois cada comando aparece em uma linha separada. Inicialmente, o cliente estabelece uma conexão de fluxo confiável com o servidor e espera que ele envie uma mensagem *220 READY FOR MAIL*. (Se o servidor estiver sobrecarregado, ele pode adiar o envio da mensagem *220* temporariamente.) Ao receber a mensagem *220*, o cliente envia um comando *HELO** (se o cliente admitir extensões das RFC 2821, ele envia o comando alternativo *EHLO*). O final de uma linha marca o final de um comando. O servidor responde identificando-se. Quando a

comunicação tiver sido estabelecida, o cliente pode transmitir uma ou mais mensagens de correio eletrônico e, em seguida, encerrar a conexão. O servidor precisa confirmar cada comando. O cliente pode abortar a conexão inteira ou mesmo a transferência da mensagem atual.

A transferência de mensagens começa com um comando *MAIL* que dá a identificação do emissor, e também um campo *FROM:* que contém o endereço a que os erros devem ser relatados. Um servidor prepara suas estruturas de dados para receber uma nova mensagem de correio e responde a um comando *MAIL* enviando a resposta *250*, que significa que tudo está bem e que o cliente deve prosseguir. A resposta completa consiste no texto *250 OK*.

Após um comando *MAIL* bem-sucedido, o cliente emite uma série de comandos *RCPT* que identificam os destinatários da mensagem de correio. O servidor precisa confirmar cada comando *RCPT* enviando *250 OK* ou então a mensagem de erro *550 No such user here*.

Depois que todos os comandos *RCPT* tiverem sido confirmados, o cliente emite um comando *DATA*. Essencialmente, tal comando *DATA* informa ao servidor que o cliente está pronto para transferir o corpo de uma mensagem de correio. O servidor responde com a mensagem *354 Start mail input* e especifica a sequência de caracteres usada para terminar a mensagem de correio. A sequência de término consiste em cinco caracteres: carriage return, line feed, period, carriage return e line feed.**

Um exemplo esclarecerá a troca SMTP. Suponha que o usuário Smith, no host *Alpha.edu*, envie uma mensagem aos usuários Jone, Green e Brown, no host *Beta.gov*. O software cliente SMTP no host *Alpha.edu* contata o software servidor de SMTP no host Beta.gov e inicia a troca mostrada na Figura 24.3.

```
S: 220 Beta.gov Simple Mail Transfer Service Ready
C: HELO Alpha.edu
S: 250 Beta.gov

C: MAIL FROM:<Smith@Alpha.edu>
S: 250 OK

C: RCPT TO:<Jones@Beta.gov>
S: 250 OK

C: RCPT TO:<Green@Beta.gov>
S: 550 No such user here

C: RCPT TO:<Brown@Beta.gov>
S: 250 OK

C: DATA
S: 354 Start mail input;end with <CR><LF>.<CR><LF>
C: ...sends body of mail message...
C: ...continues for as many lines as message contains
C: <CR><LF>.<CR><LF>
S: 250 OK

C: QUIT
S: 221 Beta.gov Service closing transmission channel
```

Figura 24.3 Exemplo de transferência SMTP de Alpha.edu para Beta.gov durante a qual o destinatário Green não é reconhecido. As linhas que começam com "C:" são transmitidas pelo cliente (Alpha), e as que começam com "S:" são transmitidas pelo servidor.

No exemplo, o servidor rejeita o destinatário *Green*, pois não reconhece o nome como um destino de correio válido (ou seja, *Green* nem é um usuário nem uma lista de correio). O protocolo SMTP não especifica os detalhes de como um cliente trata desses erros – ele mesmo precisa decidir. Embora os clientes possam abortar a entrega completamente se houver um erro, a maioria deles não faz isso. Ao contrário, eles continuam a entrega a todos os destinatários válidos e depois informam sobre problemas com o emissor original. Normalmente, o cliente informa erros usando o correio eletrônico. A mensagem de erro contém um resumo do erro, além do cabeçalho das mensagens de correio que causaram o problema.

Quando acaba de enviar todas as mensagens de correio, um cliente emite

um comando *QUIT*. O outro lado responde com o comando *221*, que significa que ele concorda em terminar. Os dois lados, então, fecham a conexão TCP de forma controlada.

O SMTP é muito mais complexo do que esboçamos aqui. Por exemplo, se um usuário tiver mudado, o servidor poderá saber o novo endereço de sua caixa de correio. O SMTP permite ao servidor informar o cliente sobre o novo endereço, de modo que ele possa usá-lo no futuro. Ao passar tal informação, o servidor pode decidir encaminhar o correio que disparou a mensagem ou pode requisitar que o cliente tenha responsabilidade pelo encaminhamento. Além disso, o SMTP inclui extensões *Transport Layer Security* (*TLS*), que possibilitam que uma sessão SMTP seja criptografada.

24.7 Recuperação de correio e protocolos de manipulação de caixas de correio

O esquema de transferência SMTP descrito anteriormente mostra que um servidor precisa permanecer pronto para aceitar o correio eletrônico o tempo todo. O cenário funciona bem se o servidor for executado em um computador que possui uma conexão permanente com a Internet, mas não funciona bem para um dispositivo que possui conectividade intermitente (por exemplo, smartphone que frequentemente esteja desligado ou indisponível). Não faz sentido para um dispositivo com conectividade intermitente executar um servidor de e-mail porque o servidor só estará disponível enquanto o dispositivo do usuário estiver conectado – todas as outras tentativas de contatar o servidor falharão, e o correio enviado ao usuário permanecerá sem ser entregue. Surge, portanto, a pergunta: "como um usuário sem uma conexão permanente recebe correio eletrônico?".

A resposta para a pergunta está em um processo de entrega em dois estágios. No primeiro, cada usuário recebe uma caixa de correio em um computador que sempre está ligada e possui uma conexão permanente com a Internet. O computador executa um servidor SMTP convencional, que permanece pronto para aceitar correio eletrônico. No segundo estágio, o usuário se conecta à Internet e depois executa um protocolo que apanha mensagens da caixa de correio permanente. O protocolo as transfere para o computador do usuário, de onde elas podem ser lidas. A Figura 24.4 ilustra a ideia.

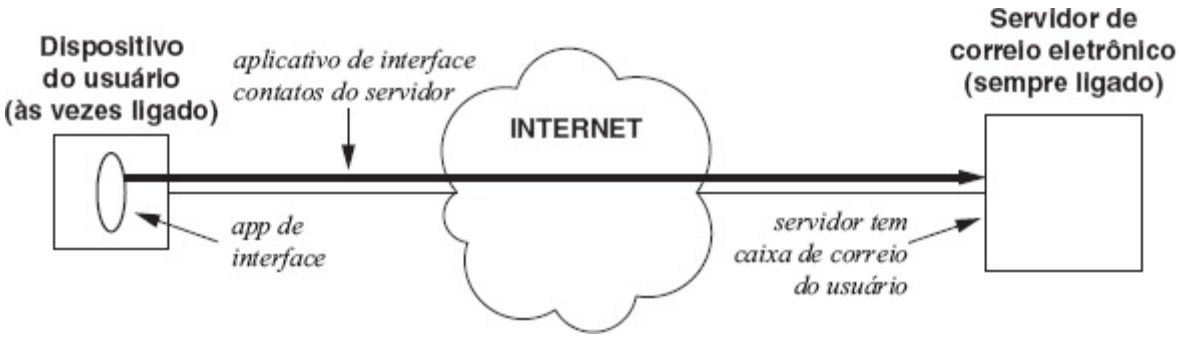


Figura 24.4 Ilustração de acesso de correio eletrônico quando o servidor de e-mail e a caixa de correio do usuário não estão localizados no computador do usuário.

Uma variedade de técnicas tem sido utilizada a fim de permitir que a caixa de correio de um usuário esteja em um computador remoto. Por exemplo, muitos provedores que oferecem serviço de correio eletrônico providenciam uma interface baseada na web para correio eletrônico. Ou seja, um usuário inicia um navegador da web e se conecta a uma página web especial que mostra correio eletrônico. Empresas como a Microsoft oferecem mecanismos proprietários que permitem a uma organização ter um servidor de correio eletrônico único, que os usuários podem acessar remotamente.

O acesso remoto foi lançado pelo IETF, que define dois protocolos que permitem a um aplicativo remoto acessar o correio em uma caixa de correio permanente, que fica armazenada em um servidor. Embora tenham funcionalidade semelhante, os protocolos utilizam técnicas opostas: um permite que o usuário baixe uma cópia das mensagens, e o outro possibilita a um usuário ver e manipular mensagens no servidor. A duas seções seguintes descrevem os dois protocolos.

24.7.1 Post Office Protocol

O protocolo mais popular usado para transferir mensagens de correio eletrônico de uma caixa de correio permanente para um computador local é conhecido como versão 3 do *Post Office Protocol* (*POP3*); uma versão segura do protocolo é conhecida como *POP3S*. O usuário invoca um cliente POP3, que cria uma conexão TCP com um servidor POP3 no computador da caixa de correio. O usuário primeiro envia um *login* e uma *senha* para autenticar a sessão. Quando a autenticação tiver sido aceita, o cliente envia comandos para apanhar uma cópia de uma ou de mais mensagens e excluir

a mensagem da caixa de correio permanente. As mensagens são armazenadas e transferidas como arquivos de texto no formato-padrão especificado pela RFC 2822.

Observe que o computador com uma caixa de correio permanente precisa executar dois servidores – um servidor SMTP e um servidor POP3. O servidor SMTP aceita o correio eletrônico enviado a um usuário e coloca cada mensagem recebida na caixa de correio permanente desse usuário. O servidor POP3, por sua vez, permite ao usuário examinar cada mensagem em sua caixa postal, salvar uma cópia dela no computador local e apagar a mensagem da caixa de correio no servidor. A fim de garantir a operação correta, os dois servidores precisam coordenar o uso da caixa de correio, para que, se uma mensagem chegar por SMTP enquanto um usuário estiver extraindo mensagens via POP3, a caixa de correio fique em um estado válido.

24.7.2 Internet Message Access Protocol

A versão 4 do *Internet Message Access Protocol* (*IMAP4*) é uma alternativa para POP3 que permite aos usuários visualizarem e manipularem mensagens no servidor; também foi definida uma versão segura do IMAP4, conhecida como IMAPS. Como o POP3, o IMAP4 define uma abstração conhecida como uma *caixa de correio*; as caixas de correio estão localizadas no mesmo computador que um servidor. Também como POP3, um usuário executa um aplicativo que se torna um cliente IMAP4. E o aplicativo contata o servidor para ver e manipular as mensagens. No entanto, ao contrário do POP3, o IMAP4 permite a um usuário acessar as mensagens de correio de vários locais (por exemplo, do trabalho ou de casa), garantindo que todas as cópias sejam sincronizadas e coerentes.

O IMAP4 também fornece funcionalidade estendida para recuperação de mensagens e processamento. Um usuário pode obter informações sobre uma mensagem ou mesmo examinar os campos de cabeçalho sem apanhar a mensagem inteira. Além disso, um usuário pode procurar uma string especificada e recuperar partes de uma mensagem. Essa recuperação parcial é especialmente útil para conexões de baixa velocidade, pois significa que um usuário não precisa fazer download de informações inúteis.

24.8 As extensões MIME para dados não ASCII

Os padrões da Internet para correio eletrônico foram criados quando as mensagens de e-mail consistiam em texto. Embora os usuários gostassem

do correio eletrônico, eles queriam uma maneira de enviar anexos (por exemplo, arquivos de dados) juntamente com elas. Consequentemente, o IETF criou *Multipurpose Internet Mail Extensions* (*MIME*) para permitir a transmissão de itens de dados não ASCII através de correio eletrônico. A MIME não altera ou substitui protocolos como SMTP, POP3 e IMAP4. Em vez disso, permite que quaisquer dados sejam codificados em ASCII e depois transmitidos em uma mensagem de correio eletrônico padrão. Para acomodar tipos de dados e representações quaisquer, cada mensagem MIME inclui informações que dizem ao destinatário o tipo de dado e a codificação utilizada. A informação MIME reside no cabeçalho de correio RFC 2822 – as linhas de cabeçalho MIME especificam a versão MIME, o tipo de dados e a codificação que foi usada para converter os dados para ASCII. A maioria dos usuários nunca viu a codificação MIME porque um tópico aplicativo leitor de correio eletrônico remove ou oculta esses detalhes.

A Figura 24.5 ilustra uma mensagem MIME em que há uma fotografia-padrão na representação JPEG.* A imagem JPEG foi convertida para uma representação ASCII de 7 bits usando a codificação *base64*.

```
From:bill@acollege.edu
To:john@example.com
MIME-Version:1.0
Content-
Type:image/jpeg
Content-Transfer-
Encoding:base64

…dados para a imagem seguem aqui…
```

Figure 24.5 Um exemplo de cabeçalho de uma mensagem MIME. Linhas de cabeçalho identificam o tipo de dados e a codificação utilizada.

Na figura, a linha de cabeçalho *MIME-Version:* declara que a mensagem foi composta usando a versão 1.0 do protocolo MIME. A declaração *Content-Type:* especifica que os dados são uma imagem JPEG, e o cabeçalho *Content-Transfer-Encoding:* declara que a codificação *base64* foi usada para converter a imagem para ASCII.

A codificação base64 é análoga ao hexadecimal por permitir que valores binários arbitrários sejam representados usando caracteres imprimíveis. Em

vez de 16 caracteres, a base64 usa 64, o que faz com que o arquivo resultante seja menor. A base64 foi escolhida para prover 64 caracteres ASCII, que têm a mesma representação por várias versões dos conjuntos de caracteres inglês Isso.** Desse modo, um receptor pode ter garantia de que a imagem extraída dos dados codificados é exatamente a mesma imagem original.

Caso se examinassem os dados realmente transferidos, eles se pareceriam com um fluxo sem sentido de caracteres. Por exemplo, a Figura 24.6 mostra as primeiras linhas de uma imagem jpeg que foi codificada em base64 para a transmissão com MIME.

```
/9j/4AAQSkZJRgABAQEAYABgAAD/4QBERXhpZgAATU0AK
gAAAAgAA0AAAAMAAAABAAAAAEABAAEAAAABAAAA
AEACAAIAAAAKAAAAMgAAAB0d2ltZy5jb20A/9sAQwAN
CQoLCggNCwsLDw4NEBQhFRQSEhQoHR4YITAqMjEvKi4tN
DtLQDQ4RzktLkJZQkdOUFRVVDM/XWNcUmJLU1RR/9sAQw
EODw8UERQnFRUnUTYuNlFRUVFRUVFRUVFRUVFR
UVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRU
VFR/8AAEQgAgACAAwEiAAIRAQMRAf/EAB8AAAEFAQEBA
QEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAw
MCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQyg
ZGhCCNCscEVUtHwJDNicoIJChYX
```

Figura 24.6 Exemplo da codificação base64 usada com MIME. As linhas são tiradas a partir de um grande arquivo jpeg.

Para visualizar a imagem em uma figura, um aplicativo de correio eletrônico do receptor deve primeiro converter a partir de codificação *base64* de volta para binário e, em seguida, executar um aplicativo que exibe uma imagem JPEG na tela do usuário. Na maioria de sistemas de correio eletrônico, a conversão é feita automaticamente; um usuário vê ícones ou arquivos reais que foram anexados a uma mensagem de correio eletrônico.

Como um aplicativo de correio eletrônico sabe como lidar com cada anexo? O padrão MIME especifica que uma declaração *Content-Type* precisa conter dois identificadores, um *tipo de conteúdo* e um *subtipo*, separados por uma barra. No exemplo, *image* é o tipo de conteúdo, e *jpeg*, o subtipo.

O padrão define sete tipos de conteúdo básicos, os subtipos válidos para cada um e codificações de transferência. Por exemplo, embora uma *image*

deva ser de subtipo *jpeg* ou *gif*, tipo de conteúdo (contente-type) *text* não pode usar qualquer um destes. Além dos tipos e subtipos-padrão, MIME permite a um emissor e um receptor definirem tipos de conteúdo privados.* A Figura 24.7 lista os sete tipos de conteúdo básicos.

Tipo de conteúdo	Usado quando os dados na mensagem são
text	**textuais (por exemplo, um documento)**
image	**uma fotografia ou imagem gerada por computador**
audio	**uma gravação de som**
video	**uma gravação de vídeo que inclui movimento**
application	**dados brutos para um programa**
multipart	**mensagens múltiplas que possuem tipo de conteúdo e codificação separados**
message	**uma mensagem de correio eletrônico inteira (por exemplo, um memorando que foi encaminhado) ou uma referência externa a uma mensagem (por exemplo, um servidor FTP e nome de arquivo)**

Figura 24.7 Os sete tipos básicos que podem aparecer em uma declaração Content-Type MIME e seus significados.

24.9 Mensagens MIME multiparte

O tipo de conteúdo MIME multiparte é útil por acrescentar uma flexibilidade considerável. O padrão define quatro subtipos possíveis para uma mensagem multiparte, cada qual com uma funcionalidade importante. O subtipo *mixed* permite que em uma única mensagem haja várias submensagens independentes, cada uma podendo ter tipo e codificação independentes. As mensagens multiparte misturadas tornam possível incluir texto, gráficos e áudio em uma única mensagem, ou enviar um memorando com segmentos de dados adicionais conectados, semelhante aos anexos incluídos com uma carta comercial. O subtipo *alternative* permite que uma única mensagem inclua várias representações dos mesmos dados. Mensagens multiparte alternativas são úteis quando se envia um memorando para muitos destinatários que não usam todos as mesmas instalações de hardware ou o mesmo aplicativo de software. Por exemplo,

pode-se enviar um documento como texto ASCII puro e formatado, permitindo que os destinatários que têm computadores com capacidades gráficas selecionem a forma formatada para exibição.** O subtipo *parallel* permite que uma única mensagem inclua subpartes exibidas em conjunto (por exemplo, subpartes vídeo e áudio que precisam ser tocadas simultaneamente). Finalmente, o subtipo *digest* permite que uma única mensagem contenha um conjunto de outras mensagens (por exemplo, uma coleção de mensagens de correio eletrônico de uma discussão).

A Figura 24.8 ilustra uma mensagem de correio eletrônico que contém duas partes. A primeira é uma mensagem em texto simples, e a segunda, uma imagem.

A Figura 24.8 ilustra alguns detalhes do MIME. Por exemplo, cada linha de cabeçalho pode conter parâmetros no formato *X=Y* depois das declarações básicas. A palavra-chave *Boundary=* após a declaração do tipo de conteúdo multiparte no cabeçalho define a string usada para separar partes da mensagem. No exemplo, o emissor selecionou a string *StartOfNextPart* para servir como limite. Se forem incluídas, as declarações do tipo de conteúdo e codificação de transferência para uma submensagem vêm imediatamente após a linha de limite. No exemplo, a segunda submensagem é declarada como sendo *Graphics Interchange Format* (*GIF*) imagem.

```
From:bill@acollege.edu
To:john@example.com
MIME-Version:1.0
Content-
Type:Multipart/Mixed;Boundary=StartOfNextPart
--StartOfNextPart
Content-Type:text/plain
Content-Transfer-Encoding:7bit
John,
Aqui está a foto de nossa pesquisa de
laboratório que prometi enviar a você.Você pode
ver o equipamento que doou.
Novamente,obrigado,
Bill
```

```
--StartOfNextPart
Content-Type:image/gif
Content-Transfer-Encoding:base64

.....dados para a imagem
```

Figura 24.8 Um exemplo de uma mensagem multiparte misturada MIME. Cada parte da mensagem pode ter um tipo de conteúdo independente.

24.10 Resumo

O correio eletrônico está entre os serviços de aplicação mais utilizados da Internet. Assim como os serviços TCP/IP, ele segue o paradigma cliente-servidor. Um sistema de correio coloca em buffer as mensagens que saem e que chegam, permitindo que uma transferência de cliente e servidor ocorra em segundo plano.

O conjunto de protocolos TCP/IP fornece dois padrões separados para o formato de mensagem de correio e transferência de correio, que especificam o formato da mensagem e os detalhes da transferência dela. O formato da mensagem de correio, chamado 2822, usa uma linha em branco para separar um cabeçalho de mensagem e o corpo. O Simple Mail Transfer Protocol (SMTP) define como um sistema de correio em uma máquina transfere correio para um servidor em outra máquina. A versão 3 do Post Office Protocol (POP3) e a versão 4 do Internet Message Access Protocol (IMAP4) especificam como um usuário pode apanhar o conteúdo de uma caixa de correio; eles permitem que um usuário tenha uma caixa de correio permanente em um computador com conectividade contínua com a Internet e acesse o conteúdo a partir de um computador com conectividade intermitente.

Multipurpose Internet Mail Extensions (MIME) é um mecanismo que permite que quaisquer dados sejam transferidos usando SMTP. MIME acrescenta linhas ao cabeçalho de uma mensagem de correio eletrônico, para definir o tipo dos dados e a codificação utilizada. O tipo multiparte misturado do MIME possibilita que uma única mensagem contenha vários tipos de dados.

EXERCÍCIOS

24.1 Descubra se o seu sistema de computação permite que você invoque o SMTP diretamente.

24.2 Construa um cliente SMTP e use-o para transferir uma mensagem de correio.

24.3 Verifique se você pode enviar um correio por um encaminhador de correio (mail forwarder) de volta para você.

24.4 Crie uma lista de formatos de endereço de correio que o seu site utiliza e escreva um conjunto de regras para analisá-los.

24.5 Descubra como um sistema Linux pode ser configurado para atuar como encaminhador de correio (mail forwader).

24.6 Descubra com que frequência o seu sistema de correio local tenta fazer uma entrega e por quanto tempo ele continuará antes de desistir.

24.7 Muitos sistemas de correio permitem que os usuários direcionem o correio que chega para um programa, em vez de armazená-lo em uma caixa de correio. Construa um programa que aceita seu correio que chega, coloca-o em um arquivo e depois envia uma resposta para dizer ao remetente que você está em férias.

24.8 Leia o padrão SMTP com cuidado. Depois, use TELNET para se conectar à porta SMTP em uma máquina remota e digite os comandos para pedir ao servidor SMTP remoto que expanda um alias de correio.

24.9 Um usuário recebe correio em que o campo *To*: especifica a string *important-people*. O correio foi enviado de um computador em que o alias *important-people* não inclui identificadores válidos de caixa de correio. Leia a especificação SMTP cuidadosamente para ver como essa situação é possível.

24.10 POP3 separa recuperação e exclusão de mensagem, permitindo que um usuário apanhe e veja uma mensagem sem excluí-la da caixa de correio permanente. Quais são as vantagens e as desvantagens dessa separação?

24.11 Leia sobre POP3. Como o comando *TOP* opera e por que ele é útil?

24.12 Leia sobre IMAP4. Como o IMAP4 garante a consistência quando vários clientes simultâneos acessam determinada caixa de correio ao mesmo tempo?

24.13 Leia as RFCs MIME com atenção. Que servidores podem ser especificados em uma referência MIME externa?

24.14 Se você usasse um smartphone com um limite para o número de bytes de dados recebidos a cada mês, você preferiria POP3 ou IMAP4?

Explique.

24.15 Para encobrir os destinatários, mensagens de spam, muitas vezes, listam destinatários não revelados (*Undisclosed Recipients*) no campo *To:*. A sua interface de correio eletrônico permite a você enviar uma mensagem para um usuário a qual aparece na caixa postal dele como enviada para *Undisclosed Recipients*? Explique.

---

*Uma área de spool de correio às vezes é chamada de *fila de correio*, embora o termo seja tecnicamente impreciso.

*Tecnicamente, o nome de domínio especifica um permutador de correio, e não um host.

*Na prática, a maior parte dos encaminhadores de correio (mail forwarders) termina as mensagens após o número de trocas atingir um limite predeterminado.

**O padrão original foi especificado na RFC 822; o IETF adiou a emissão do substituto até a RFC 2822, para que os números se correlacionassem.

*HELO é uma abreviação para "hello".

**SMTP utiliza CR-LF para terminar uma linha e não permite que o corpo de uma mensagem tenha um ponto isolado em uma linha.

*JPEG é o padrão Joint Picture Encoding Group usado para imagens digitais.

**Os caracteres consistem em 26 letras maiúsculas, 26 letras minúsculas, dez dígitos, o sinal de "+" e o caractere "/" (barra).

*Para evitar conflitos de nome em potencial, o padrão exige que os nomes escolhidos para tipos de conteúdo privados comecem com dois caracteres string *X-*.

**Muitos sistemas de correio eletrônico usam a alternativa do subtipo MIME para enviar uma mensagem em ambos os formatos, ASCII e HTML.

CAPÍTULO 25

World Wide Web (HTTP)

CONTEÚDOS DO CAPÍTULO

25.1 Introdução

Este capítulo continua a discussão sobre aplicações que usam a tecnologia TCP/IP, focando a aplicação de maior impacto: a *World Wide Web* (a *Web*). Após uma breve visão geral dos conceitos, o capítulo examina o principal protocolo usado para transferir uma página web entre um servidor e um navegador. A discussão aborda ainda o cache (caching) e também o mecanismo básico de transferência.

25.2 A importância da Web

Durante o início da história da Internet, as transferências de dados usando o *File Transfer Protocol* (*FTP*) eram responsáveis por um terço do tráfego da Internet, mais do que qualquer outra aplicação. Entretanto, desde que surgiu, no início da década de 1990, a Web teve uma alta taxa de crescimento. Por volta de 1995, o tráfego da Web ultrapassou o FTP, para se tornar o maior consumidor de largura de banda do backbone da Internet, e manteve-se como o principal aplicativo.

O impacto da Web não pode ser entendido apenas pelas estatísticas de tráfego. Mais pessoas sabem a respeito dela e a usam mais do que qualquer outra aplicação da Internet. Na verdade, para muitos usuários, a Internet e a Web significam a mesma coisa.

25.3 Componentes arquitetônicos

Conceitualmente, a Web é enorme conjunto de documentos, chamado *páginas da web* (web pages), que estão disponíveis aos usuários da Internet. Cada página da web é classificada como um documento de *hipermídia*. O prefixo *hiper* é usado porque um documento pode conter links selecionáveis, que se referem a outros documentos relacionados, e o sufixo *mídia* é usado para indicar que um documento pode conter itens que não sejam texto (por exemplo, imagens gráficas).

Dois blocos de montagem principais são usados para implementar a Web no topo da Internet global: um *navegador web* (web browser) e um *servidor web* (web server). Um navegador web consiste em um aplicativo que um usuário invoca para acessar e exibir uma página web e torna-se um cliente que contata o servidor web apropriado para obter uma cópia da página especificada. Como determinado servidor pode gerenciar mais de uma página web, um navegador precisa especificar a página exata quando faz uma solicitação.

O padrão de representação de dados para uma página web depende de seu conteúdo. Por exemplo, representações gráficas padrão, como *Graphics Interchange Format* (*GIF*) ou *Joint Picture Encoding Group* (*JPEG*), podem ser usadas para uma página que contém uma única imagem gráfica. Páginas que apresentam uma mistura de texto e outros itens são representadas usando *HyperText Markup Language* (*HTML*). Um documento HTML consiste em um arquivo que contém texto junto com comandos embutidos, chamados de *tags*, que dão orientações para exibição. Uma tag é delimitada por símbolos de menor e maior; algumas tags vêm em pares que se aplicam a todos os itens entre o par. Por exemplo, os dois comandos <CENTER> e </CENTER> fazem com que os itens entre eles sejam centralizados na janela do navegador.

25.4 Uniform Resource Locators

Cada página web recebe um nome exclusivo usado para identificá-la. O nome, que é *Uniform Resource Locator* (*URL*), começa com uma especificação do esquema usado para acessar a página. Com efeito, o esquema especifica o protocolo de transferência; o formato do restante do URL depende do esquema. Por exemplo, um URL* que segue o *esquema*

http tem o seguinte formato:**

nomehost [:porta] / caminho [; parâmetros][? consulta]

Onde o itálico indica um item a ser fornecido, e os colchetes indicam um item opcional. Por enquanto, é suficiente entender que a string *nomehost* (hostname) especifica o nome de domínio, endereço decimal com ponto do IPv4 ou endereço IPv6 hexadecimal com dois pontos do computador em que o servidor da página opera o servidor; *:porta* é um número opcional de porta de protocolo necessário apenas em casos em que o servidor não usa uma porta bem conhecida (*80*); *caminho* é uma string que identifica um documento em particular no servidor; *;parâmetros* é uma string opcional que especifica parâmetros adicionais fornecidos pelo cliente; *?consulta* é uma string opcional usada quando o navegador envia uma pergunta. Um usuário provavelmente nem sequer verá ou usará as partes opcionais diretamente. Em vez disso, os URLs que um usuário insere contêm apenas um *nomehost* e um *caminho*.

Por exemplo, o URL

http://www.cs.purdue.edu/ people / comer /

especifica a página web do autor na Purdue University. O servidor opera no computador *www.cs.purdue.edu*, e o documento tem o nome */people/comer/*.

Os padrões de protocolo distinguem entre a forma absoluta de um URL, ilustrada aqui, e uma forma relativa. Um URL relativo, que raramente é visto por um usuário, só tem significado depois de ter sido estabelecido com um servidor web específico. Por exemplo, ao se comunicar com o servidor *www.cs.purdue.edu*, somente a string */people/comer/* precisa especificar o documento nomeado pelo URL absoluto anterior. Podemos resumir da maneira a seguir.

Cada página web recebe um identificador exclusivo conhecido como Uniform Resource Locator (URL). O formato absoluto de um URL contém uma especificação completa; um formato relativo que omite o endereço do servidor só é útil quando o servidor é conhecido

implicitamente.

25.5 Um documento HTML de exemplo

Um exemplo ilustrará como um URL é produzido a partir de um *link selecionável* em um documento. Para cada link desse tipo, um documento contém um par de valores: um item a ser exibido na tela e um URL para acompanhar se o usuário selecionar o item. Em HTML, um par de tags e, que são conhecidas como *âncora*, define um link; um URL é acrescentado à primeira tag, e os itens a serem exibidos são colocados entre as duas tags. Por exemplo, o documento HTML a seguir contém um link selecionável:

```
<HTML>
  The author of this text is
  <A HREF="http://www.cs.purdue.edu/people/comer">
  Douglas Comer.</A>
</HTML>
```

Quando o documento é exibido, uma única linha de texto aparece na tela:

O autor deste texto é Douglas Comer

O navegador sublinha a expressão Douglas Comer para indicar que ela corresponde a um link selecionável. Internamente, o navegador armazena o URL da tag <A>, que ele segue quando o usuário seleciona o link.

25.6 Hypertext Transfer Protocol

O protocolo usado para a comunicação entre um navegador e um servidor web ou entre máquinas intermediárias e servidores web é conhecido como *HyperText Transfer Protocol* (HTTP). HTTP tem o conjunto de características descrito a seguir.

- *Camada de aplicação.* O HTTP opera no nível de aplicação. Ele considera um protocolo de transporte confiável, orientado à conexão, como TCP, mas não provê confiabilidade ou retransmissão por si só.
- *Requisição/resposta.* Quando a sessão de transporte tiver sido estabelecida, um lado (normalmente, um navegador) precisa enviar

uma requisição HTTP à qual o outro lado responde.

- *Sem estado.* Cada requisição HTTP é autocontida; o servidor não mantém um histórico das requisições anteriores ou sessões anteriores.
- *Transferência bidirecional.* Na maioria dos casos, um navegador solicita uma página web, e o servidor transfere uma cópia para o navegador. O HTTP também permite a transferência de um navegador para um servidor (por exemplo, quando um usuário fornece dados).
- *Negociação de capacidade.* O HTTP permite que os navegadores e servidores negociem detalhes como o conjunto de caracteres a ser usado durante as transferências. Um emissor pode especificar as capacidades que oferece, e um receptor pode especificar as capacidades que ele aceita.
- *Suporte para uso de cache.* Para melhorar o tempo de resposta, um navegador coloca em cache uma cópia de cada página web que apanha. Se um usuário requisitar uma página novamente, o navegador pode interrogar o servidor para determinar se o conteúdo da página foi alterado desde que a cópia foi para o cache.
- *Suporte para intermediários.* O HTTP permite que uma máquina ao longo do caminho entre um navegador e um servidor atue como um *servidor proxy* que coloca em cache as páginas web e responde à requisição de um navegador a partir do seu cache.

25.7 A requisição GET do HTPP

No caso mais simples, um navegador contata um servidor web diretamente para obter uma página. O navegador começa com um URL, extrai a seção do nome de host, usa o DNS para mapear o nome a um endereço IP equivalente e utiliza o resultante endereço IP para formar uma conexão TCP com o servidor web. Quando a conexão TCP está pronta, o navegador e o servidor web usam HTTP para se comunicar; o navegador envia uma requisição para recuperar uma página específica e o servidor responde enviando uma cópia da página.

Um navegador envia um comando *GET* do HTTP para requisitar uma página web a partir de um servidor.* A requisição consiste em uma única linha de texto que começa com a palavra-chave *GET* e é seguida por um URL e um número de versão HTTP. Por exemplo, para recuperar a página web no exemplo anterior do servidor *www.cs.purdue.edu*, um navegador

pode enviar a seguinte requisição com um URL absoluto:

```
GET http: / / www . cs . purdue . edu / people / comer /  HTTP / 1.1
```

Quando uma conexão TCP é estabelecida, não existe necessidade de enviar um URL absoluto – o URL relativo a seguir recuperará a mesma página:

```
GET  /  people/comer/  HTTP/1.1
```

Podemos fazer o resumo a seguir.

O Hypertext Transfer Protocol (HTTP) é usado entre um navegador e um servidor web. O navegador envia uma requisição GET à qual um servidor responde enviando o item solicitado.

25.8 Mensagens de erro

Como um servidor web deveria responder ao receber uma requisição ilegal? Na maioria dos casos, a requisição foi enviada por um navegador, e este tentará exibir aquilo que o servidor retornar. Consequentemente, é comum os servidores gerarem mensagens de erro* em HTML válida. Por exemplo, um servidor gera a seguinte mensagem de erro como uma resposta sempre que o servidor não pode honrar um pedido:

```
<HTML>
  <HEAD><TITLE>400 Bad Request</TITLE>
  </HEAD>
  <BODY>
  <H1>Error  In  Request</H1>  Your  browser  sent  a
request that this server could not understand.
  </BODY>
</HTML>
```

O navegador usa a "cabeça" do documento (ou seja, os itens entre <HEAD>e</HEAD>) internamente e mostra apenas o "corpo" ao usuário. O par de tags <H1>e</H1> faz com que o navegador apresente "erro no pedido" (Error In Request) como cabeçalho (ou seja, grande e em negrito), resultando em duas linhas de saída na tela do usuário:

Error In Request (erro no pedido)

Seu navegador enviou um pedido que este servidor não conseguiu entender.

25.9 Conexões persistentes

A primeira versão do HTTP segue o mesmo paradigma do FTP, usando uma conexão TCP por transferência de dados. Um cliente abre uma conexão TCP e envia uma requisição *GET*. O servidor envia uma cópia da página solicitada e fecha a conexão. O paradigma apresenta a vantagem de não ser ambíguo – o cliente simplesmente lê até que uma condição de *fim de arquivo* seja encontrada, e depois fecha sua extremidade de conexão.

A versão 1.1 do HTTP mudou o paradigma básico de um modo fundamental: em vez de usar uma conexão TCP por transferência, a versão 1.1 adota uma técnica de *conexão persistente* como padrão. Ou seja, quando um cliente abre uma conexão TCP para determinado servidor web, ele deixa a conexão estabelecida durante várias requisições e respostas. Assim que um cliente ou um servidor está pronto para fechar a conexão, ele informa o outro lado, e a conexão é fechada.

A principal vantagem das conexões persistentes está no overhead reduzido – menos conexões TCP significa menor latência de resposta, menos overhead nas redes subjacentes, menos memória usada para buffers e menos tempo de CPU utilizado. Um navegador que utilize uma conexão persistente pode otimizar ainda mais com o *pipelining* de requisições (ou seja, enviar requisições de um lado para outro sem esperar por uma resposta). O pipelining é especialmente atraente em situações em que várias imagens precisam ser encaminhadas para determinada página web, e a internet subjacente possui rendimento alto e longa espera.

A principal desvantagem de usar uma conexão persistente está na necessidade de identificar o início e o final de cada item enviado pela conexão. Existem duas técnicas possíveis para lidar com a situação: enviar um tamanho seguido pelo item ou enviar um valor de sentinela após o item, para marcar o final. O HTTP não pode reservar um valor de sentinela porque os itens transmitidos incluem imagens gráficas que podem conter quaisquer sequências de octetos. Assim, para evitar a ambiguidade entre valores de sentinela e dados, o HTTP usa a técnica de enviar um tamanho seguido por um item com esse tamanho.

25.10 Tamanho de dados e saída do programa

Pode não ser conveniente ou mesmo possível para um servidor web saber o tamanho da página web antes de enviá-la. Para entender por que, é preciso saber que algumas páginas web são geradas na requisição. Ou seja, o servidor usa uma tecnologia do tipo *Common Gateway Interface* (*CGI*), que permite a um programa executando na máquina servidora criar uma página na web. Quando chega uma requisição que corresponde a uma página gerada por CGI, o servidor executa o programa CGI apropriado e envia a saída do programa de volta ao cliente como resposta. A geração dinâmica de página web permite a criação de informações que são atuais (por exemplo, uma lista dos placares atuais em eventos esportivos ou uma lista de sites que correspondam a um termo de pesquisa), mas o servidor não pode saber o tamanho exato dos dados com antecedência. Além do mais, não é desejável salvar os dados em arquivo antes de enviá-los por dois motivos: isso utiliza recursos no servidor e atrasa a transmissão. Assim, para providenciar páginas dinâmicas na web, o padrão HTTP especifica que, se o servidor não souber o tamanho de um item *a priori*, ele poderá informar ao navegador que fechará a conexão depois de transmitir o item. Podemos fazer o resumo a seguir.

Para permitir que uma conexão TCP persista por várias requisições e respostas, o HTTP envia um tamanho antes de cada resposta. Se ele não souber o tamanho, um servidor informa ao cliente, envia a resposta e depois fecha a conexão.

25.11 Codificação por tamanho e cabeçalhos

Que representação um servidor deve usar para enviar informações de tamanho? É interessante notar que o HTTP pega emprestado o formato básico do e-mail, usando o mesmo formato especificado no RFC 2822 e MIME Extensions.* Como um padrão de mensagem de e-mail RFC 2822, cada transmissão HTTP contém um cabeçalho, uma linha em branco e o documento sendo enviado. Além do mais, cada linha no cabeçalho contém uma palavra-chave, um sinal de dois pontos e informações. A Figura 25.1 lista alguns dos cabeçalhos possíveis acompanhados de seus significados.

Cabeçalho	Significado
Content-Length	**Tamanho do item em octetos**

Content-Type	**Tipo do documento**
Content-Encoding	**Codificação usada para o documento**
Content-Language	**Linguagem(ns) usada(s) no documento**

Figura 25.1 Exemplos de cabeçalhos que podem aparecer antes de um documento. O Content-Type e Content-Encoding são tomados diretamente do MIME.

Como exemplo, considere a Figura 25.2, que mostra alguns dos cabeçalhos que são usados quando um documento HTML curto (34 caracteres) é transferido por uma conexão TCP persistente.

```
Content-Length:34
Content-Language:en
Content-Encoding:ascii

<HTML>Atrivialexample.</HTML>
```

Figura 25.2 Uma ilustração de uma transferência HTTP com linhas de cabeçalho usadas para especificar atributos, uma linha em branco e o próprio documento.
É necessário um cabeçalho *Content-Length* se a conexão for persistente.

Além dos exemplos mostrados na figura, o HTTP inclui uma grande variedade de cabeçalhos que permitem que um navegador e um servidor troquem metainformações. Por exemplo, se um servidor não souber o tamanho de uma página, ele fecha a conexão depois de enviar o documento. Porém, o servidor não atua sem advertência – cabe a ele informar ao navegador para esperar um fechamento. Para fazer isso, o servidor inclui um cabeçalho *Connection* antes do documento no local de um cabeçalho *Content-Length*:

```
Connection: close
```

Ao receber um cabeçalho de conexão, o navegador sabe que o servidor pretende fechar a conexão depois da transferência; desse modo, o navegador é proibido de enviar outras requisições. As próximas seções descrevem as finalidades de outros cabeçalhos.

25.12 Negociação

Além de especificar detalhes sobre um documento sendo enviado, o HTTP usa cabeçalhos para permitir a um cliente e a um servidor *negociarem* capacidades. O conjunto de capacidades negociáveis inclui uma grande variedade de características sobre a conexão (por exemplo, se o acesso for autenticado), representação (por exemplo, se imagens gráficas no formato JPEG são aceitáveis ou quais tipos de compactação podem ser usados), conteúdo (por exemplo, se os arquivos de texto precisam estar em inglês) e controle (por exemplo, o período de tempo em que uma página permanece válida).

Existem dois tipos básicos de negociação: *controlada pelo servidor* e *controlada pelo agente* (ou seja, controlada por navegador). A negociação controlada pelo servidor começa com uma requisição de um navegador. A requisição especifica uma lista de preferências junto com o URL do documento desejado. O servidor seleciona, dentre as representações disponíveis, uma que satisfaça às preferências do navegador. Se vários documentos satisfizerem tais preferências, o servidor usa uma política local para selecionar uma. Por exemplo, se um documento for armazenado em vários idiomas e uma requisição especificar uma preferência pelo inglês, o servidor enviará a versão em inglês.

A negociação controlada pelo agente significa que um navegador usa um processo em duas etapas para realizar a seleção. Primeiro, ele envia uma requisição ao servidor para perguntar o que está disponível. O servidor retorna uma lista de possibilidades. O navegador seleciona uma das possibilidades e envia uma segunda requisição para obter o documento. A desvantagem da negociação controlada pelo agente é que ela exige duas interações do servidor; a vantagem, por sua vez, é que um navegador retém controle completo sobre a escolha.

Um navegador usa um cabeçalho HTTP *Accept* para especificar qual mídia ou representações são aceitáveis. O cabeçalho lista nomes de formatos com um valor de preferência atribuído a cada um. Por exemplo,

```
Accept: text/html, text/plain; q=0.5, text/x-dvi;
                                q=0.8
```

especifica que o navegador deseja aceitar o tipo de mídia *text/html*, mas, se este não existir, o navegador aceitará *text/x-dvi*, e, se esse não existir, *text/plain*. Os valores numéricos associados à segunda e à terceira

entradas podem ser imaginados como um nível de preferência, no qual nenhum valor é equivalente a *q=1*, e um valor *q=0* significa que o tipo é inaceitável. Para tipos de mídia em que a "qualidade" é significativa (por exemplo, áudio), o valor de *q* pode ser interpretado como um desejo de aceitar determinado tipo de mídia, se for o melhor disponível após outras formas serem reduzidas em qualidade por *q* por cento.

Existem diversos cabeçalhos *Accept* que correspondem aos cabeçalhos *Content* descritos anteriormente. Por exemplo, um navegador pode enviar qualquer um dos seguintes para especificar quais codificações, conjuntos de caracteres e idiomas deseja aceitar:

```
Accept-Encoding:
Accept-Charset:
Accept-Language:
```

Podemos resumir a discussão sobre negociação conforme descrito a seguir.

O HTPP usa cabeçalhos tipo MIME para transportar metainformações. Navegadores e servidores enviam cabeçalhos que lhes permitem negociar acordo sobre a representação do documento e a codificação a ser utilizada.

25.13 Requisições condicionais

O HTTP permite que um emissor crie uma requisição *condicional*. Ou seja, quando um navegador envia uma solicitação, ele inclui um cabeçalho que qualifica as condições sob as quais a requisição deve ser honrada. Se a condição especificada não for atendida, o servidor não retorna o documento requisitado. Requisições adicionais possibilitam que um navegador otimize a recuperação evitando transferências desnecessárias. Uma das condições mais úteis usa uma requisição *If-Modified-Since*, que permite que um navegador evite transferir um documento a menos que este tenha sido atualizado desde uma data especificada. Por exemplo, um navegador pode incluir o cabeçalho

```
If-Modified-Since: Mon, 01 Apr 2013 05:00:01 GMT
```

com uma requisição *GET* para evitar uma transferência se o item for mais anterior a 1º de abril de 2005.

25.14 Servidores proxy e cache

Os servidores proxy são uma parte importante da arquitetura web, pois oferecem uma otimização que pode diminuir a latência e reduzir a carga sobre os servidores. Existem duas formas de servidores proxy: *não transparentes* e *transparentes*. Como o nome sugere, um servidor *não transparente* é visível a um usuário – o usuário configura um navegador para contatar o proxy no lugar da origem. Um proxy *transparente* não exige quaisquer mudanças na configuração de um navegador. Em vez disso, ele examina todas as conexões TCP que passam pelo proxy e intercepta qualquer conexão para a porta 80. De qualquer forma, um proxy coloca no cache as páginas web e responde a requisições subsequentes para uma página a partir do cache.

O HTTP inclui suporte explícito para servidores proxy. O protocolo especifica exatamente como um proxy trata de cada requisição, como os cabeçalhos devem ser interpretados pelos proxies, como um navegador negocia com um proxy e como este negocia com um servidor. Além do mais, vários cabeçalhos HTTP foram projetados especificamente para serem usados por proxies. Por exemplo, um cabeçalho permite que um proxy se autentique para um servidor, e outro permite a cada proxy que trata de uma página web registrar sua identidade, de modo que o destinatário final receba uma lista de todos os proxies intermediários. Finalmente, o HTTP permite que um servidor controle como os proxies lidam com cada página web. Por exemplo, um servidor pode incluir o cabeçalho *Max-Forwards* em uma resposta para limitar o número de proxies que lidam com uma página antes que ele seja entregue a um navegador. Se o servidor especificar uma contagem de um, como em

```
Max-Forwards: 1
```

no máximo um proxy pode tratar do item ao longo do caminho do servidor até o navegador. Um contador zero proíbe qualquer proxy de tratar da página.

25.15 Cache

O objetivo do uso de cache é melhorar a eficiência: um cache reduz tanto a latência quanto o tráfego da rede, eliminando transferências desnecessárias. O aspecto mais óbvio do uso da técnica de cache é o armazenamento. Assim, quando uma página web é acessada inicialmente, uma cópia é armazenada em disco, seja pelo navegador, seja por um proxy intermediário ou por ambos. Novas solicitações subsequentes da mesma página podem reduzir o processo de pesquisa e apanhar uma cópia da página web a partir do cache, em vez do servidor.

Uma pergunta central em todos os esquemas de uso de cache se refere ao tempo: "por quanto tempo um item deve ser mantido em um cache?". Por um lado, manter uma cópia no cache por muito tempo resulta em uma cópia velha, significando que as mudanças no original não são refletidas na cópia em cache. Por outro lado, se a cópia no cache não for mantida por tempo suficiente, a eficiência diminui, pois a próxima requisição terá de voltar ao servidor.

O HTTP permite que um servidor controle o cache de duas maneiras. Primeiro, quando ele responde a uma requisição de página, um servidor pode especificar detalhes do cache, incluindo se a página pode ser colocada no cache, se um proxy pode colocar a página no cache, se a comunidade com a qual a cópia em cache pode ser compartilhada, a hora em que a cópia em cache deverá expirar e limites sobre transformações que podem ser aplicadas à cópia. Em segundo lugar, o HTTP permite que um navegador force a *revalidação* de uma página. Para fazer isso, o navegador envia uma requisição à página e usa um cabeçalho para especificar que a *idade* máxima (ou seja, a hora desde que uma cópia da página foi armazenada) não pode ser maior que zero. Nenhuma cópia da página em um cache pode ser usada para satisfazer à requisição, pois a cópia terá uma idade diferente de zero. Assim, somente o servidor original responderá à requisição. Proxies intermediários ao longo do caminho receberão uma cópia nova para seu cache, assim como o navegador que emitiu a requisição.

Podemos resumir conforme descrito a seguir.

O uso de cache é a chave para a operação eficiente da web. O HTTP permite que os servidores web controlem se e como uma página pode ser colocada em cache além do seu tempo de vida; um navegador pode forçar uma requisição a uma página a fim de evitar caches e obter uma

cópia nova do servidor que possui a página.

25.16 Outras funcionalidades do HTTP

Nossa descrição do HTTP focalizou exclusivamente a recuperação em que um cliente, normalmente um navegador, emite uma requisição *GET* para apanhar uma cópia de uma página web de um servidor. Porém, o HTTP inclui facilidades que permitem interações mais complexas entre um cliente e servidor. Em particular, o HTTP fornece métodos *PUT* e *POST* que permitem que um cliente envie dados a um servidor. Assim, é possível montar um script que peça ID e senha de um usuário e depois transfira os resultados ao servidor.

É surpreendente que, embora permita a transferência em qualquer direção, o protocolo HTTP subjacente permaneça sem estado (ou seja, não exige que uma conexão da camada de transporte persistente continue ativa durante uma interação). Assim, informações adicionais normalmente são usadas para coordenar uma série de transferências. Por exemplo, em resposta a um ID e senha, um servidor poderia enviar um inteiro de identificação, conhecido como cookie, que o cliente retornaria em transferências sucessivas.

25.17 HTTP, segurança e e-commerce

Embora defina um mecanismo que possa ser usado para acessar páginas web, o HTTP não oferece segurança. Assim, antes que façam compras na Web as quais exijam a transferência de informações como números de cartão de crédito, os usuários precisam de garantias de que a transação é segura. Existem dois aspectos relevantes: confidencialidade dos dados sendo transferidos e autenticação do site que oferece os itens para venda. Conforme veremos no Capítulo 29, a criptografia é usada para garantir a confidencialidade. Além disso, um mecanismo de certificado pode ser usado para autenticar o comerciante.

Uma tecnologia de segurança foi criada para ser usada com as transações na web. Conhecida como *HTTP over SSL* (*HTTPS*), a tecnologia executa HTTP sobre o protocolo *Secure Socket Layer* (*SSL*) e soluciona os dois problemas de segurança relacionados ao e-commerce: por estarem criptografadas, as transferências de dados são confidenciais e, como o SSL usa uma árvore de certificado, um comerciante é autenticado.

25.18 Resumo

A World Wide Web é composta por documentos hipermídia armazenados em um conjunto de servidores web e acessados por navegadores. Cada documento recebe um URL que o identifica exclusivamente; o URL especifica o protocolo usado para apanhar o documento, o local do servidor e o caminho até o documento nesse servidor.

A HyperText Markup Language, HTML, permite não só que um documento contenha texto junto com comandos embutidos que controlam a formatação, mas também que um documento contenha links para outros documentos.

Um navegador e servidor utilizam o HyperText Transfer Protocol, HTTP, para transferir informação. O HTTP é um protocolo em nível de aplicação, com suporte explícito para negociação, servidores proxy, cache e conexões persistentes. Uma tecnologia relacionada, conhecida como HTTPS, utiliza SSL para oferecer comunicação HTTP segura.

EXERCÍCIOS

25.1 Leia o padrão para URLs. O que um sinal # seguido por uma string significa no final de um URL?

25.2 Estenda o exercício anterior. É válido enviar o sufixo # em um URL para um servidor web? Sim ou não? Por quê?

25.3 Como um navegador distingue entre um documento que contém HTML e um documento que contém um texto qualquer? Para descobrir, experimente usando um navegador para ler de um arquivo. Um navegador usa o nome do arquivo ou o conteúdo para decidir como interpretar o arquivo?

25.4 Qual é a finalidade de um comando HTTP *TRACE*?

25.5 Qual é a diferença entre um comando HTTP *PUT* e um comando HTTP *POST*? Quando cada um deles é útil?

25.6 Quando é usado um cabeçalho HTTP *Keep-Alive*?

25.7 Um servidor web qualquer pode funcionar como um proxy? Para descobrir, escolha um servidor web qualquer e configure seu navegador para usá-lo como um proxy. Os resultados o surpreendem?

25.8 Baixe e instale o cache de proxy transparente Squid. Quais facilidades de rede no sistema operacional o Squid utiliza para o cache de páginas web?

25.9 Leia sobre a diretiva de controle de cache *must-revalidate* do HTTP. Dê um exemplo de uma página web que usaria essa diretiva.

25.10 Se um navegador não envia um cabeçalho HTTP *Content-Length* antes de uma requisição, como um servidor responde?

25.11 Suponha que você trabalhe para uma empresa que configura o seu laptop sempre usando o servidor proxy da web da empresa. Explique o que acontece se você viajar e se conectar à Internet em um hotel.

25.12 Leia mais sobre HTTPS e explique o seu impacto sobre o uso de cache. Em que circunstâncias um proxy pode colocar páginas web em cache ao usar HTTPS?

25.13 Leia cuidadosamente a especificação HTTP. Pode ela ser usada para transmissão de vídeo? Sim ou não? Por quê?

25.14 Considere um ataque de navegação de serviço em um servidor web no qual um agressor providencia, para ter muitos clientes, uma forma de conexão com o servidor e envia repetidamente solicitações para páginas web inexistentes. Como esse tipo de ataque pode ser evitado?

25.15 Como muitas páginas da web incluem anúncios, na maioria delas há pelo menos algum conteúdo dinâmico (ou seja, o conteúdo que é gerado quando a página é carregada). Como deve um web designer organizar tais páginas para maximizar a eficácia de cache?

---

* Um URL é um tipo específico do *Uniform Resource Identifier* (URI) mais genérico.

** Parte da literatura refere-se à string inicial, *http*, como uma *pragmática*.

* O padrão usa o termo orientado a objeto método no lugar de comando.

* Nota do Revisor Técnico: eventualmente o servidor web provê suporte à língua nacional, permitindo optar por mensagens de erro na língua local.

* Veja o capítulo anterior para uma discussão sobre e-mail e MIME.

CAPÍTULO 26

Voz e vídeo sobre IP (RTP, RSVP, QoS)

CONTEÚDOS DO CAPÍTULO

26.1 Introdução

Os capítulos anteriores consideraram aplicativos que transferem mensagens de correio eletrônico e arquivos de dados. Este capítulo foca a transferência de dados em tempo real, como voz e vídeo, sobre uma rede IP. Além de discutir os protocolos usados para transportar esses dados, o capítulo considera dois aspectos amplos. Primeiro, ele examina protocolos e tecnologias necessários para o serviço de telefonia IP comercial. Segundo, examina a questão de como os roteadores em uma rede IP podem assegurar qualidade de serviço suficiente para fornecer reprodução de áudio e vídeo de alta qualidade.

26.2 Digitalização e codificação

Antes que a voz ou o vídeo possam ser enviados através de uma rede de pacotes, um hardware conhecido como *codificador/decodificador* (*codec*) precisa ser usado a fim de converter o sinal analógico para a forma digital. O tipo mais comum de codec, um codificador de forma de onda (*waveform coder*), mede a amplitude do sinal de entrada em intervalos regulares e converte cada amostra em um valor digital (ou seja, um inteiro).* No lado receptor, um codec aceita uma sequência de inteiros como entrada e recria um sinal analógico contínuo que corresponde aos

valores digitais.

Existem vários padrões de codificação digital, sendo que a troca é entre a qualidade de reprodução e o tamanho da representação digital. Por exemplo, o sistema telefônico convencional usa o padrão *Pulse Code Modulation* (PCM), que especifica tirar uma amostra de 8 bits a cada 125 μ segundos (ou seja, 8.000 vezes por segundo). Como resultado, uma chamada telefônica digitalizada produz dados em uma taxa de 64 Kbps. A codificação PCM produz uma quantidade surpreendente de saída – armazenar um clipe de áudio de 128 segundos requer um megabyte de memória.

Existem três maneiras de reduzir a quantidade de dados gerados pela codificação digital: tirar menos amostras por segundo, usar menos bits para codificar cada amostra ou, então, usar um esquema de compactação digital para reduzir o tamanho da saída resultante. Há vários esquemas que usam uma ou mais dessas técnicas, possibilitando encontrar produtos que geram áudio codificado em uma taxa de apenas 2,2 Kpbs. Entretanto, cada técnica possui desvantagens. A principal desvantagem de tirar menos amostras ou usar menos bits para codificar uma amostra é a menor qualidade de áudio – o sistema não pode reproduzir uma faixa tão grande de sons. A principal desvantagem da compactação é o retardo – a saída digitalizada precisa ser mantida enquanto é compactada. Além disso, como a maior redução no tamanho requer mais processamento, a melhor compactação exige uma CPU mais rápida ou introduz um retardo mais longo. Portanto, a compactação torna-se mais útil quando o retardo não é importante (por exemplo, ao armazenar a saída de um codec em um arquivo).

26.3 Transmissão e reprodução em áudio e vídeo

Muitas aplicações de áudio e vídeo são classificadas como tempo real porque exigem transmissão e distribuição apropriadas.* Por exemplo, as chamadas telefônicas interativas e os vídeos são classificados como em tempo real, pois áudios e vídeos devem ser entregues sem atraso significativo, ou os usuários julgarão o resultado insatisfatório. A transferência apropriada significa retardo mais do que baixo, pois o sinal resultante é ininteligível, a menos que seja apresentado exatamente na mesma ordem do original e exatamente com o mesmo tempo. Portanto, se um emissor tirar uma amostra a cada 125 μ segundos, o receptor precisa converter valores digitais em analógicos exatamente na mesma velocidade em que foram amostrados.

Como uma rede pode garantir que o fluxo (stream) seja distribuído

exatamente na mesma taxa que o emissor usou? O sistema telefônico convencional dos Estados Unidos trouxe uma resposta: uma arquitetura *isócrona*. No projeto isócrono, o sistema inteiro, incluindo o circuito digital, precisa ser projetado para proporcionar a saída com exatamente o mesmo intervalo de tempo que foi utilizado para gerar a entrada. Desse modo, um sistema isócrono com vários caminhos entre dois pontos deve ser concebido para que todos os caminhos tenham exatamente o mesmo atraso.

A tecnologia TCP/IP e a Internet global não são isócronas. Já vimos que os datagramas podem ser duplicados, retardados ou chegar fora de ordem. A variação no retardo, conhecida como *jitter*, é especialmente insidiosa em redes IP. Para permitir transmissão e reprodução significativa de sinais digitalizados através de uma rede com semântica IP, é necessário suporte de protocolo adicional. A fim de tratar a duplicação de datagrama e a entrega fora de ordem, cada transmissão precisa conter um número sequencial. E, para tratar o jitter, cada transmissão precisa conter uma *estampa de tempo* (timestamp) que informe ao receptor em que horário os dados no pacote devem ser reproduzidos. Separar informações de sequência e sincronização permite que um receptor reconstrua o sinal com precisão, independentemente de como os pacotes chegam. Essas informações de sincronização são especialmente importantes quando um datagrama é perdido ou se o emissor para de codificar durante períodos de silêncio; elas permitem que o receptor pause durante a reprodução da quantidade de tempo especificado pelas marcas de hora. Para resumir a seguir em poucas palavras.

Como uma internet IP não é isócrona, é necessário suporte de protocolo adicional ao enviar dados em tempo real, tais como áudio e vídeo. Além das informações de sequência básicas que permitem a detecção de pacotes duplicados ou reordenados, cada pacote precisa transportar uma estampa de tempo separada que diz ao receptor a hora exata em que os dados no pacote devem ser executados.

26.4 Jitter e retardo de reprodução

Como um receptor pode recriar um sinal precisamente se a rede introduzir jitter? O receptor precisa implementar um *buffer de reprodução*

(*playback buffer*),** como mostra a Figura 26.1.

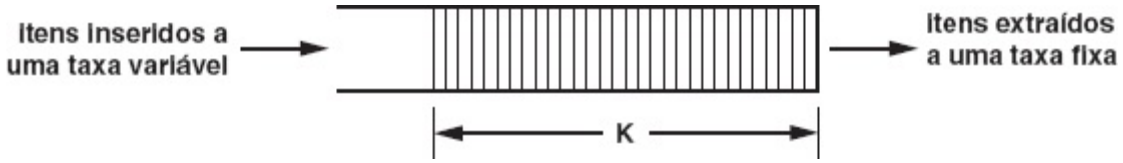


Figura 26.1 A organização conceitual de um buffer de reprodução que compensa o jitter. O buffer armazena K unidades de tempo dos dados.

Quando uma sessão se inicia, o receptor retarda a reprodução e coloca os dados que chegam ao buffer. Logo que os dados no buffer alcançam um limite predeterminado, conhecido como *ponto de reprodução* (*playback point*), a saída inicia. O ponto de reprodução, rotulado como *K* na figura, é medido em unidades de tempo dos dados a serem reproduzidos. Assim, a reprodução se iniciará quando um receptor tiver acumulado *K* unidades de tempo dos dados.

Os aplicativos que desempenham transmissão de áudio e vídeo, em geral, apresentam aos usuários uma representação gráfica de buffer de reprodução. Tipicamente, a tela é composta por uma barra horizontal que representa o tempo necessário para apresentar o objeto. Por exemplo, se um usuário reproduz o vídeo para um programa de televisão de trinta minutos, a tela representa o tempo de zero a trinta minutos. A qualquer momento, o sombreamento é utilizado para dividir a barra em três segmentos. O que fica do lado esquerdo mostra a quantidade de vídeo que foi reproduzido; o segmento seguinte mostra a quantidade do vídeo não reproduzido que foi baixado; o terceiro segmento, por sua vez, mostra a quantidade de vídeo que ainda tem de ser baixado. Usamos o termo *ponto de reprodução* para nos referir ao ponto no vídeo sendo exibido no momento presente, e o termo *ponto de download* como referência à quantidade de vídeo baixado no momento atual. A Figura 26.2 mostra como uma visualização de reprodução pode aparecer para um usuário com os segmentos rotulados.

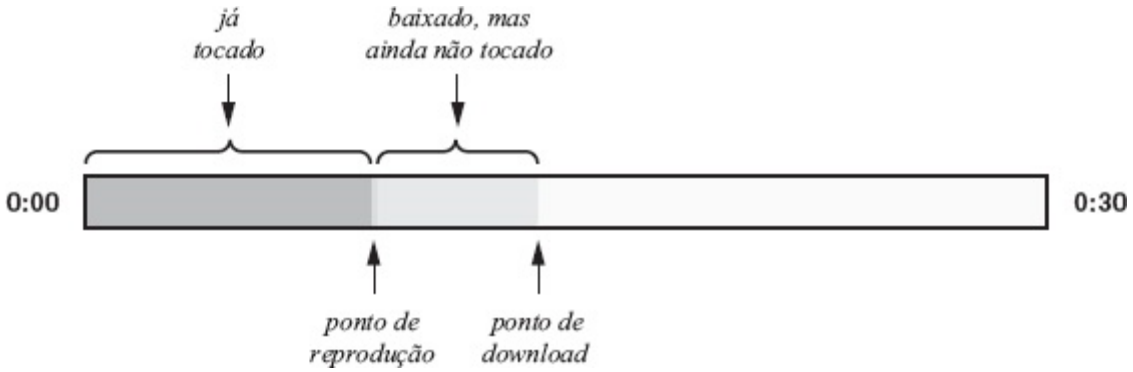


Figura 26.2 Ilustração de um display que mostra o buffer de reprodução para um streaming de vídeo de 30 minutos com segmentos e pontos marcados.

Conforme a reprodução prossegue, os datagramas continuam a chegar. Se não houver jitter, novos dados chegarão exatamente na mesma taxa em que os antigos dados estão sendo extraídos e executados, significando que o buffer sempre conterá exatamente K unidades de tempo dos dados não executados. Se um datagrama experimentar um pequeno retardo, a reprodução não será afetada. O tamanho do buffer diminui gradualmente à medida que os dados são extraídos, e a reprodução continua ininterrupta por K unidades de tempo. Quando um datagrama retardado chega, o buffer é preenchido novamente.

É claro que um buffer de reprodução não pode compensar perda de datagrama. Nesses casos, a reprodução eventualmente atinge uma posição não preenchida no buffer. Na figura, o ponto de reprodução atinge o ponto de download. Quando a reprodução esgota todos os dados disponíveis, a saída deve parar pelo período de tempo correspondente aos dados em falta.

A escolha de K é uma conciliação entre perda e retardo.* Se K for muito pequeno, uma pequena quantidade de jitter fará com que o sistema esgote o buffer de reprodução antes que os dados necessários cheguem. Se K for muito grande, o sistema permanecerá imune ao jitter, mas o retardo extra, quando adicionado ao retardo de transmissão na rede subjacente, pode ser perceptível aos usuários. Apesar das desvantagens, a maioria das aplicações que enviam dados em tempo real através de uma internet IP depende do buffer de reprodução como a principal solução para o jitter.

26.5 Real-time Transport Protocol (RTP)

O protocolo usado para transmitir sinais de áudio ou vídeo digitalizados

por uma internet IP é conhecido como *Real-time Transport Protocol* (*RTP*). Interessantemente, no RTP não há mecanismos que garantam que os pacotes atravessem uma internet em tempo hábil. Essas garantias, se elas existem, precisam ser dadas pelo sistema subjacente. Em vez disso, o RTP fornece dois recursos principais: um número sequencial em cada pacote, que permite que um receptor detecte distribuição fora de ordem ou perda, e uma estampa de tempo, que permite a um receptor controlar a reprodução.

Como o RTP é projetado para transportar uma grande variedade de dados em tempo real, incluindo áudio e vídeo, ele não impõe uma codificação específica para dados. Em vez disso, cada pacote começa com um cabeçalho; os campos iniciais no cabeçalho especificam como interpretar os campos de cabeçalho restantes e também o payload. A Figura 26.3 ilustra o formato do cabeçalho do RTP.

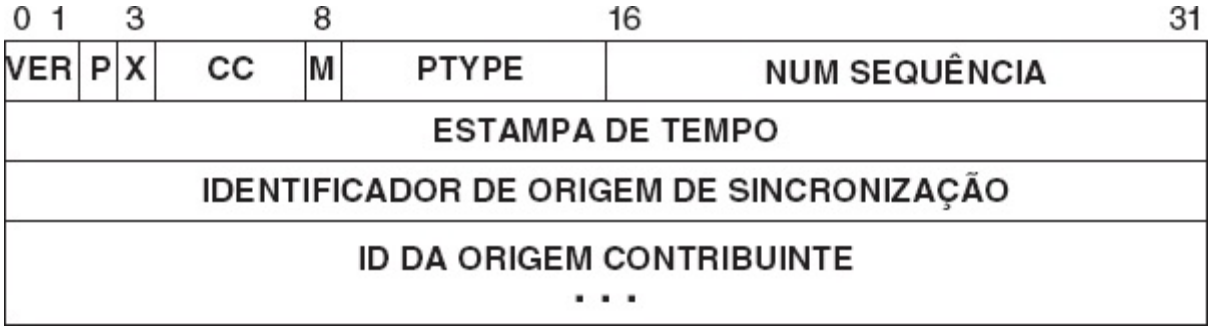


Figura 26.3 Ilustração do cabeçalho fixo usado com o RTP. Cada mensagem começa com seu cabeçalho; a interpretação exata e os campos de cabeçalho adicionais dependem do tipo de payload, PTYPE.

Como mostra a figura, cada pacote começa com um número de versão RTP de dois bits no campo VER; a versão atual é 2. O bit *P* especifica se o preenchimento de zero segue o payload; ele é usado com a criptografia que requer a alocação dos dados em blocos de tamanho fixo. Algumas aplicações definem uma extensão de cabeçalho opcional a ser colocada entre o cabeçalho mostrado acima e o payload. Se o tipo de aplicação permitir uma extensão, o bit *X* é usado para especificar se a extensão está presente no pacote. O campo CC de quatro bits contém uma contagem que é contribuição dos Ids de origem no cabeçalho. A interpretação do bit *M* ("marcador") também depende da aplicação; ele é usado por aplicações que precisam marcar pontos no fluxo de dados (por exemplo, o início de cada frame ao enviar vídeo). O campo PTYPE sete bits especifica o tipo de

payload a ser enviado na mensagem; a interpretação dos demais campos no cabeçalho e no payload depende do valor em PTYPE. O campo NUM DE SEQUÊNCIA de 16 bits contém um número sequencial para o pacote. O primeiro número sequencial em uma determinada sessão é escolhido aleatoriamente.

O tipo de payload afeta a interpretação do campo ESTAMPA DE TEMPO (TIMESTAMP). No plano conceitual, uma *estampa de tempo* é um valor de 32 bits que informa a hora em que foi feita a amostra do primeiro octeto dos dados digitalizados, com a estampa de tempo inicial para uma sessão escolhida aleatoriamente. O padrão especifica que a estampa de tempo seja incrementada continuamente, mesmo durante períodos em que nenhum sinal é detectado e nenhum valor é enviado, mas não especifica a granularidade exata. Em vez disso, a granularidade é determinada pelo tipo de payload, o que significa que cada aplicação pode escolher uma granularidade de relógio que permita a um receptor posicionar itens na saída com a precisão apropriada para a aplicação. Por exemplo, se um fluxo de dados de áudio está sendo transmitido por RTP, uma granularidade de estampa de tempo de uma subdivisão de clock por amostra é apropriada.* Desse modo, uma estampa de tempo de áudio pode ter uma granularidade de um tick para cada 125 μ segundos. Quando um fluxo contém dados de vídeo, uma amostra pode corresponder a um frame. Entretanto, uma granularidade de um tick por frame será indesejável – uma granularidade maior alcançará uma reprodução mais suave.

A separação do número de sequência e da estampa do tempo é importante para os casos em que a amostra abrange vários pacotes. Em particular, o padrão permite à estampa de tempo em dois pacotes ser idêntica no caso em que tais pacotes contenham os dados que foram amostrados no mesmo tempo.

26.6 Fluxos, mistura e multicasting

Uma parte importante do RTP é seu suporte para *tradução* ou *mistura*. A tradução refere-se a alterar a codificação de um fluxo em uma estação intermediária (por exemplo, reduzir a resolução de uma transmissão de vídeo antes de enviá-lo para um telefone celular). Mistura refere-se ao processo de receber fluxos de dados a partir de múltiplas fontes, combinando-os em um único fluxo, e enviar o resultado. Para entender a necessidade da mistura, imagine que pessoas em diversos locais participem de uma teleconferência usando IP. Visando minimizar o número de fluxos de RTP, o grupo pode designar um mixer e cuidar para que cada local

estabeleça uma sessão RTP com ele. O *mixer* combina os fluxos de áudio (possivelmente convertendo-os de volta para analógicos e fazendo a reamostragem do sinal resultante) e envia o resultado como um único fluxo digital.

Os campos no cabeçalho RTP suportam mistura, indicando que ela ocorreu e identificando as fontes de dados. O campo na figura 26.3, rotulado como IDENTIFICADOR DE ORIGEM DE SINCRONIZAÇÃO (SYNCHRONIZATION SOURCE IDENTIFIER) especifica a origem de um fluxo. Cada origem precisa escolher um identificador de 32 bits único; o protocolo inclui um mecanismo para resolver conflitos, caso surjam. Quando um mixer combina vários fluxos, ele se torna a origem de sincronização para o novo fluxo. No entanto, as informações sobre as origens iniciais não são perdidas, pois o mixer usa o campo ID DE ORIGEM CONTRIBUINTE (CONTRIBUTING SOURCE ID) para fornecer os IDs de sincronização dos fluxos que foram mixados juntos. O campo CC de quatro bits fornece uma contagem das origens contribuintes, o que significa que quinze origens, no máximo, podem ser listadas.

O RTP é projetado para operar com multicasting IP, e a mistura torna-se especialmente atrativa em um ambiente multicast. Para entender por que, imagine uma teleconferência que inclua muitos participantes. O unicasting exige que uma estação envie a cada participante uma cópia de cada pacote RTP que sai. Entretanto, com o multicasting, uma estação só precisa enviar uma cópia do pacote, que será distribuída para todos os participantes. Além disso, se a mistura for usada, todas as origens podem difundir para um mixer, que as combina em um único fluxo antes do multicasting. Portanto, a combinação de mistura e multicasting resulta em substancialmente menos datagramas sendo distribuídos a cada host participante.

26.7 Encapsulamento de RTP

O nome implica que o RTP é um protocolo em nível de transporte. Sem dúvida, se funcionasse como um protocolo de transporte convencional, o RTP exigiria que cada mensagem fosse encapsulada diretamente em um datagrama IP. De fato, o nome é inadequado porque o RTP não funciona como um protocolo de transporte.* Ou seja, o encapsulamento direto de mensagens de RTP em datagramas IP não ocorre na prática. Em vez disso, o RTP é executado sobre o UDP, significando que cada mensagem RTP é encapsulada em um datagrama UDP. A principal vantagem de usar UDP é a concorrência – um único computador pode ter múltiplas aplicações usando RTP sem interferência.

Diferentemente de muitos dos protocolos de aplicação, o RTP não usa o número de porta UDP reservado. Em vez disso, uma porta é alocada para uso com cada seção, e a aplicação remota precisa ser informada sobre o número de porta. Por convenção, o RTP escolhe uma porta UDP de número par; a seção a seguir explica que um protocolo associado, o RTCP, usa o próximo número sequencial de porta.

26.8 RTP Control Protocol (RTCP)

Nossa descrição da transmissão em tempo real focou os mecanismos de protocolo que permitem a um receptor reproduzir conteúdo. Outro aspecto da transmissão em tempo real é igualmente importante: o monitoramento da rede subjacente durante a sessão e o fornecimento de comunicação fora de banda entre as extremidades. Esse mecanismo é especialmente importante nos casos em que esquemas de codificação adaptados são usados. Por exemplo, uma aplicação pode escolher uma codificação de largura de banda inferior quando a rede subjacente se torna congestionada, ou um receptor pode variar o tamanho do seu buffer de reprodução quando o retardo de rede ou jitter é alterado. Finalmente, um mecanismo de fora de banda pode ser usado para enviar informações em paralelo com os dados em tempo real (por exemplo, legendas para acompanhar o fluxo de vídeo).

Um protocolo associado e parte integral do RTP, conhecido como *RTP Control Protocol* (*RTCP*), fornece a funcionalidade de controle necessária. O RTCP permite que emissores e receptores transmitam uns aos outros uma série de relatórios que contenham informações adicionais sobre os dados sendo transferidos e o desempenho da rede. As mensagens RTCP são encapsuladas em UDP para transmissão* e enviadas usando o número de porta subsequente ao número de porta do fluxo RTP ao qual elas pertencem.

26.9 Operação do RTCP

A Figura 26.4 relaciona os cinco tipos de mensagem básicos que o RTCP usa para permitir que emissores e receptores troquem informações sobre uma sessão.

Tipo	Significado
200	Relatório do emissor
201	Relatório do receptor
202	Mensagem de descrição de origem
203	Mensagem de despedida

Figura 26.4 Os cinco tipos de mensagem RTCP. Cada mensagem começa com um cabeçalho fixo que identifica o tipo.

As suas últimas mensagens da lista são mais fáceis de entender. Um emissor transmite uma mensagem de despedida (*bye*) quando encerra um fluxo e uma mensagem *específica a aplicação* para definir um novo tipo de mensagem. Por exemplo, para enviar informações de legenda junto com um fluxo de vídeo, uma aplicação poderia escolher definir um novo tipo de mensagem RTCP.

Receptores periodicamente transmitem mensagens de *relatório do receptor* que informam a origem sobre as condições da recepção. Os relatórios do receptor são importantes por duas razões. Primeiro, as mensagens permitem que cada receptor participe em uma sessão, além do emissor, para aprender sobre as condições de recepção dos outros receptores. Segundo, as mensagens permitem aos os receptores adaptarem sua frequência de geração de relatório para evitar usar largura de banda excessiva e sobrecarregar o emissor. O esquema adaptativo garante que o tráfego de controle total permanecerá menos de 5% do tráfego de dados em tempo real, e que os relatórios do receptor gerarão menos de 75% do tráfego de controle. Cada relatório do receptor identifica uma ou mais origens de sincronização e contém uma seção separada para cada origem. Uma seção especifica o pacote de número sequencial mais alto recebido da origem, a perda de pacote cumulativa e percentual experimentada, o tempo decorrido desde que o último relatório RTCP chegou da origem e o jitter entre as chegadas.

Os emissores transmitem periodicamente uma mensagem *relatório do emissor*, que fornece uma estampa de tempo absoluta. Para entender a necessidade de uma estampa de tempo, lembre-se de que o RTP permite que cada fluxo escolha uma granularidade para sua estampa de tempo e de que a primeira estampa de tempo é escolhida aleatoriamente. A estampa de tempo absoluta em um relatório do emissor é essencial porque ela fornece o único mecanismo que um receptor possui para *sincronizar* múltiplos fluxos. Em especial, como o RTP exige um fluxo separado para cada tipo de mídia, a transmissão associada do vídeo e do áudio requer dois fluxos. A informação de estampa de tempo absoluta permite que um receptor execute

os dois fluxos simultaneamente.

Além das mensagens de notificação periódicas do emissor, os emissores também transmitem mensagens de *descrição de origem*, que fornecem informações gerais sobre o usuário que possui ou controla a origem. Cada mensagem contém uma seção para cada fluxo RTP que sai; o conteúdo é destinado à leitura de humanos. Por exemplo, o único campo necessário consiste em um *nome canônico* para o possuidor do fluxo, uma string de caracteres na forma

usuario@host

em que *host* é o nome de domínio do computador ou seu endereço IP na forma decimal ou hexadecimal pontuada, e *usuário* (user) é um nome de login. Os campos opcionais na descrição de origem contêm detalhes adicionais, como o endereço de e-mail do usuário (que pode diferir do nome canônico), o número de telefone, a localização geográfica da posição, o aplicativo ou a ferramenta usada para criar o fluxo, ou outras notas textuais sobre a origem.

26.10 Telefonia e sinalização IP

Um aspecto da transmissão em tempo real se destaca como especialmente importante: o uso do IP como a base para o serviço telefônico. Conhecida como *telefonia IP* ou *Voice over IP* (VoIP), a abordagem é agora usada por muitas empresas de telefonia. Surge, então, a seguinte pergunta: "que tecnologias adicionais são necessárias antes de VoIP poder substituir completamente o sistema telefônico isócrono existente?". Embora não exista uma resposta simples, três componentes básicos são necessários. Primeiro, vimos que um protocolo como RTP é necessário ao transferir dados em tempo real através de uma internet IP. O protocolo define cada amostra com uma estampa de tempo que permite a um receptor recriar um sinal de saída analógico, que corresponde exatamente ao sinal de entrada original. Segundo, um mecanismo é necessário para estabelecer e terminar chamadas telefônicas. Terceiro, os pesquisadores estão explorando meios em que uma internet IP pode ser criada para operar como uma rede isócrona.

O setor de telefonia usa o termo *sinalização* (*signaling*) para se referir ao processo de estabelecer uma chamada telefônica. Especificamente, o mecanismo de sinalização usado na rede pública de telefonia comutada

(*Public Switched Telephone Network* – PSTN) é o sistema de sinalização 7 (*Signaling System 7* – SS7). O SS7 realiza roteamento de chamada antes de qualquer áudio ser enviado. Dado um número telefônico, o SS7 forma um circuito através da rede, toca o telefone designado e conecta o circuito quando o telefone é respondido. Além disso, o SS7 também trata detalhes como o encaminhamento de chamada e as condições de erro, como o fato de telefone de destino estar ocupado.

Antes de o IP ser usado para fazer chamadas telefônicas, a funcionalidade de sinalização precisa estar disponível. Além disso, para permitir a adoção pelas companhias telefônicas, a telefonia IP precisa ser compatível com o sistema de sinalização utilizado pela telefonia existente– ele precisa ser capaz de interoperar com o sistema de telefonia convencional em todos os níveis. Portanto, é necessário traduzir entre a sinalização usada com o IP e o SS7, bem como traduzir entre a codificação de voz usada com o IP e a codificação PCM padrão. Consequentemente, os dois mecanismos de sinalização terão funcionalidade equivalente.

O método geral para a interoperabilidade utiliza um *gateway* entre o sistema de telefonia IP e o sistema telefônico convencional. Uma chamada pode ser iniciada em qualquer lado do gateway. Quando uma requisição de sinalização chega, o gateway a traduz e a encaminha; além disso, ele também precisa traduzir e encaminhar a resposta. Finalmente, após a sinalização estar completa e uma chamada ter sido estabelecida, o gateway precisa encaminhar a voz nos dois sentidos, traduzindo da codificação usada em um lado para a codificação usada no outro.

Dois grupos propuseram padrões para a telefonia IP. O ITU definiu um conjunto de protocolos conhecido como *Hs323* e o IETF propôs um protocolo de sinalização conhecido como *Session Initiation Protocol* (*SIP*). As seções a seguir resumem os dois métodos.

26.10.1 Padrões H.323

O ITU, originalmente, criou o H.323 para permitir a transmissão da voz sobre tecnologias de rede local. O padrão foi estendido para permitir a transmissão da voz sobre redes IP, e as companhias telefônicas o têm adotado. O H.323 não é um protocolo único; ele especifica como vários protocolos podem ser combinados para formar um sistema de telefonia IP funcional. Por exemplo, além dos gateways, o H.323 define dispositivos conhecidos como *gatekeepers*, que fornecem, cada um, um ponto de

contato para telefones que utilizam IP. Para obter permissão para estabelecer chamadas que saem e permitir que o sistema telefônico direcione chamadas que entram para o destino correto, cada telefone IP precisa se registrar com um gatekeeper; o H.323 inclui os protocolos necessários.

Além de especificar um protocolo para a transmissão de voz e vídeo em tempo real, o frame H.323 fornece protocolos que especificam como os participantes transferem e compartilham dados. De relevante significado é a partilha de dados relativos a uma teleconferência em tempo real. Por exemplo, um par de usuários engajados em uma áudio vídeo conferência também pode compartilhar uma prancheta na tela, enviar imagens estáticas ou trocar cópias de documentos.

A Figura 26.5 lista os quatro principais protocolos que formam os blocos de construção de H.323.

Protocolo	Finalidade
H.225.0	Sinalização usada para estabelecer uma chamada
H.245	Controle e feedback durante a chamada
RTP	Transferência de dados em tempo real (sequência e sincronização)
T.120	Troca de dados associados a uma chamada

Figura 26.5 Os quatro protocolos básicos usados pelo H.323 para telefonia IP.

Tomados juntos, os protocolos do conjunto cobrem todos os aspectos da telefonia IP, incluindo registro de telefone, sinalização, codificação e transferência de dados em tempo real (voz e vídeo) e controle.

A Figura 26.6 ilustra as relações entre os protocolos que constituem o H.323. Como a figura mostra, o conjunto inteiro, em última análise, depende de UDP e TCP sendo executados sobre IP.

Figura 26.6 Relação entre os principais protocolos que constituem o padrão de telefonia IP H.323 do ITU. Os protocolos que estão omitidos tratam de detalhes como segurança e transmissão de fax.

26.10.2 Session Initiation Protocol (SIP)

O IETF propôs uma alternativa para o H.323, chamada *Session Initiation Protocol* (*SIP*), que cobre apenas sinalização; ele não recomenda codecs específicos nem exige o uso do RTP para transferência em tempo real. Portanto, o SIP não fornece toda a funcionalidade do H.323.

O SIP usa interação cliente-servidor, com servidores sendo divididos em dois tipos. Um *servidor agente de usuário* é executado em um telefone SIP. A cada servidor do agente de usuário é atribuído um identificador (por exemplo, *usuario®site*), permitindo o recebimento de chamadas. O segundo tipo de servidor é um *servidor intermediário* colocado entre dois telefones SIP, que trata de tarefas como configuração e encaminhamento de chamadas. Um servidor intermediário funciona como um servidor proxy, que pode encaminhar uma requisição de chamada entrando para o próximo servidor proxy, que pode encaminhar um pedido de chamada ao longo do caminho para o próximo servidor proxy ao longo do caminho ou para o telefone chamado. Um servidor intermediário pode também funcionar como um servidor de redirecionamento, que diz a um chamador como alcançar o destino.

Para fornecer informações sobre uma chamada, o SIP se baseia em um protocolo associado, o *Session Description Protocol* (*SDP*). O SDP é especialmente importante em uma chamada de conferência porque permite aos participantes entrarem e deixarem a chamada dinamicamente. O SDP também especifica detalhes como a codificação de mídia, os números de porta de protocolo e o endereço multicast.

26.11 Controvérsia da qualidade de serviço

O termo *Qualidade de Serviço* (*Quality of Service – QoS*) refere-se às garantias de desempenho estatístico que um sistema de rede pode dar com relação a perda, retardo, vazão e jitter. Uma rede isócrona que é projetada para atender a limites de desempenho estritos fornece garantias de QoS, enquanto uma rede de comutação de pacotes que usa distribuição de melhor esforço não fornece qualquer garantia de QoS. A QoS necessária

para transferência em tempo real de voz e vídeo está sobre IP? Se sim, como ela deve ser implementada? Uma importante controvérsia ronda as duas perguntas. Por um lado, os engenheiros que projetaram o sistema de telefonia insistem que a reprodução de voz de qualidade toll exige que o sistema subjacente forneça garantias de QoS quanto a retardo e perda para cada chamada telefônica. Por outro, os engenheiros que projetaram o IP insistem que a Internet funciona razoavelmente bem sem as garantias de QoS e que acrescentar QoS por fluxo é inviável porque os roteadores tornarão o sistema caro e lento.

A controvérsia da QoS gerou muitas propostas, implementações e experiências. Embora a Internet funcione sem QoS, ela já costuma enviar áudio e vídeo. Provedores comerciais oferecem serviços de telefonia IP, e as empresas de telefonia em todo o mundo estão migrando para IP. Muitos esforços para oferecer QoS não obtiveram êxito. Por exemplo, o modo de transferência assíncrona (*Asynchronous Transfer Mode – ATM*), que foi criado por companhias telefônicas como uma alternativa para a Internet, quase desapareceu. O ATM tentou fornecer garantias de QoS para cada conexão individual (ou seja, cada fluxo). Após um esforço chamado *Serviços Integrados* (*Integrated Services – IntServ*), que investigou a definição da qualidade de serviço por fluxo, o IETF mudou de direção e adotou o conservador *Serviços Diferenciados* (*Differentiated Services – DiffServ*), que divide o tráfego em classes separadas. Esse esquema de serviços diferenciados, que sacrifica o controle fino para encaminhamento menos complexo, algumas vezes é chamado de método de *Classe de Serviço* (*Class of Service – CoS*).

26.12 QoS, utilização e capacidade

O debate sobre a QoS é remanescente das discussões anteriores sobre alocação de recursos, como aquelas travados sobre políticas de sistema operacional para alocação de memória e programação de processador. Nos debates anteriores, os defensores argumentam que uma melhor alocação de recursos seria otimizar o rendimento global de um sistema de computação, oferecendo aos usuários, desse modo, um serviço melhor. O argumento tem um apelo intuitivo, e muita pesquisa foi realizada. Mas, infelizmente, nenhum dos sistemas de gestão de processador e memória funcionava bem na prática. Os usuários permaneciam insatisfeitos. No entanto, após décadas, a computação melhorou, e os usuários ficaram satisfeitos com os

resultados. O que mudou? Os processadores tornaram-se muito mais rápidos, e as memórias, muito maiores. Em vez de depender de algoritmos de escalonamento para encontrar a melhor maneira de compartilhar um processador lento, entre muitos cálculos, o hardware tornou-se tão rápido que um processador poderia acompanhar o cálculo requerido.

A analogia com o sistema de rede é forte. Os defensores da QoS estão lançando mão do mesmo argumento sobre recursos de rede. Eles afirmam que, se os recursos da rede forem programados de forma eficaz (ou seja, a rede dá alguns pacotes de prioridades sobre outros), os usuários estarão felizes. A premissa é especialmente atraente para os operadores de rede porque, se verdadeira, lhes permitirá vender serviços atualizados com a infraestrutura básica existente. Infelizmente, a experiência com a rede revela a constatação a seguir.

> *Quando a rede tem recursos suficientes para todo o tráfego, restrições de QoS são desnecessárias; quando o tráfego excede a capacidade da rede, nenhum sistema de QoS pode satisfazer as demandas de todos os usuários.*

O problema central é a utilização. Por um lado, uma rede com 1% de utilização não precisa de QoS porque nenhum pacote é mesmo bloqueado. Por outro lado, uma rede cuja utilização excede 100% de capacidade falhará sob qualquer QoS. No entanto, os defensores de mecanismos de QoS alegam que mecanismos de QoS sofisticados devem ser capazes de alcançar dois objetivos. Primeiro, dividindo os recursos existentes entre mais usuários, eles tornam o sistema mais "justo"; segundo, modelando o tráfego de cada usuário, eles permitem que a rede opere em uma taxa de utilização mais alta, sem o risco de um colapso.

Uma das principais razões para o fato de os mecanismos de QoS mais complicados não terem sido mais adotados foram os aumentos no desempenho das redes. A capacidade de rede aumentou dramaticamente durante os últimos trinta anos, e ela continuará a crescer num futuro previsível. Enquanto rápidos aumentos de desempenho possibilitam a capacidade de exceder a demanda, os mecanismos de QoS meramente representam uma sobrecarga desnecessária. Entretanto, se a demanda surgir mais rapidamente do que a capacidade, a QoS pode se tornar um problema econômico – associando preços maiores a níveis de serviço mais altos, os provedores da Internet podem usar o custo para racionar a capacidade (e

colher lucros mais altos porque não será necessário nenhum aumento de infraestrutura).

26.13 Serviços de emergência e direito de preferência

Há razões válidas para priorizar o tráfego? Claro que sim. Por exemplo, se o tráfego de gerenciamento de rede tem baixa prioridade, um administrador pode não ser capaz de diagnosticar a origem do congestionamento ou mesmo de tomar medidas para corrigir o problema. Da mesma forma, considere uma rede que lide com serviços de telefonia VoIP, na qual chamadas de emergência (911 nos EUA) não devem ser bloqueadas à espera de tráfego normal. Assim, os pacotes que carregam amostras de voz a partir de uma chamada de emergência devem ter prioridade.

Pode parecer que os exemplos apresentem um forte argumento para um mecanismo de QoS que compreende o propósito de cada fluxo. Como será explicado mais adiante, não se faz necessário um mecanismo por fluxo. As chamadas de emergência e de tráfego de gerenciamento de rede podem ser manipuladas por um sistema que tem apenas dois ou três níveis de prioridade. Mais importante, um mecanismo de QoS que apenas garanta uma percentagem da capacidade da rede subjacente pode não funcionar se a capacidade for reduzida. Por exemplo, considere um sistema de QoS que reserve 1% da capacidade de rede para tráfego de gerenciamento, o que, em situações normais, pode ser mais do que suficiente. No entanto, considere o que acontece se uma falha de hardware de rede começa a corromper uns poucos bits nos pacotes aleatoriamente. A garantia de QoS implica que, depois de um pacote de gerenciamento de rede ser enviado, noventa e nove outros pacotes serão enviados antes de outro pacote de gerenciamento de rede ser enviado (por exemplo, uma retransmissão). Assim, o tempo entre sucessivos pacotes de gestão de rede é longo. Portanto, se sucessivos pacotes de gestão estão corrompidos, pode ser longo o tempo necessário para diagnosticar o problema.

O necessário para lidar com casos como o descrito anteriormente não é um sistema de QoS típico, mas um esquema de prioridade absoluta. Em particular, precisamos de um mecanismo que permita ao tráfego de emergência prevenir a rede. Sob uma política de preempção, ao tráfego de emergência é concedida a mais absoluta prioridade, significando que, se um pacote de transporte de tráfego de emergência chega, este é enviado imediatamente, sem esperar outro tráfego.

26.14 IntServ e reserva de recursos

Duas décadas atrás, o IETF começou a considerar a questão da alocação de recursos. Especificamente, a IETF começou com a pergunta: Se QoS é necessário, como pode uma rede IP fornecê-lo? Na época, muitos grupos defenderam o QoS de grão fino, o que levou o IETF a um programa de pesquisa chamado Serviços Integrados (*Integrated Services – IntServ*). A abordagem IntServ apresenta duas partes. Primeira: antes que os dados sejam transferidos, as extremidades precisam enviar uma requisição que especifique os recursos necessários, e todos os roteadores ao longo do caminho entre as extremidades precisam concordar em fornecer os recursos; o procedimento pode ser visto como uma forma de sinalização. Segunda: à medida que os datagramas atravessam o fluxo, os roteadores precisam monitorar e controlar o encaminhamento de tráfego. O monitoramento, às vezes chamado de *policiamento de tráfego*, é necessário para garantir que o tráfego enviado em um fluxo não exceda os limites especificados.

Garantias de QoS são especialmente difíceis em uma rede de comutação de pacotes porque o tráfego é muitas vezes em rajadas. Por exemplo, um fluxo que especifica uma vazão média de 1 Mbps pode ter 2 Mbps de tráfego para dez milissegundos seguido de nenhum tráfego para dez milissegundos. Embora o roteador deva lidar com uma inundação de pacotes para dez milissegundos, o fluxo ainda atende à média exigida. Para controlar fila e encaminhamento, um roteador que oferece QoS geralmente implementa um mecanismo a fim de lidar com rajadas de pacotes. A ideia, conhecida como *modelador de tráfego*, é suavizar cada rajada. Para isso, um roteador enfileira temporariamente datagramas de entrada e os envia a uma taxa constante de 1 Mbps.

26.14.1 Resource ReSerVation Protocol (RSVP)

Como parte do trabalho IntServ, o IETF desenvolveu dois protocolos para fornecer QoS: o *Resource ReSerVation Protocol* (*RSVP*), para reservar recursos, e o *Common Open Policy Services* (*COPS*)* para impor restrições. Ambos exigem mudanças na infraestrutura básica da Internet – todos os roteadores precisam concordar em reservar recursos (por exemplo, capacidade do link) para cada fluxo entre duas extremidades. O RSVP cuida de requisições e respostas de reserva. Ele não é um protocolo de propagação de rota, nem impõe políticas quando um fluxo é estabelecido. Em vez disso, o RSVP opera antes que quaisquer dados sejam enviados. Para iniciar um fluxo ponto a ponto, uma extremidade primeiro envia uma

mensagem de caminho RSVP de modo a determinar o caminho até o destino; o datagrama transportando a mensagem usa a opção roteador alerta (IPv4) ou um cabeçalho salto a salto especial (IPv6) para garantir que os roteadores examinem a mensagem. Após receber uma resposta à sua mensagem de caminho, a extremidade envia uma mensagem requisição *RSPV* para reservar recursos ao fluxo. A requisição especifica os limites de QoS desejados; cada roteador que encaminha a requisição para o destino precisa concordar em reservar os recursos que a requisição especifica. Se qualquer roteador no caminho negar a requisição, o roteador usa o RSVP para enviar uma resposta negativa de volta à origem. Se todos os sistemas ao longo do caminho concordarem em atender à requisição, o RSVP retorna uma resposta positiva.

Cada fluxo RSVP é simplex (isto é, unidirecional). Se uma aplicação exigir garantias de QoS nos dois sentidos, cada extremidade precisa usar RSVP para requisitar um fluxo. Como o RSVP utiliza encaminhamento existente, não há qualquer garantia de que os dois fluxos atravessarão os mesmos roteadores, nem a aprovação de um fluxo em um dos sentidos implica a aprovação do fluxo no outro sentido. Podemos fazer o resumo a seguir.

Uma extremidade usa RSVP para requisitar um fluxo simplex através de uma internet IP com limites de QoS especificados. Se cada roteador ao longo do caminho concordar em aceitar a requisição, o fluxo será aprovado; caso contrário, o fluxo será negado. Se um par de aplicações precisar de QoS nos dois sentidos, cada extremidade precisará usar RSVP para requisitar um fluxo separado.

26.14.2 Imposição do IntServ (COPS)

Quando chega uma requisição RSVP, um roteador precisa considerar dois aspectos: viabilidade (ou seja, se o roteador possui os recursos necessários para satisfazer à requisição) e política (ou seja, se a requisição se encaixa dentro das restrições políticas). A viabilidade é uma decisão local – um roteador pode decidir como gerenciar a largura de banda, a memória e a capacidade de processamento disponível localmente. Entretanto, a imposição de política exige cooperação global – todos os roteadores precisam concordar com o mesmo conjunto de políticas.

Para implementar políticas globais, a arquitetura IETF usa um modelo

de dois níveis, com interação cliente-servidor entre os níveis. Quando um roteador recebe uma requisição RSVP, ele se torna um cliente que consulta um servidor conhecido como ponto de decisão política (*Policy Decision Point – PDP*) para determinar se a requisição atende a restrições políticas. O PDP não trata o tráfego; ele meramente avalia requisições para ver se elas satisfazem a políticas globais. Se um PDP aprovar uma requisição, o roteador precisa atuar como um ponto de aplicação de política (*Policy Enforcement Point – PEP*) para garantir que o tráfego se ajuste à política aprovada.

O protocolo COPS define a interação cliente-servidor entre um roteador e um PDP (ou entre um roteador e um PDP local se a organização tiver múltiplos níveis servidores de política). Embora o COPS defina seu próprio cabeçalho de mensagem, o formato básico compartilha muitos detalhes com o RSVP. Em especial, o COPS usa o mesmo formato do RSVP para itens individuais em uma mensagem requisição. Portanto, quando um roteador recebe uma requisição RSVP, ele pode extrair itens relacionados à política, colocá-los em uma mensagem COPS e enviar o resultado para um PDP.

26.15 DiffServ e comportamento por salto

Após muito trabalhar no RSVP e no IntServ, o IETF decidiu seguir uma abordagem totalmente diferente: em vez de olhar para as tecnologias que fornecem QoS para cada fluxo individual, o novo trabalho concentra-se em grupos de fluxos. Assim, um pequeno conjunto de categorias é criado, e cada fluxo é atribuído a uma das categorias. O resultado é o DiffServ,* que difere do IntSErv em dois aspectos significativos. Primeiro, em vez de especificar os recursos necessários para um fluxo individual, o DiffServ aloca serviço para uma *classe* (por exemplo, um conjunto de fluxos que corresponde a um determinado conjunto de parâmetros). Segundo, diferentemente do esquema RSVP, em que uma reserva é feita de fim a fim, o DiffServ permite que cada nó no caminho defina o serviço que uma determinada classe receberá. Por exemplo, um roteador pode escolher dividir largura de banda de modo que a classe DiffServ conhecida como *Expedited Forwarding* (*EF*) receba 50% da largura de banda e os 50% restantes sejam divididos entre as classes que são conhecidas como *Assured* Forwarding (AF). O próximo roteador no caminho pode escolher dar à classe EF 90% da largura de banda e dividir as classes AF entre os 10% restantes. Usamos a expressão “comportamento por salto” para

descrever o método e para enfatizar que o DiffServ não fornece garantias fim a fim.

26.16 Escalonamento de tráfego

Para implementar qualquer forma de QoS, um roteador precisa atribuir prioridades para o tráfego que sai e escolher que pacote enviar em um dado momento. O processo de selecionar entre um conjunto de pacotes é conhecido como *escalonamento de tráfego*, e o mecanismo é chamado de *scheduler de tráfego*. Há quatro aspectos a serem considerados ao construir um scheduler que são descritos a seguir.

- *Justiça.* O scheduler deve garantir que os recursos (ou seja, largura de banda) consumidos por um fluxo estejam dentro da quantidade atribuída para esse fluxo.**
- *Retardo.* Os pacotes em um determinado fluxo não devem ser retardados excessivamente.
- *Adaptabilidade.* Se um determinado fluxo não tiver pacotes para enviar, o scheduler deve dividir a largura de banda extra entre outros fluxos proporcionalmente aos seus recursos atribuídos.
- *Overhead computacionais.* Como opera no caminho rápido, um scheduler não pode incorrer em muito overhead computacional. Em especial, os algoritmos teóricos, como *Generalized Processor Scheduling* (*GPS*), não podem ser usados.

O esquema de programação de tráfego prático mais simples chama-se *Weighted Round Robin* (*WRR*), pois atribui um peso a cada fluxo e tenta enviar dados do fluxo de acordo com o peso dele. Por exemplo, podemos imaginar três fluxos, *A*, *B* e *C*, com pesos 2, 2 e 4, respectivamente. Se todos os três fluxos tiverem pacotes esperando para serem enviados, o scheduler deve enviar através do fluxo *C* (peso 4) o dobro do que através do fluxo *A* ou *B* (cada um com peso 2).

Talvez pareça que um scheduler WRR poderia atingir os pesos desejados selecionando dos fluxos na seguinte ordem:

C C A B

Ou seja, o scheduler faz seleções repetidamente:

C C A B C C A B C C A B ...

O padrão parece alcançar os pesos desejados porque metade das seleções vem do fluxo *C*, um quarto vem de *B* e um quarto vem de *A*. Além disso, o padrão serve cada fila em intervalos regulares durante toda a sequência, significando que nenhum fluxo é retardado desnecessariamente (isto é, a frequência com que os pacotes são enviados de um determinado fluxo é constante).

Embora a sequência acima faça a taxa de pacotes igualar os pesos atribuídos, a abordagem WRR não atinge o objetivo de tornar as taxas de dados coincidentes com os pesos porque os datagramas não têm tamanho uniforme. Por exemplo, se o tamanho médio do datagrama sobre o fluxo *C* é metade do tamanho médio do datagrama do fluxo *A*, selecionar o fluxo *C* duas vezes mais que o fluxo *A* vai tornar igual a taxa de dados dos dois fluxos.

Para resolver o problema, foi criado um algoritmo modificado que acomoda os pacotes de tamanho variável. Conhecido como *Deficit Round Robin (DRR)*, o algoritmo calcula os pesos em função do total de octetos enviados, e não pelo número de pacotes. Inicialmente, o algoritmo aloca um número de octetos para cada fluxo proporcionalmente à largura de banda que o fluxo deve receber. Quando um fluxo é selecionado, o DRR transmite o máximo de pacotes possível sem exceder o número predeterminado de octetos. O algoritmo, então, calcula o resto (ou seja, a diferença entre o número de octetos que foi alocado e o tamanho dos pacotes efetivamente enviados) e o adiciona à quantidade que será enviada na próxima vez. Assim, o DRR mantém um total atualizado do "déficit" que cada fluxo deve receber. Mesmo que o déficit obtido em uma determinada rodada seja pequeno, o valor crescerá através de várias rodadas até que seja grande o bastante para acomodar um pacote extra. Portanto, ao longo do tempo, a proporção dos dados que o DRR envia de um determinado fluxo atinge o valor ponderado para esse fluxo.

Algoritmos de programação Round Robin, como o WRR e o DRR, apresentam vantagens e desvantagens. A principal vantagem decorre da eficiência: uma vez que os pesos tenham sido atribuídos, pouca computação é necessária para fazer uma seleção de pacote. Na verdade, se todos os pacotes forem do mesmo tamanho e os pesos forem selecionados como

múltiplos, a seleção ponderada pode ser obtida por meio do uso de um array, em vez de computação.

Apesar das vantagens, os algoritmos round robin possuem pontos negativos. Primeiro, o retardo que um determinado fluxo experimenta depende do número dos outros fluxos que possuem tráfego para enviar. Na pior hipótese, um determinado fluxo pode precisar esperar enquanto o scheduler envia um ou mais pacotes de cada um dos outros fluxos. Segundo, como enviam uma rajada de pacotes de uma determinada fila e, depois, retardam enquanto servem outras filas, os algoritmos round robin podem produzir jitter.

26.17 Policiamento e conformação de tráfego

Um *controlador de tráfego* é necessário para verificar se o tráfego que chega não excede um perfil estatístico declarado. Imagine que um scheduler aloca 25% da largura de banda de saída para uma classe DiffServ, *Q*. Se três fluxos de entrada mapearem para a classe *Q*, estes competirão pela largura de banda alocada para *Q*. Se o sistema não monitora o tráfego de entrada, um dos fluxos pode levar toda a largura de banda alocada para a classe *Q*. Assim, um mecanismo de policiamento protege outros fluxos para garantir que nenhum deles receba mais do que sua justa fatia.

Vários mecanismos foram propostos para o policiamento de tráfego. Em geral, a ideia é diminuir o tráfego a uma taxa acordada, ou seja, a ideia de conformação de tráfego antes mencionada. Um dos primeiros mecanismos de tráfego de modelagem, baseado no método *leaky bucket*, usa um contador para controlar a taxa de pacote. Conceitualmente, o algoritmo incrementa periodicamente o contador; cada vez que um pacote chega, o algoritmo diminui o contador. Se este se tornar negativo, significa que o fluxo que entra excedeu sua taxa de pacote alocada.

Usar uma taxa de pacotes para moldar o tráfego não faz sentido na Internet, porque datagramas variam em tamanho. Portanto, esquemas de policiamento mais sofisticados foram propostos para acomodar pacotes de tamanho variável. Por exemplo, um mecanismo de *token bucket* estende o método descrito antes fazendo o contador corresponder aos bits em vez de aos pacotes. O contador é aumentado periodicamente de acordo com a taxa de dados desejada e diminuído pelo número de bits de cada pacote que chega.

Na prática, os mecanismos de policiamento descritos não exigem um

timer para atualizar periodicamente um contador. Em vez disso, cada vez que um pacote chega, o policial examina o relógio para determinar quanto tempo decorreu desde a última vez que o fluxo foi processado e usa o período de tempo para calcular um incremento para o contador. Calcular um incremento tem menos sobrecarga computacional, além de tornar o policiamento de tráfego mais eficiente.

26.18 Resumo

Dados em tempo real consistem de áudio ou de vídeo em que a reprodução de uma amostra deve corresponder ao tempo em que esta foi capturada. Um dispositivo de hardware chamado codec codifica dados analógicos, como áudio, na forma digital. O padrão telefônico para codificação de áudio, o Pulse Code Modulation (PCM), produz valores digitais em 64 Kbps; outras codificações sacrificam alguma fidelidade para obter taxas de bits mais altas.

O RTP é usado para transferir dados em tempo real através de uma rede IP. Cada mensagem RTP contém duas peças-chave de informação: um número sequencial e uma estampa de tempo de mídia. Um receptor usa o número de sequência para colocar mensagens em ordem e detectar datagramas perdidos e usa uma estampa de tempo para determinar quando executar os valores codificados. Um protocolo de controle associado, o RTCP, é usado para fornecer informações sobre origens e para permitir que um mixer combine vários fluxos. Para acomodar rajadas e jitter, um aplicativo que reproduz dados em tempo real usa um buffer de reprodução e apresenta um ligeiro atraso antes de executar um item.

Há serviços de telefonia IP comerciais que usam a tecnologia VoIP; a maioria das empresas de telefonia, no entanto, está mudando para IP. Dois modelos foram criados para uso com telefonia IP: o ITU criou o modelo H.323, e o IETF, o SIP.

Um debate continua sobre se as garantias de Qualidade de Serviço (Quality of Service – QoS) são necessárias para fornecer serviços em tempo real. Inicialmente, o IETF seguia um programa conhecido como Serviços Integrados (Integrated Services – IntServ), que explorava QoS por fluxo. Mais tarde, o IETF decidiu se mudar para uma abordagem de Serviços Diferenciados (Differentiated Services – DiffServ), que fornece QoS entre as classes de fluxos em vez de fluxos individuais.

A implementação da QoS requer um mecanismo de programação de tráfego para selecionar pacotes de filas que saem e um policiamento de tráfego para monitorar fluxos que chegam. Como é computacionalmente

eficiente e trata pacotes de tamanho variável, o algoritmo Deficit Round Robin está entre os mais práticos para tráfego scheduling. O algoritmo de leaky bucket é um dos mais pragmáticos para o policiamento de trânsito e modelagem.

EXERCÍCIOS

26.1 Leia sobre o Real-Time Streaming Protocol (RTSP). Quais são as principais diferenças entre o RTSP e o RTP?

26.2 Descubra como o serviço telefônico de voz Skype opera. Como se configura uma conexão?

26.3 Os operadores de rede têm um ditado: você sempre pode comprar mais largura de banda, mas não pode comprar o menor atraso. O que isso quer dizer?

26.4 Se uma mensagem RTP chegar com um número sequencial muito maior que a sequência expedida, o que o protocolo faz? Por quê?

26.5 Considere um vídeo gravado pelo seu telefone celular e transferido através da Internet em tempo real. Qual é a capacidade necessária? (Dica: o que é a resolução da câmera em seu telefone celular?)

26.6 Considere uma chamada para uma conferência via telefone que usa RTP para conectar N usuários. Dê duas implementações possíveis que podem alcançar a chamada.

26.7 Um filme geralmente tem dois fluxos conceituais: um de vídeo e de áudio. Como o RTP pode ser utilizado para transferir um filme?

26.8 Os números sequenciais são necessários no RTP ou uma estampa de tempo pode ser usada em seu lugar? Explique.

26.9 Um engenheiro insiste que "SIP é para as crianças; adultos usam H.323". O que o engenheiro quer dizer? Imagine o tipo de empresa em que o engenheiro trabalha.

26.10 Quando VoIP foi introduzido pela primeira vez, alguns países decidiram tornar a tecnologia ilegal. Descubra por quê.

26.11 Você preferiria uma Internet na qual a QoS fosse obrigatória para todo tráfego? Justifique sua resposta.

26.12 Meça a utilização em sua conexão com a Internet. Se todo tráfego necessitasse de reserva de QoS, o serviço seria melhor ou pior? Explique.

26.13 Suponha que você seja convidado a criar classes DiffServ para um cabo ISP. A rede do ISP, que utiliza apenas o IPv6, deve lidar com:

canais de transmissão televisiva, voz (ou seja, VoIP), download de filmes, serviço de Internet residencial e transmissão de vídeo sob demanda (on demand). Como você especifica classes DiffServ? Por quê?

26.14 Se a entrada de um modelador de tráfego é extremamente em rajadas (ou seja, rajadas esporádicas de pacotes com muitos longos períodos de nenhum tráfego), a saída pode não ser constante. Qual a técnica que pode ser usada para garantir a saída suave? (Dica: considere um método descrito no capítulo anterior.)

---

* Uma alternativa conhecida como *codificador/decodificador de voz* (*vocodec*) reconhece e codifica a fala humana em vez de formas de onda gerais.

* A oportunidade é mais importante do que a confiabilidade; não há tempo para a retransmissão – dados que não chegam a tempo devem ser ignorados.

** Um buffer de reprodução também é chamado buffer de jitter.

* Embora o retardo de rede e o jitter possam ser usados para determinar um valor para K dinamicamente, muitos esquemas de buffer de reprodução usam uma constante.

* O campo ESTAMPA DE TEMPO às vezes é chamado de ESTAMPA DE TEMPO DA MÍDIA, a fim de enfatizar que sua granularidade depende do tipo de sinal sendo medido.

* O nome protocolo de transferência em tempo real (*Real-time Transfer Protocol*) seria mais apropriado.

* Como algumas mensagens são curtas, o padrão permite que múltiplas mensagens sejam combinadas em um único datagrama UDP para transmissão.

* O nome COPS é entendido como uma referência bem-humorada à polícia de trânsito (polícia de tráfego).

* O Capítulo 7 descreve o mecanismo DiffServ e o campo de cabeçalho do datagrama que ele usa.

** Em toda esta seção, discutiremos a programação entre fluxos; o leitor deve entender que, quando DiffServ é usado, as mesmas técnicas são usadas para realizar o schedule de tráfego entre as classes.

CAPÍTULO 27

Gerenciamento de rede (SNMP)

CONTEÚDOS DO CAPÍTULO

27.1 Introdução

Além dos protocolos que fornecem serviços em nível de rede e programas aplicativos que utilizam esses serviços, é necessário um subsistema que permita a um administrador configurar uma rede, controlar roteamento, depurar problemas e identificar situações em que computadores violam políticas. A essas atividades chamamos *gerenciamento de rede*. Este capítulo aborda as ideias por trás do gerenciamento de rede TCP/IP e descreve um protocolo usado para gerenciamento de rede.

27.2 O nível dos protocolos de gerenciamento

Quando as redes de dados apareceram pela primeira vez, os designers seguiram uma abordagem de gestão utilizada em sistemas de telefonia, por meio da concepção de mecanismos de gestão especiais para a rede. Por exemplo, redes de área ampla geralmente definiam mensagens de gerenciamento como parte de seu protocolo de nível de link. Se um comutador de pacote começasse a se comportar erroneamente, o gerente de rede podia instruir um comutador de pacote vizinho a lhe enviar um pacote de *controle especial*. Um pacote de controle de entrada fazia o receptor suspender a operação normal e responder a comandos do próprio pacote. O gerente podia interrogar o comutador de pacote, examinar ou alterar rotas, testar uma das interfaces de comunicação ou reiniciar o comutador.

Quando o problema era reparado, o gerente podia instruir o comutador a retomar as operações normais. Como as ferramentas de gerenciamento eram parte do protocolo de nível inferior, os gerentes frequentemente eram capazes de controlar comutadores, mesmo que os protocolos de nível superior falhassem.

Diferentemente de uma rede remota homogênea, a Internet não possui um protocolo em nível de link único. Em vez disso, a Internet consiste em múltiplos tipos de rede física e dispositivos de diversos fornecedores. Como resultado, ela exige um novo modelo de gerenciamento de rede que ofereça três capacidades importantes. Primeiro, um único gerente deve ser capaz de controlar muitos tipos de dispositivos, incluindo roteadores IP, pontes, modems, estações de trabalho e impressoras. Segundo, como na Internet há vários tipos de redes, as entidades controladas não compartilham um protocolo comum de nível link. Terceiro, o conjunto de máquinas que um gerente controla pode se conectar a uma variedade de redes. Em particular, um gerente talvez tenha de controlar uma ou mais máquinas que não se conectam à mesma rede física que o computador do gerente. Portanto, talvez não seja possível para o gerente se comunicar com máquinas sendo controladas, a menos que o software de gerenciamento use protocolos que forneçam coletividade ponto a ponto através da Internet. Consequentemente, o protocolo de gerenciamento de rede usado com TCP/IP opera acima do nível de transporte.

Em uma rede TCP/IP, o gerente precisa examinar e controlar roteadores e outros dispositivos de rede. Como esses dispositivos se conectam a redes arbitrárias, os protocolos para o gerenciamento de rede operam no nível de aplicação e se comunicam usando protocolos em nível de transporte TCP/IP.

Projetar software de gerenciamento de rede para operar no nível de aplicação possui várias vantagens. Como os protocolos podem ser projetados sem levar em conta o hardware de rede subjacente, um conjunto de protocolos pode ser usado para todas as redes. Uma vez que protocolos podem ser projetados sem levar em conta o hardware no dispositivo gerenciado, os mesmos protocolos podem ser usados para todos os dispositivos gerenciados. Do ponto de vista de um gerente, ter um único conjunto de protocolos de gerenciamento significa uniformidade – todos os roteadores respondem exatamente ao mesmo conjunto de comandos. Além

disso, como o software de gerenciamento usa IP para comunicação, um gerente pode controlar os roteadores através de uma internet TCP/IP inteira sem ter conexão direta com cada rede ou roteador físico.

É claro, construir software de gerenciamento no nível de aplicação também tem desvantagens. Salvo se o sistema operacional, o software de IP e o software de transporte de protocolo funcionarem corretamente, o gerente pode não ser capaz de contatar um roteador que precise de gerenciamento. Por exemplo, se a tabela de encaminhamento do roteador ficar danificada, pode ser impossível corrigi-la ou reiniciar a máquina a partir de um local remoto. Caso o sistema de um roteador falhe, será impossível chegar ao programa aplicativo que implementa os protocolos de gerenciamento da internet, mesmo que o roteador ainda possa processar interrupções de hardware e encaminhar pacotes. Quando a ideia de construir gerenciamento de rede na camada de aplicação foi proposta pela primeira vez, muitos engenheiros de rede declararam que toda a técnica era falha. De fato, muitos pesquisadores de rede também levantaram sérias objeções.

27.3 Modelo arquitetônico

A pesar das desvantagens em potencial, ter software de gerenciamento de rede TCP/IP operando no nível de aplicação funcionava bem na prática. A vantagem mais significativa de colocar protocolos de gerenciamento de rede em um alto nível se torna aparente quando se considera uma internet grande, na qual o computador de um gerente não precisa se conectar diretamente a todas as redes físicas que contêm entidades gerenciadas. A Figura 27.1 ilustra um exemplo de intranet que ajuda a explicar a arquitetura de gerenciamento.

Como mostra a figura, o software cliente geralmente é executado na estação de trabalho do gerente. Cada sistema de rede participante, que pode ser um roteador ou um dispositivo de rede, executa um sistema de gerenciamento.* Na terminologia IETF, o software servidor é chamado de *agente de gerenciamento* ou simplesmente *agente*. Um gerente chama o software cliente no host local e especifica um agente com o qual se comunicar. Depois que o cliente se conecta ao agente especificado, o gerente pode solicitar que o cliente software envie consultas para obter informações ou comandos que configuram e controlam o dispositivo gerenciado.

É claro que nem todos os dispositivos em uma grande internet estão subordinados a um único gerente. A maioria dos gerentes apenas controla

dispositivos em suas posições locais; uma grande posição pode ter vários gerentes. O software de gerenciamento de rede utiliza um mecanismo de autenticação para garantir que apenas gerentes autorizados possam acessar ou controlar um determinado dispositivo. Alguns protocolos de gerenciamento aceitam múltiplos níveis de autorização, permitindo que um gerente especifique privilégios em cada dispositivo. Por exemplo, um roteador específico pode ser configurado para permitir que vários gerentes obtenham informações, mas permitir que apenas um seleto subconjunto de gerentes mude informações ou controle o roteador.

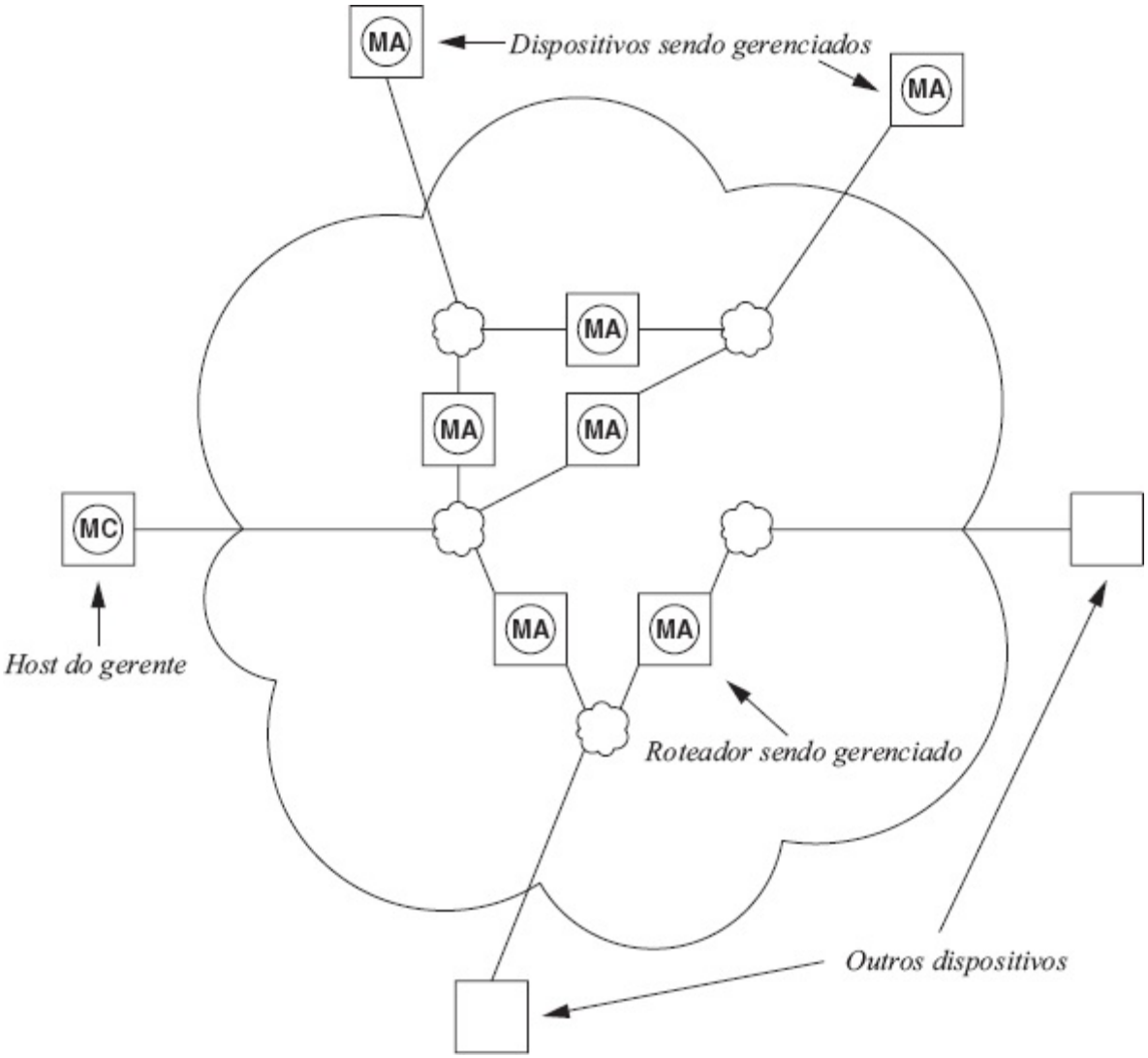


Figura 27.1 Exemplo do gerenciamento de rede em que um gerente chama o software do cliente de gerenciamento (MC) que pode contatar o software do agente de gerenciamento (MA), executado em dispositivos em toda uma intranet.

27.4 Estrutura do protocolo

Os protocolos de gerenciamento de rede TCP/IP dividem o problema do gerenciamento em duas partes e especificam padrões separados para cada parte. A primeira parte se refere à comunicação de informações entre o software cliente em execução no host de um gerente e um agente de execução em um dispositivo de gerenciamento. O protocolo define o formato e o significado das mensagens trocadas entre clientes e servidores, bem como a forma de nomes e endereços. A segunda parte relaciona-se aos dados sendo gerenciados. Veremos que o protocolo especifica um conjunto de itens de dados que um dispositivo de gerenciamento deve disponibilizar ao gerente, o nome de cada item de dados, a sintaxe usada para expressar o nome e a semântica associada ao acesso ou à modificação do item de dados.

27.4.1 O protocolo TCP/IP para gerenciamento de rede

O *Simple Network Management Protocol* (*SNMP*) é o padrão para gerenciamento de rede no conjunto de protocolos TCP/IP. O SNMP evoluiu por meio de três gerações. Consequentemente, a versão atual é conhecida como *SNMPv3*. As mudanças entre as versões têm sido relativamente pequenas – todas as três versões usam a mesma estrutura geral, e muitos recursos são compatíveis com as versões anteriores.

Além de especificar detalhes como o formato de mensagem e o uso dos protocolos de transporte, o padrão SNMP define o conjunto de operações e o significado de cada um. Veremos que o método é minimalista – poucas operações fornecem total funcionalidade. Iniciaremos examinando SNMP para IPv4; uma seção posterior resumirá as mudanças para IPv6.

27.4.2 Um padrão para informações gerenciadas

Um dispositivo sendo gerenciado mantém o controle e as informações de status que um gerente pode acessar. Por exemplo, um roteador mantém as estatísticas sobre o estado de suas interfaces de rede, juntamente com a contagem de pacotes que entram e saem, datagramas descartados e mensagens de erro geradas. Um modem mantém estatísticas sobre o número de bits (ou caracteres) enviados e recebidos, bem como sobre o estado do transportador (se o modem na outra extremidade da ligação está respondendo). Embora permita que um gerenciador acesse estatísticas, o SNMP não especifica exatamente quais dados podem ser acessados em quais dispositivos. Em vez disso, um padrão separado especifica os detalhes para cada tipo de dispositivo. Conhecido como *Management*

Information Base (*MIB*), o padrão especifica os itens de dados que cada um dispositivo gerenciado precisa manter, as operações permitidas em cada um e o seu significado. Por exemplo, o MIB para IP especifica que o software precisa manter uma contagem de todos os octetos que chegam através de cada interface de rede e que o software de gerenciamento só pode ler a contagem.

O MIB para TCP/IP divide as informações de gerenciamento em muitas categorias. A escolha delas é importante porque os identificadores usados para especificar itens incluem um código para a categoria. A Figura 27.2 lista alguns exemplos de categorias utilizadas com IPv4.

Categoria do MIB	Inclui informações sobre
sistema	O sistema operacional do host ou roteador
interfaces	Interfaces de rede individuais
at	Tradução de endereço (por exemplo, mapeamentos ARP)
ip	Software do Internet Protocol versão 4
ipv6	Software do Internet Protocol versão 6
icmp	Software do Internet Control Message Protocol versão 4
IPV	Software do Internet Control Message Protocol versão 6
tcp6Icmp	Software do Transmission Control Protocol
udp	Software do User Datagram Protocol
ospf	Software do Open Shortest Path First
bgp	Software do Border Gateway Protocol
rmon	Monitoramento de rede remota
ri	Software do Routing Information Protocol
dnsp-2	Software do sistema de nome de domínio

Figura 27.2 Exemplo de categoria de informação MIB. A categoria é codificada no identificador utilizado para especificar um objeto.

Manter a definição MIB independente do protocolo de gerenciamento de rede traz vantagens tanto para fornecedores quanto para usuários. Um fornecedor pode incluir software agente SNMP em um produto como um roteador, com a garantia de que o software continuará a aderir ao padrão após novos itens de MIB serem definidos. Um consumidor pode usar o mesmo software cliente de gerenciamento de rede para gerenciar diversos dispositivos que tenham versões ligeiramente diferentes de um MIB. É claro, um dispositivo que não possui novos itens de MIB não pode fornecer as informações nesses itens. Entretanto, como todos os dispositivos

gerenciados usam a mesma linguagem para comunicação, cada dispositivo pode analisar uma consulta e fornecer as informações solicitadas ou enviar uma mensagem de erro explicando que o item solicitado não está disponível.

27.5 Exemplos de variáveis MIB

As primeiras versões do SNMP coletavam variáveis em um único e grande MIB, com o conjunto inteiro documentado em uma única RFC. Para evitar tornar a especificação MIB impraticável, o IETF decidiu permitir a publicação de muitos documentos MIB individuais que especifiquem as variáveis para um tipo de dispositivo específico. Como resultado, mais de 100 documentos MIBs separados foram definidos como parte do processo de padrões; eles especificam mais de 10 mil variáveis individuais. Por exemplo, agora existem RFCs separadas que especificam as variáveis MIB associadas a dispositivos como: uma ponte de hardware, um nobreak, um comutador Ethernet e um modem a cabo. Além disso, muitos fornecedores definiram variáveis MIB para seus produtos de hardware ou software específicos.

Examinar alguns dos itens de dados MIB associados aos protocolos TCP/IP ajudará a esclarecer o conteúdo. A Figura 27.3 lista variáveis MIB de exemplo, juntamente com suas categorias.

A maioria dos itens listados na Figura 27.3 tem valores numéricos – cada um pode ser armazenado em um único inteiro. Entretanto, o MIB também define estruturas mais complexas. Por exemplo, a variável MIB *ipRoutingTable* se refere a uma tabela de roteamento inteira.* Variáveis MIBs adicionais sob a tabela (não listadas na figura) definem o conteúdo na entrada da tabela do encaminhamento, e permitem que os protocolos de gerenciamento de rede referenciem uma entrada individual na tabela de encaminhamento, incluindo o prefixo, a máscara de endereço e os campos próximo salto. É claro, as variáveis MIB apresentam apenas uma definição lógica de cada item de dados – as estruturas de dados internas que um roteador utiliza podem diferir da definição MIB. Quando uma consulta chega, o software no agente no roteador é responsável por mapear entre as referências de variáveis MIB e a estrutura de dados do roteador usado para armazenar as informações.

27.6 A estrutura das informações de gerenciamento

Além dos padrões que especificam as variáveis MIB e seus significados, um padrão separado especifica um conjunto de regras usadas para definir e

identificar variáveis MIB. As regras são conhecidas como a especificação *Structure of Management Information* (*SMI*). Para manter os protocolos de gerenciamento de rede simples, o SMI impõe restrições sobre os tipos de variáveis permitidos no MIB, especifica as regras para nomear essas variáveis e cria regras para definir tipos de variável. Por exemplo, o padrão SMI inclui definições do termo *Counter* (definindo-o para ser um número inteiro no intervalo de 0 a 2^{32}-1) e o termo *InetAddress* (definindo-o para ser uma string de octetos) e especifica que a definição de variáveis MIB deve usar a terminologia. Mais importante, as regras no SMI descrevem como o MIB se refere a tabelas de valores (por exemplo, uma tabela de roteamento IPv4).

Variáveis MIB	Significado
Categoria do sistema	
sysUpTime	Tempo desde a última reinicialização.
Categoria do sistema	
ifNumber	Número de interfaces de rede.
ifMtu	Mtu para uma interface (IPv4).
ipv6IfEffectiveMtu	Mtu para uma interface (IPv6).
Categoria Ip	
ipDefaultTTL	Valor IPv4 usa como um TTL.
ipv6DefaultHopLimit	Valor IPv6 usa como limite de salto.
ipInReceives	Número de datagramas IPv4 recebidos.
ipv6IfStatsInReceives	Número de datagramas IPv6 recebidos.
ipForwDatagrams	Número de datagramas IPv4 enviados.
ipv6IfStatsOutForwDatagrams	Número de datagramas IPv6 enviados.
ipOutNoRoutes	Número de falhas de rota.
ipReasmOKs	Número de datagramas remontados.
ipFragOKs	Número de datagramas IPv4 fragmentados.
ipv6IfStatsOutFragOKs	Número de datagramas de IPv6 fragmentados.
ipRoutingTable	Tabela de roteamento IPv4.
ipv6RouteTable	Tabela de roteamento IPv6.
ipv6AddrTable	Tabela de endereço IPv6 da interface.
ipv6IfStatsTable	Estatísticas IPv6 para cada interface.
Categoria icmp	
icmpInEchos	Número de requisições de eco ICMP recebido.

Categoria tcp	
tcpRtoMin	Tempo de retransmissão mínimo para TCP.
tcpMaxConn	Máximo de conexões TCP permitidas.
tcpInSegs	Número de segmentos que o TCP recebeu.
Categoria udp	
udpInDatagrams	Número de datagramas UDP recebidos.

Figura 27.3 Exemplos de variáveis MIB e suas categorias.

27.7 Definições formais usando ASN.1

O padrão SMI especifica que todas as variáveis MIB precisam ser definidas e referenciadas usando a *Abstract Syntax Notation* 1 (*ASN.1*)* da ISO. A ASN.1 é uma linguagem formal que possui dois recursos principais: uma notação usada em documentos lidos por humanos e uma representação codificada compacta das mesmas informações usadas nos protocolos de comunicação. Nos dois casos, o uso de uma notação precisa e formal remove ambiguidades da representação e do significado. Por exemplo, em vez de dizer que uma variável contém um valor inteiro, um projetista de protocolo que usa a ASN.1 precisa declarar a forma e a faixa exatas de cada valor numérico. Essa precisão é especialmente importante quando implementações incluem computadores heterogêneos que não usam, todos eles, as mesmas representações para itens de dados.

Além de especificar o nome e o conteúdo de cada item, a ASN.1 define um conjunto de regras de codificação básicas (*Basic Encoding Rules – BER*) que especifica precisamente como codificar nomes e itens de dados em uma mensagem. Portanto, uma vez que a documentação de um MIB tenha sido expressa usando ASN.1, as variáveis podem ser traduzidas direta e mecanicamente para a forma codificada usada nas mensagens. Podemos fazer o resumo a seguir.

Os protocolos de gerenciamento de rede TCP P utilizam uma notação formal chamada ASN.1 a fim de definir nomes e tipos para varreduras na base de informações de gerenciamento. A notação precisa torna a forma e o conteúdo das variáveis inequívocos.

27.8 Estrutura e representação dos nomes de objeto MIB

Dissemos que a ASN.1 especifica como representar itens de dados e nomes. Entretanto, entender os nomes usados para variáveis MIB requer que conheçamos o namespace subjacente. Os nomes usados para variáveis MIB são tirados do namespace identificador de objeto administrado pela ISO e ITU. A ideia central por trás do namespace identificador de objeto é que ele fornece um namespace em que todos os objetos possíveis podem ser designados. O namespace não está restrito às variáveis usadas no gerenciamento de rede – ele inclui nomes para objetos arbitrários (por exemplo, cada documento padrão de protocolo internacional possui um nome).

O namespace identificador de objeto é *absoluto* (*global*), significando que os nomes são estruturados para torná-los globalmente únicos. Como a maioria dos namespaces que é grande e absoluto, o namespace identificador de objeto é hierárquico. A autoridade para partes do namespace subdivide-se em cada nível, permitindo que grupos individuais obtenham autoridade para atribuir alguns dos nomes sem consultar uma autoridade central para cada atribuição.**

A Figura 27.4 ilustra partes pertinentes da hierarquia do identificador de objeto e mostra a posição dos nós do *mgmt* e *mib* usada pelos protocolos de gerenciamento de rede TCP/IP. A raiz da hierarquia do identificador de objeto não é nomeada, mas possui três descendentes diretos gerenciados por ISO, ITU e, conjuntamente, por ISO e ITU, como o nível superior da figura ilustra. A cada nó na hierarquia é atribuído um nome textual curto e um identificador único inteiro (os seres humanos usam a string de texto para ajudar a entender os nomes de objetos; o software de computador usa o inteiro para formar uma representação compacta, codificada para uso em mensagens). A ISO alocou uma subárvore *org* para uso por outras organizações de padrões nacionais ou internacionais (incluindo as organizações de padrões dos Estados Unidos). O Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (U.S. National Institute for Standards and Technology – NIST)*** alocou uma subárvore dod sob org para o Departamento de Defesa americano. Finalmente, o IAB requisitou que o Departamento de Defesa para alocar uma subárvore da *internet* no namespace e, em seguida, alocar quatro subárvores, incluindo mgmt. A subárvore *mib* era alocada sob *mgmt*.

O nome de um objeto na hierarquia é a sequência dos rótulos numéricos nos nós ao longo de um caminho da raiz até o objeto. A sequência é escrita com pontos separando os componentes individuais. Caso seja expressa para

os seres humanos lerem, nomes textuais são usados. Quando os nomes são enviados nas mensagens, usam-se os valores numéricos. Por exemplo, o nome *1.3.6.1.2.1* indica o nó rotulado *mib*. Como eles caem sob o nó MIB, todas as variáveis MIB têm nomes que começam com os prefixos *1.3.6.1.2.1*.

Anteriormente, dissemos que o MIB agrupa variáveis em categorias. O significado exato das categorias agora pode ser explicado: elas são as subárvores do nó *mib* do namespace identificador de objeto. A Figura 27.5 ilustra a ideia mostrando os primeiros poucos nós de subárvores sob o nó *mib*.

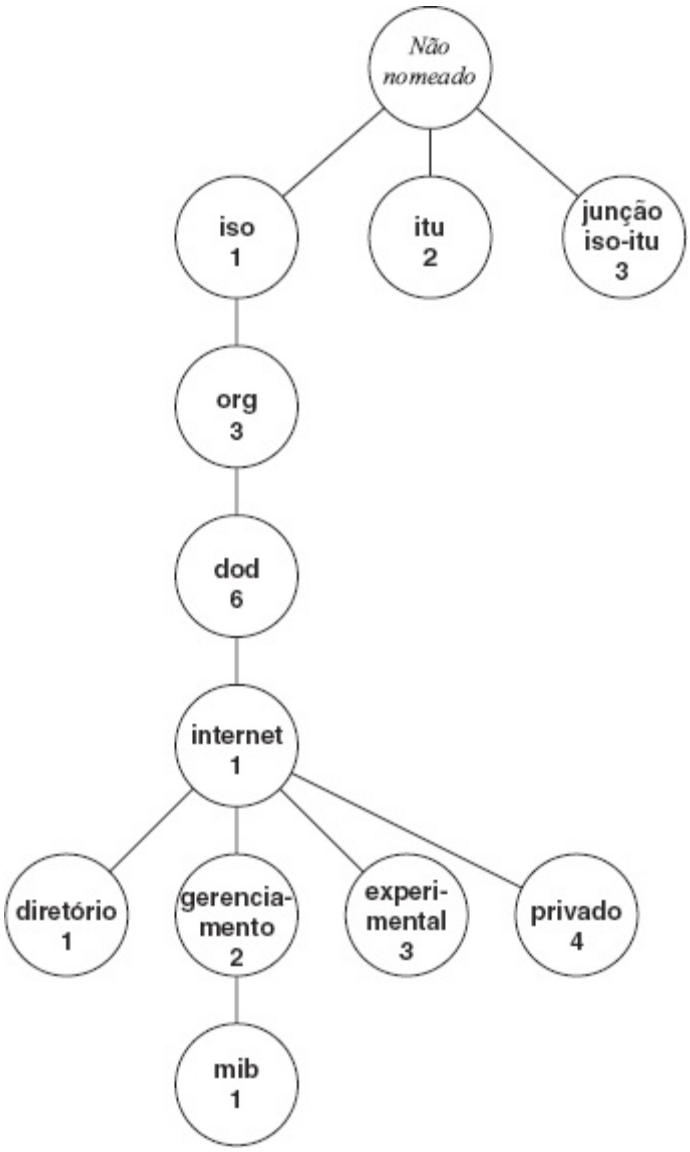


Figura 27.4 Parte do namespace identificador de objeto hierárquico usado para nomear variáveis MIB. O nome de um objeto começa com um caminho através da hierarquia.

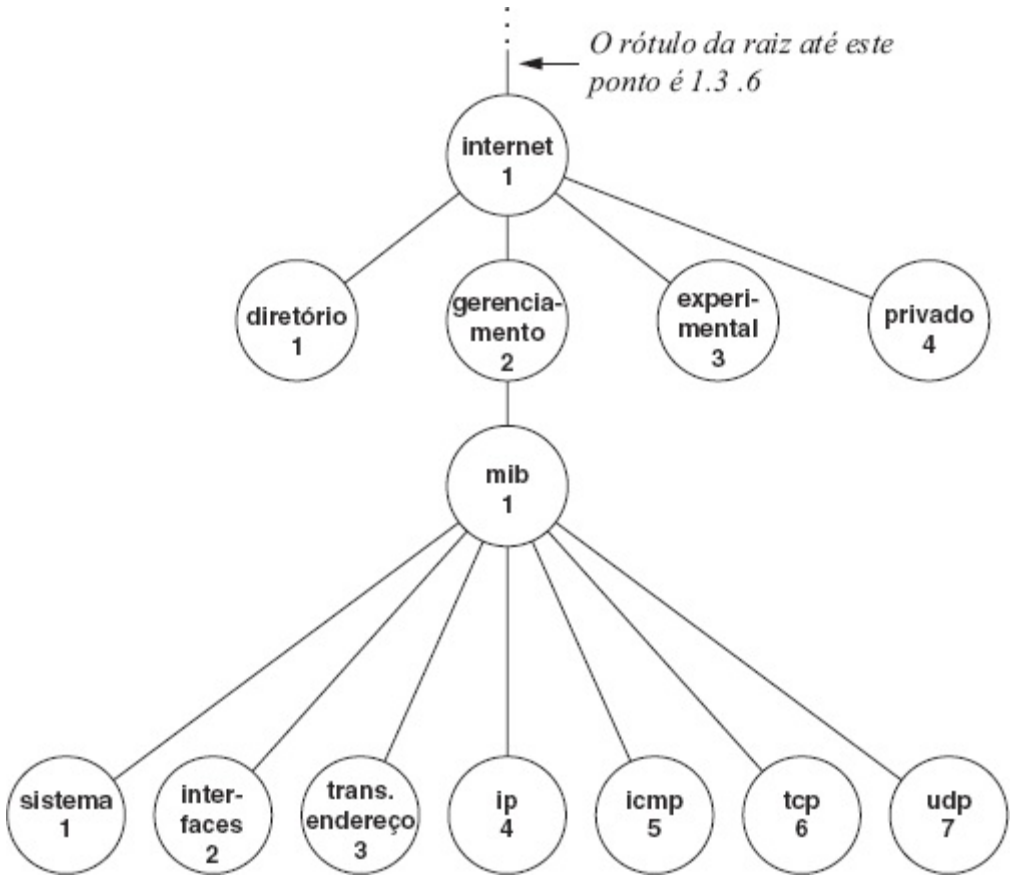


Figura 27.5 Parte do namespace identificador de objeto sob o nó *mib* do IAB. Cada subárvore corresponde a uma das categorias de variáveis MIB.

Dois exemplos tornarão a sintaxe de nomeação mais clara. A Figura 27.5 mostra que à categoria rotulada *ip* foi atribuído o valor numérico *4*. Portanto, os nomes de todas as variáveis MIB correspondentes ao IP têm uma identificação que começa com o prefixo *1 . 3 . 6 . 1 . 2 . 1 . 4*. Se uma pessoa desejasse escrever os rótulos textuais em vez da representação numérica, o nome seria:

iso . org . dod . internet . gerencto . mib . ip

A uma variável MIB nomeada *ipInReceives* foi atribuído o identificador numérico *3* sob o nó *ip* no namespace; portanto, seu nome é:

iso . org . dod . internet . mgmt . mib . ip . ipInReceives

E a representação numérica correspondente é:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 3

Quando os protocolos de gerenciamento de rede usam nomes de variáveis MIB em mensagens, cada nome tem um sufixo anexado. Para variáveis simples, o sufixo *0* se refere à instância da variável com esse nome. Assim, quando ela aparece em uma mensagem enviada para um roteador, a representação numérica de *ipInReceives* é

1 . 3 . 6 . 1 . 2 . 1 . 4 . 3 . 0

que se refere à instância de *ipInReceives* nesse roteador. Observe que não há um meio de supor o valor numérico ou o sufixo atribuído a uma variável. É necessário consultar os padrões publicados para descobrir que valores numéricos foram atribuídos a cada tipo de objeto. Portanto, os programas que fornecem mapeamentos entre a forma textual e os valores numéricos subjacentes fazem isso apenas consultando tabelas de equivalências – não existe qualquer computação em forma fechada que realize a transformação.

Como um segundo exemplo, mais complexo, considere a variável MIB *ipAddrTable*, que contém uma lista dos endereços IPv4 para cada interface de rede. A variável existe no namespace como uma subárvore sob *ip*, e a ela foi atribuído o valor numérico *20*. Portanto, uma referência a essa variável tem o prefixo:

iso . org . dod . internet . mgmt . mib . ip . ipAddrTable

Com um equivalente numérico:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 20

Em termos de linguagem de programação, pensamos na tabela de endereço IP como um array unidimensional, no qual cada elemento do array consiste em uma estrutura (registro) que contém cinco itens: um endereço IP, o índice inteiro de uma interface correspondente à entrada, uma máscara de sub-rede IP, um endereço de broadcast IP e um inteiro que especifica o tamanho de datagrama máximo que o roteador irá remontar. Obviamente, é improvável que um roteador tenha um array na memória. O

roteador pode manter essas informações em muitas variáveis ou pode precisar seguir ponteiros para encontrá-las. Entretanto, o MIB fornece um nome para o array como se ele existisse e permite que softwares de gerenciamento de rede em roteadores individuais mapeiem referências de tabela para variáveis internas apropriadas. O ponto principal é descrito a seguir.

> *Embora pareçam especificar detalhes sobre estruturas de dados, os padrões MIB não ditam a implementação. Em vez disso, as definições MIB fornecem uma interface virtual uniforme que os gerentes usam para acessar dados; um agente precisa traduzir entre os itens virtuais em um MIB e a implementação interna.*

Usando uma notação no estilo ASN.1, podemos definir *ipAddrTable*:

```
ipAddrTable ::= SEQUENCE OF IpAddrEntry
```

Onde SEQUENCE e OF são palavras-chave que definem uma *ipAddrTable* para ser um array unidimensional de *IpAddrEntrys*. Cada entrada no array é definida para consistir em cinco campos (a definição considera que IpAddress já foi definido).

```
IpAddrEntry ::= SEQUENCE {
  ipAdEntAddr
     IpAddress,
  ipAdEntIfIndex
     INTEGER,
  ipAdEntNetMask
     IpAddress,
  ipAdEntBcastAddr
     IpAddress,
  ipAdEntReasmMaxSize
     INTEGER (0..65535)
}
```

Definições adicionais precisam ser dadas para atribuir valores numéricos

a *ipAddrEntry* e a cada item na sequência *IpAddrEntrys*. Por exemplo, a definição

```
ipAddrEntry { ipAddrTable 1 }
```

especifica que *ipAddrEntry* entra sob *ipAddrTable* e possui valor numérico *1*. Da mesma forma, a definição

```
ipAdEntNetMask { ipAddrEntry 3 }
```

atribui a *ipAdEntNetMask* o valor numérico *3* sob *ipAddrEntry*.

Dizemos que *ipAddrTable* é como um array unidimensional. Entretanto, existe uma diferença significativa no modo como os programadores usam arrays e o modo como o software de gerenciamento de rede usa tabelas no MIB. Os programadores pensam em um array como um conjunto de elementos que tem um índice usado para selecionar um elemento específico. Por exemplo, o programador pode escrever xyz[3] para selecionar o terceiro elemento de um array xyz. A sintaxe da ASN.1 não usa índices inteiros. Em vez disso, as tabelas MIB anexam um sufixo no nome para selecionar um elemento específico na tabela. Para nosso exemplo de uma tabela de endereço IP, o padrão especifica que o sufixo usado para selecionar um item consiste em um endereço IP. Sintaticamente, o endereço IP (em notação decimal pontuada) é concatenado no final do nome do objeto para formar a referência. Portanto, para especificar o campo de máscara de rede na entrada da tabela de endereço IP correspondente ao endereço 128 . 10 . 2 . 3, usamos o nome:

```
iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddr
ipAdEntNetMask.128.10.2.3
```

Que, na forma numérica, se torna:

```
1 . 3 . 6 . 1 . 2 . 1 . 4 . 20 . 1 . 3 . 128 . 10 . 2 . 3
```

Embora a concatenação de um índice com o final de um nome possa parecer algo pouco prático, isso fornece uma poderosa ferramenta que permite aos clientes pesquisar tabelas sem saber o número dos itens ou o tipo dos dados usados como um índice. Uma seção mais adiante mostra como os protocolos de gerenciamento de rede usam esse recurso para percorrer uma tabela elemento por elemento.

27.9 As alterações no MIB e adições para o IPv6

O IPv6 muda o MIB superficialmente. Em vez de usar as variáveis atuais MIB que correspondem a IP (por exemplo, a contagem de todos datagramas IP que chegaram), o IETF decidiu usar variáveis separadas para IPv6. Assim, definiram-se novos nomes para o IPv6, e as variáveis MIB previamente definidas para IP agora se referem apenas ao IPv4. Similarmente, foi estabelecida uma nova categoria para ICMPv6.

Parte da motivação para uma nova estrutura MIB surge porque o IPv6 não alterou simplesmente o tamanho dos endereços. Em vez disso, o IPv6 muda a forma como os endereços são atribuídos. Em particular, permite que vários prefixos IP sejam atribuídos para uma dada interface simultaneamente. Portanto, o IPv6 MIB precisa ser estruturado de modo a criar uma tabela (ou seja, um array) de entradas que mantenham endereços. De modo semelhante, uma vez que o IPv6 usa *Neighbor Discovery* (ND) em vez de ARP, uma tabela IPv6 dá vinculações de endereços de IP para MAC. A Figura 27.6 lista as tabelas usadas com IPv6 e explica a finalidade de cada uma.

Tabela	Propósito
ipv6IfTable	Informações sobre as interfaces IPv6
ipv6IfStatsTable	Estatísticas de tráfego para cada interface
ipv6AddrPrefixTable	Prefixos IPv6 para cada interface
ipv6AddrTable	Endereços IPv6 para cada interface
ipv6RouteTable	O IPv6 (unicast) tabela de roteamento
ipv6NetToMediaTable	Endereço IPv6 para endereço físico equivalente

Figura 27.6 As seis maiores tabelas MIB introduzidas para IPv6 e uma descrição de seu conteúdo.

27.10 Simple Network Management Protocol

Os protocolos de gerenciamento de rede especificam comunicação entre um aplicativo de gerenciamento de rede em execução no computador do gerente e um agente de gerenciamento de rede (por exemplo, servidor) em execução em um dispositivo gerenciado. Além de definirem a forma e o significado das mensagens trocadas e a representação dos nomes e valores nessas mensagens, os protocolos de gerenciamento de rede também definem relações administrativas entre roteadores sendo gerenciados. Ou seja, eles cuidam da autenticação dos gerentes.

Seria de se esperar que os protocolos de gerenciamento de rede

contivessem muitos comandos. Alguns protocolos iniciais, portanto, suportavam comandos que permitiam a um gerente: *reinicializar* o sistema, *acrescentar* ou *excluir* roteadores, *desativar* ou *ativar* uma determinada interface de rede e remover vinculações de endereço em cache. A principal desvantagem de construir protocolos de gerenciamento em torno de comandos deriva da complexidade resultante. O protocolo requer um comando separado para cada operação em um item de dados. Por exemplo, o comando para excluir uma entrada da tabela de roteamento difere do comando para desativar uma interface. Consequentemente, o protocolo precisa mudar para acomodar novos itens de dados.

O SNMP usa um interessante método alternativo ao gerenciamento de rede. Em vez de definir um grande conjunto de comandos, o SNMP distribui todas as operações em um *modelo buscar-armazenar*.* Conceitualmente, o SNMP contém apenas dois comandos que permitem a um gerente buscar um valor de um item de dados ou armazenar um valor em um item de dados. Todas as outras operações são definidas como efeitos secundários dessas duas operações. Por exemplo, embora o SNMP não tenha uma operação reinicializar explícita, a reinicialização do sistema é definida por declarar uma variável MIB que dá o tempo até a próxima reinicialização, permitindo a um gerente atribuir à variável um valor. Se o gerente atribui o valor zero, o dispositivo será reinicializado imediatamente (ou seja, a atribuição funciona como um comando de reinicialização).

As principais vantagens de usar um modelo buscar-armazenar são a estabilidade, a simplicidade e a flexibilidade. O SNMP é particularmente estável porque sua definição permanece fixa, ainda que novos itens de dados sejam acrescentados ao MIB e novas operações sejam definidas como efeitos secundários do armazenamento nesses itens. O SNMP é simples de implementar, entender e depurar, pois ele evita a complexidade de haver casos especiais para cada comando. Por fim, o SNMP é especialmente flexível porque pode acomodar comandos arbitrários em uma elegante estrutura.

Do ponto de vista de um gerente, é claro, o SNMP permanece oculto. A interface de usuário com o software de gerenciamento de rede pode expressar operações como comandos imperativos (por exemplo, reinicializar). Portanto, existe pouca diferença visível entre o modo como um gerente usa o SNMP e outros protocolos de gerenciamento de rede. Na verdade, os fornecedores vendem software de gerenciamento de rede que oferece uma interface gráfica com o usuário. Esse software exibe diagramas da conectividade de rede e utiliza um estilo “apontar-e-clicar” de interação.

Na prática, o SNMP oferece mais do que operações de busca e armazenamento. A Figura 27.7, lista as oito operações, ainda que na prática somente algumas delas sejam essenciais. Por exemplo, *get-request* e *set-request* fornecem as operações básicas de busca e armazenamento, e *set-request* fornece as operações básicas de busca e armazenamento. Após um dispositivo receber uma mensagem e executar a operação, ele envia uma response. Uma única mensagem *get-request* ou *set-request* pode especificar operações em muitas variáveis MIB. O SNMP especifica que as operações precisam ser atômicas, significando que o agente tem de realizar todas as operações de uma mensagem ou nenhuma delas. Em particular, se a *set-request* especifica múltiplas atribuições e qualquer um dos itens está em erro, nenhuma atribuição será feita.

Comando	Significado
get-request	Busca um valor de uma variável específica
get-next-request	Busca um valor sem saber seu nome exato
get-bulk-request	Busca um volume grande de dados (por exemplo, uma tabel
response	Uma resposta a qualquer das requisições anteriores
set-request	Armazena um valor em uma variável específica
inform-request	Referência a dados de terceiros (por exemplo, para um prox
snmpv2-trap	Resposta acionada por um evento
report	Não definido no momento

Figura 27.7 O conjunto das operações SNMP possíveis. *Get-next-request* permite que o gerente repita através de uma tabela de itens.

Dissemos que o SNMP segue um paradigma *request-response* em que um gerente emite um comando e o dispositivo gerenciado responde. Na verdade, o SNMP permite uma exceção: um gerente pode configurar um dispositivo para enviar mensagens de *snmpv2-trap* de forma assíncrona. Por exemplo, um servidor SNMP pode ser configurado para enviar uma mensagem *snmpv2-trap* para o gerente sempre que uma de suas redes associadas se torna inacessível (ou seja, sempre que uma interface cai). De modo similar, um dispositivo pode ser configurado para enviar uma mensagem de *snmpv2-trap* sempre que um dos contadores excede um limite predefinido.

27.10.1 Pesquisando tabelas usando nomes

Lembre-se de que a ASN.1 não fornece mecanismos para declarar arrays ou indexá-las no sentido usual. Entretanto, é possível indicar elementos individuais de uma tabela anexando um sufixo ao identificador de objeto para a tabela. Infelizmente, um programa cliente pode desejar examinar entradas em uma tabela para as quais ele não sabe todos os sufixos válidos. A operação *get-next-request* lida com o problema, permitindo a um gerente percorrer uma tabela sem saber quantos itens ela contém. As regras são bastante simples. Ao enviar um *get-next-request*, o cliente fornece um prefixo de um identificador de objeto válido, *P*. O agente examina o conjunto de identificadores de objeto para todas as variáveis que ele controla, e envia uma resposta para a variável que ocorre a seguir na ordem lexicográfica. Ou seja, o agente precisa saber os nomes ASN.1 de todas as variáveis e ser capaz de selecionar a primeira variável com identificador de objeto maior que *P*. Cada tabela tem um nome. Ao armazenar um item em uma tabela, o SNMP cria um nome que começa com o nome da tabela e tem um sufixo que identifica um objeto em particular na tabela. A ideia de atribuir um nome a cada tabela é a chave: o nome não corresponde a uma variável, mas permite que um cliente forme um *get-next-request* especificando o nome da tabela. Assumindo que a tabela não esteja vazia, o dispositivo de gestão irá devolver o valor do primeiro elemento da tabela. Uma vez que o primeiro item da tabela foi recuperado, o cliente pode usar o nome desse item em uma posterior solicitação *get-next-request* para recuperar o segundo item, e assim por diante. A interação continua até que o dispositivo retorne um item cujo nome não corresponda ao prefixo da tabela (ou seja, um item para além do fim da tabela).

Considere uma pesquisa de exemplo. Lembre-se de que o *ipAddrTable* utiliza endereços IP para identificar entradas na tabela. Um cliente que não sabe quais endereços IP estão na tabela em um determinado roteador não pode formar um identificador de objeto completo. Entretanto, o cliente ainda pode usar a operação *get-next-request* para pesquisar a tabela enviando o prefixo

iso . org . dod . internet . mgmt . mib . ip . ipAddrTable . ipAddrEntry . ipAdEntNetMask que, na forma numérica, é:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 20 . 1 . 3

O servidor retorna o campo máscara de rede da primeira entrada em *ipAddrTable*. O cliente usa o identificador de objeto completo retornado pelo servidor para requisitar o próximo item na tabela.

27.11 Formato de mensagem SNMP

Diferentemente da maioria dos protocolos TCP/IP, as mensagens SNMP não possuem campos fixos. Em vez disso, elas usam a codificação ASN.1 padrão. Portanto, uma mensagem pode ser difícil de ser decodificada e entendida por humanos. Após examinar a definição de mensagem SNMP na notação ASN.1, examinaremos brevemente o esquema de codificação ASN.1 e veremos um exemplo de uma mensagem SNMP codificada.

A Figura 27.8 mostra como uma mensagem SNMP pode ser descrita com uma gramática no estilo ASN.1. Em geral, cada item na gramática consiste em um nome descritivo seguido de uma declaração do tipo do item. Por exemplo, um item como

```
msgVersion INTEGER (0..2147483647)
```

declara o nome msgVersion ser um inteiro positivo inferior ou igual a 2147483647.

```
SNMPv3Message ::=
  SEQUENCE {
     msgVersion INTEGER (0..2147483647),
          -- nota: o número de versão 3 é usado
   para o SNMPv3
     msgGlobalData HeaderData,
     msgSecurityParameters OCTET STRING,
     msgData ScopedPduData

}
```

Figura 27.8 O formato de mensagem SNMP na notação ao estilo ASN.1. O texto seguinte a dois traços consecutivos é um comentário.

Como a figura mostra, cada mensagem SNMP consiste em quatro partes principais: um inteiro que identifica a versão do protocolo, dados de cabeçalho adicionais, um conjunto de parâmetros de segurança e uma área de dados que transporta o payload. É necessário que uma definição precisa

seja fornecida para cada um dos termos usados. Por exemplo, a Figura 27.9 ilustra como o conteúdo da seção *HeaderData* pode ser especificado.

```
HeaderData :: = SEQUENCE {
  msgID INTEGER (0..2147483647),
     -- usado para comparar respostas com
  requisições          msgMaxSize          INTEGER
  (484..2147483647),
     -- tamanho máximo de resposta que o emissor
  pode aceitar msgFlags OCTET STRING (SIZE(1)),
     --   Identificador   de   bit   individual
  especifica características da mensagem
     -- usado bit 7 de autorização
     -- usado bit 6 de privacidade
     -- bit 5 reportability (i.e., necessária
  uma    resposta)    msgSecurityModel    INTEGER
  (1..2147483647)
     -- determina o formato exato dos parâmetros
  de segurança como seguem
}
```

Figura 27.9 A definição da área *HeaderData* numa mensagem SNMP.

A área de dados em uma mensagem SNMP é dividida em *Protocol Data Units* (*PDUs*). Cada PDU consiste em uma requisição (enviada pelo cliente) ou uma resposta (enviada por um agente). O SNMP permite que cada PDU seja enviado como texto simples ou, ainda, criptografado para confidencialidade. Portanto, a gramática especifica uma escolha. Na terminologia de linguagem de programação, o conceito é conhecido como uma união discriminada.

```
ScopedPduData ::= CHOICE {
  plaintext ScopedPDU,
  encryptedPDU  OCTET  STRING  --  valor  do
ScopedPDU criptografado
}
```

Uma PDU criptografada com um identificador do *motor** que a produz. O ID de mecanismo é seguido pelo nome do contexto e os octetos da

mensagem criptografada.

```
ScopedPDU ::= SEQUENCE {
  contextEngineID OCTET STRING,
  contextName OCTET STRING,
  data ANY -- por exemplo, uma PDU como
definida a seguir
}
```

O item rotulado como dados na definição de ScopedPDU tem um tipo ANY porque o campo contextName define os detalhes exatos do item. O modelo de processamento de mensagem SNMPv3 (v3MP) especifica que os dados precisam consistir em uma das PDUs SNMP, como mostra a Figura 27.10.

```
PDU ::=
  CHOICE {
     get-request
        GetRequest-PDU,
     get-next-request
        GetNextRequest-PDU,
     get-bulk-request
        GetBulkRequest-PDU,
     response
        Response-PDU,
     set-request
        SetRequest-PDU,
     inform-request
        InformRequest-PDU,
     snmpV2-trap
        SNMPv2-Trap-PDU,
     report
        Report-PDU,
}
```

Figura 27.10 As definições ASN.1 de uma PDU SNMP. A sintaxe para cada tipo de requisição precisa ser mais especificada.

A definição especifica que cada unidade de dados de protocolo consiste em um de oito tipos. Para completar a definição de uma mensagem SNMP, precisamos especificar mais a sintaxe dos oito tipos individuais. Por

exemplo, a Figura 27.11 mostra a definição de um *get-request*.

```
GetRequest-PDU ::= [0]
  IMPLICIT SEQUENCE {
     request-id
         Integer32,
     error-status
         INTEGER (0..18),
     error-index
         INTEGER (0..max-bindings),
     variable-bindings
         VarBindList
}
```

Figura 27.11 Definição ASN.1 de uma mensagem *get-request*. Formalmente, a mensagem é definida para ser um Get-Request.PDU.

Mais definições no padrão especificam o significado dos outros termos indefinidos. Tanto *error-status* quanto *error-index* são inteiros de único octeto que contêm o valor zero em uma requisição.

Se um erro ocorrer, os valores enviados em uma resposta identificam a sua causa. Finalmente, *VarBindList* contém uma lista dos identificadores de objeto para os quais o cliente busca valores. Em termos da ASN.1, as definições especificam que *VarBindList* é uma sequência de pares de nome de objeto e valor. A ASN.1 representa os pares como uma sequência de dois itens. Portanto, na requisição mais simples possível, *VarBindList* é uma sequência de dois itens: um nome e um *null*.

27.12 Um exemplo de mensagem SNMP codificada

A forma codificada da ASN.1 utiliza campos de tamanho variável para representar itens. Em geral, cada campo começa com um cabeçalho que especifica o tipo de objeto e seu tamanho em bytes. Por exemplo, cada SEQUÊNCIA começa com um octeto contendo 30 (hexadecimal); o próximo octeto especifica o número de octetos seguintes que constituem a sequência.

A Figura 27.12 contém um exemplo de mensagem SNMP que ilustra como os valores são codificados em octetos. A mensagem é um *get-request*

que especifica o item de dados *sysDescr* (identificador de objeto numérico 1 3 6 1 2 1 1 1 0). Como o exemplo mostra uma mensagem real, ele inclui muitos detalhes que não têm sido discutidos. Em especial, a mensagem contém uma seção *msgSecurityParameters*; a mensagem de exemplo especificamente usa a forma *UsmSecurity Parameters* dos parâmetros de segurança. Deve ser possível, no entanto, correlacionar outras seções da mensagem com as definições anteriores.

Como a Figura 27.12 mostra, a mensagem inicia com um código para SEQUÊNCIA que tem um tamanho de 103 octetos.* O primeiro item na sequência é um inteiro de 1 octeto que especifica a versão do protocolo; o valor 3 indica que essa é uma mensagem SNMPv3. Os campos sucessivos definem um ID de mensagem e o tamanho de mensagem máximo que o emissor pode aceitar em uma resposta. As informações de segurança, incluindo o nome do usuário (*ComerBook*), seguem o cabeçalho de mensagem.

O *GetRequest-PDU* ocupa a parte final da mensagem. A sequência rotulada *ScopedPDU* especifica um contexto no qual se deve interpretar o restante da mensagem. O octeto A0 especifica a operação como um *get-Request*. O bit cinco de A0 indica um tipo de dados não primitivo, e o bit de ordem superior significa que a interpretação do octeto é específica ao contexto. Ou seja, o valor hexadecimal A0 especifica um *GetRequest-PDU* apenas quando usado no contexto; ele não é um valor universalmente reservado. Seguindo o octeto de requisição, o octeto de tamanho especifica que a requisição possui 26 octetos. O tamanho do ID de requisição é 2 octetos; o *error-status* e *error-index* estão cada qual em um octeto. Finalmente, a sequência de pares *Var-BindList* contém uma vinculação, um identificador de objeto único vinculado a um valor *null*. O identificador é codificado como esperado, exceto que os dois primeiros rótulos numéricos são combinados em um único octeto.

```
   30      67      02      01      03
SEQUENCE len=103 INTEGER  len=1   vers=3

   30      0D      02      01      2A
SEQUENCE len=13 INTEGER  len=1   msgID=42

   02      02      08      00
INTEGER  len=2   maxmsgsize=2048

   04      01      04
 string  len=1  msgFlags=0x04 (bits mean noAuth, noPriv, reportable)

   02      01      03
INTEGER  len=1  used-based security

   04      25      30      23
 string  len=37 SEQUENCE len=35 UsmSecurityParameters

   04      0C      00      00      00      63      00      00      00
 string  len=12  msgAuthoritativeEngineID ...
```

```
   A1      C0      93      8E      23
engine is at IP address 192.147.142.35, port 161

   02      01      00
INTEGER  len=1  msgAuthoritativeEngineBoots=0

   02      01      00
INTEGER  len=1  msgAuthoritativeEngineTime=0

   04      09      43      6F      6D      65      72      42      6F
 string  len=9     -----msgUserName value is "ComerBook"------------
   6F      6B
------------

   04      00
 string  len=0   msgAuthenticationParameters (none)

   04      00
 string  len=0   msgPrivacyParameters (none)

   30      2C
SEQUENCE len=44  ScopedPDU

   04      0C      00      00      00      63      00      00
 string  len=12    ------------------contextEngineID-------
   00      A1      c0      93      8E      23
   ---------------------------------------

   04      00
 string  len=0  contextName = "" (default)

CONTEXT [0] IMPLICIT SEQUENCE

   A0      1A
 getreq. len=26

   02      02      4D      C6
 INTEGER len=2   request-id = 19910

   02      01      00
 INTEGER len=1 error-status = noError(0)

   02      01      00
 INTEGER len=1 error-index=0

   30      0E
SEQUENCE  len=14 VarBindList

   30      0C
SEQUENCE  len=12 VarBind

   06                      08
 OBJECT IDENTIFIER name  len=8

   2B      06      01      02      01      01      01      00
 1.3  .   6   .   1   .   2   .   1   .   1   .   1   .   0 (sysDescr.0)

   05      00
 null   len=0 (no value specified)
```

Figura 27.12 A forma codificada de um *get-request* SNMPv3 para o item de dados *sysDescr* com octetos mostrados em hexadecimal e um comentário explicando seu significado abaixo. Octetos

relacionados foram agrupados em linhas; elas são contíguas na mensagem.

27.13 Segurança em SNMPv3

A versão 3 do SNMP representa uma evolução que segue e estende a estrutura básica das versões anteriores. As principais mudanças aparecem nas áreas da segurança e da administração. Os objetivos são duplos. Primeiro, o SNMPv3 é projetado para ter políticas de segurança gerais e flexíveis, possibilitando que as interações entre um gerente e dispositivos gerenciados estejam de acordo com as políticas de segurança especificadas por uma organização. Segundo, o sistema é projetado para facilitar a administração da segurança.

Para alcançar generalidade e flexibilidade, o SNMPv3 inclui facilidades a vários aspectos da segurança e permite que cada um deles seja configurado independentemente. Por exemplo, o SNMPv3 aceita *autenticação de mensagem* para garantir que instruções se originem de um gerente válido, *privacidade* para assegurar que ninguém possa ler mensagens enquanto são transferidas da estação de um gerente para um dispositivo gerenciado, e *autorização* e *controle de acesso* baseado em visão, para garantir que apenas gerentes autorizados acessem determinados itens. Para facilitar a configuração ou alteração do sistema de segurança, o SNMPv3 permite *configuração remota*, significando que um gerente autorizado pode mudar a configuração dos itens de segurança listados aqui sem estar fisicamente presente no dispositivo.

27.14 Resumo

Os protocolos de gerenciamento de rede permitem que um gerente monitore e controle dispositivos de rede, tais como hosts e roteadores. Um programa cliente de gerenciamento de rede executa na estação de trabalho do gerente e pode contatar um ou mais servidores, chamados agentes, rodando nos dispositivos gerenciados. Como uma internet consiste em máquinas e redes heterogêneas, o software de gerenciamento TCP/IP é executado como programas aplicativos e utiliza protocolos de transporte TCP/IP (por exemplo, UDP) para comunicação entre clientes e servidores.

O protocolo de gerenciamento de rede TCP/IP padrão é o Simple Network Management Protocol (SNMP). O SNMP define um protocolo de gerenciamento de rede de baixo nível que fornece duas operações conceituais: buscar um valor de uma variável ou armazenar um valor em

uma variável. No SNMP, outras operações ocorrem como efeitos secundários da mudança de valores em variáveis. O SNMP define o formato das mensagens que viajam entre o computador de um gerente e uma entidade gerenciada.

Um conjunto de padrões associados ao SNMP define o conjunto de variáveis que uma entidade gerenciada mantém. O conjunto de variáveis constitui uma base de informações de gerenciamento (MIB). As variáveis MIB são descritas usando ASN.1, uma linguagem formal que apresenta uma forma codificada concisa, bem como uma notação precisa e legível por humanos para nomes e objetos. A ASN.1 usa um namespace hierárquico para garantir que todos os nomes MIB sejam globalmente únicos, ao mesmo tempo em que permite que subgrupos atribuam partes do namespace.

EXERCÍCIOS

27.1 Capture um pacote SNMP com um analisador de rede e decodifique os campos.

27.2 Leia o padrão para descobrir como a ASN.1 codifica os dois primeiros valores numéricos de um identificador de objeto em um único octeto. Qual é a motivação para a decodificação?

27.3 Suponha que os projetistas do MIB precisem definir uma variável que corresponda a um array bidimensional. Explique como a notação ASN.1 pode acomodar referências para essa variável.

27.4 Quais são as vantagens e desvantagens de definir nomes ASN.1 globalmente únicos para variáveis MIB?

27.5 Consulte os padrões e associe cada item na Figura 27.12 a uma definição ASN.1 correspondente.

27.6 Se você tiver um código de cliente SNMP disponível, experimente usá-lo para ler variáveis MIB em um roteador local. Qual é a vantagem de permitir que gerentes arbitrários leiam variáveis em todos os roteadores? E a desvantagem?

27.7 Leia a especificação MIB para encontrar a definição da variável ipRoutingTable que corresponde a uma tabela de roteamento IPv4. Escreva um programa que use o SNMP para contatar múltiplos roteadores e veja se qualquer entrada em suas tabelas de roteamento causa um loop de encaminhamento. Exatamente que nomes ASN.1 esse programa deve gerar?

27.8 Estenda o exercício anterior para executar a mesma tarefa para o IPv6.

27.9 Considere a implementação de um agente SNMP. Faz sentido

organizar variáveis MIB na memória exatamente da forma como o SNMP as descreve? Justifique sua resposta.

27.10 Argumente que o SNMP é um nome impróprio porque o SNMP não é "simples".

27.11 Leia sobre o padrão de segurança IPSec, descrito no Capítulo 29. Se uma organização usar IPSec, a segurança no SNMPv3 também será necessária? Justifique sua resposta.

27.12 Faz sentido usar SNMP para gerenciar todos os dispositivos? Sim ou não? Por quê? (Dica: considere um dispositivo de hardware simples, como um modem DSL.)

---

* Usamos o termo *sistema de gerenciamento* para incluir dispositivos convencionais, como roteadores e computadores desktop, bem como dispositivos como impressoras e sensores.

* Quando o MIB foi definido, usava-se a terminologia tabela de roteamento em vez de tabela de encaminhamento.

* ASN.1 normalmente é pronunciado lendo-se o ponto: "A-S-N ponto 1".

** O Capítulo 23 explica como a autoridade é delegada em um namespace hierárquico.

*** O NIST é a antiga National Bureau of Standards (agência nacional de padrões).

* Os leitores familiarizados com arquitetura de hardware observarão que o barramento I/O, em um computador típico, também lança todas as operações em um modelo buscar-armazenar.

* O SNMPv3 distingue entre uma aplicação que usa o serviço fornecido pelo SNMP e um motor, que é o software subjacente que transmite requisições e recebe respostas.

* Os itens de sequência ocorrem frequentemente em uma mensagem SNMP porque o SNMP usa SEQUÊNCIA em vez das construções de linguagem de programação convencionais, como *array* ou *struct*.

CAPÍTULO 28

Software Defined Networking (SDN, OpenFlow)

CONTEÚDOS DO CAPÍTULO

28.1 Introdução

Os capítulos anteriores descrevem o paradigma fundamental de comunicação oferecido pelo IP e usado na Internet global: um serviço de entrega de pacotes de melhor esforço. O sistema de comunicação constitui-se de roteadores que usam o endereço de destino de cada datagrama para decidir como encaminhá-lo para o seu destino.

Este capítulo considera uma alternativa: um sistema de comunicação que pode orientar o tráfego através de caminhos prescritos pelos gerentes de rede, segundo uma ampla variedade de critérios. Assim, faz uma breve análise da motivação para a nova abordagem e de alguns de seus usos potenciais. Em seguida, o presente capítulo explora o uso do hardware existente e examina uma tecnologia em particular que gerentes podem usar para especificar caminhos. Veremos que a abordagem aqui apresentada não é completamente nova. Em vez disso, ela combina muitas ideias dos capítulos anteriores, incluindo MPLS (Capítulo 16), a classificação de pacotes (Capítulo 17), a comutação Ethernet (Capítulo 2) e a virtualização de rede (Capítulo 19).

28.2 Rotas, caminhos e conexões

O paradigma básico de encaminhamento IP pode ser caracterizado como *igualitário*, no sentido de que todo o tráfego de uma determinada origem para um determinado destino segue o mesmo caminho. Além disso, uma

vez que qualquer datagrama chega a um roteador, encaminha recursos usando o destino do datagrama e é independente da fonte ou o conteúdo do datagrama. Ou seja, uma tabela de encaminhamento em um roteador só contém uma entrada para um determinado destino.

Como vimos, diversas variações foram criadas. Por exemplo, o Capítulo 15 introduz o conceito de tunelamento e o Capítulo 16 discute MPLS, que permite que um roteador de borda considere pacotes individuais e escolha um caminho ao longo do qual enviar cada pacote. Especificamente, nós vimos como um datagrama pode ser encapsulado em um cabeçalho MPLS e como roteadores intermediários usam a informação no cabeçalho de encapsulamento em vez de endereço de destino do datagrama ao tomar decisões de encaminhamento.

28.3 Engenharia de tráfego e controle de seleção de caminho

Lembre-se, do Capítulo 16, de que a principal motivação para MPLS surge do desejo dos operadores de rede de realizar *engenharia de tráfego*. Ou seja, em vez de enviar todo o tráfego para um destino ao longo de um único caminho, um gerente de rede pode querer escolher caminhos com base no tipo de tráfego, a prioridade atribuída a datagramas, a quantia que cada remetente pagou, ou outras considerações econômicas. Em essência, a engenharia de tráfego muda de um sistema em que os protocolos de roteamento tomam decisões para um sistema em que os operadores de rede têm o controle dos caminhos que os datagramas seguem. Um operador pode especificar como cada datagrama individual é mapeado em um dos caminhos MPLS predefinidos apropriados para o datagrama. Observe que o caminho MPLS selecionado para um determinado datagrama pode não ser um caminho mais curto através de roteadores intermediários. Mais importante, as redes MPLS geralmente garantem que todos os datagramas de um dado fluxo são mapeados para o mesmo caminho MPLS. O ponto importante é descrito a seguir.

Tecnologias de engenharia de tráfego mudam de um paradigma em que protocolos de roteamento encontram caminhos mais curtos que todos os datagramas devem seguir para um sistema em que um gerente de rede pode controlar o caminho para cada fluxo individual.

28.4 Redes orientadas à conexão e sobreposições de roteamento

Duas abordagens gerais têm sido utilizadas para proporcionar controle de cada fluxo de um sistema de comunicação:

- usar uma infraestrutura de rede orientada à conexão;
- impor roteamento de sobreposição em uma infraestrutura de comutação de pacote.

Rede orientada à conexão. Uma rede de orientada à conexão define uma rota de encaminhamento independente para cada fluxo. Várias tecnologias de rede de orientada à conexão têm sido criadas (por exemplo, X.25 e ATM). Embora os detalhes variem, cada tecnologia segue a mesma abordagem genérica: antes de um aplicativo poder enviar dados, deve contatar a rede para solicitar que uma conexão seja estabelecida. A rede pode escolher um caminho específico para cada ligação. Depois de usar a conexão para se comunicar, o aplicativo contata novamente a rede para solicitar que a conexão seja terminada.

Como cada uso requer uma nova conexão fim a fim, uma rede orientada à conexão dá a um gerente controle sobre seleção de caminho e encaminhamento. Normalmente, um gerente configura um conjunto de políticas em cada comutador na rede. A configuração de conexão exige que cada comutador ao longo de um caminho concorde com a conexão. Quando um pedido de conexão chega a um comutador, ele consulta as políticas para determinar se e como satisfazer o pedido.

Tecnologias de roteamento de sobreposição. Um roteamento de sobreposição (routing overlay) consiste em um sistema de encaminhamento que impõe uma topologia de rede virtual e, em seguida, usa o encaminhamento de internet existente para entregar pacotes entre os nós na topologia virtual. Em essência, a sobreposição de roteamento cria um conjunto de túneis. Do ponto de vista dos protocolos de roteamento, cada túnel funciona como uma conexão de rede ponto a ponto entre roteadores. Assim, os protocolos de roteamento apenas encontram caminhos através dos túneis, o que significa que o encaminhamento será condicionado à topologia virtual.

O Capítulo 19 discute a ideia geral de redes sobrepostas e outros capítulos fornecem exemplos específicos. O Capítulo 18 discute como o IP móvel usa o tunelamento para transmitir datagramas para um equipamento móvel que está temporariamente fora de casa; o Capítulo 16 descreve o uso

de sobreposições em tecnologias de mudança de rótulo, como o MPLS.

Ambas as tecnologias de rede orientada à conexão e de rede de sobreposição têm vantagens e desvantagens. Uma rede orientada à conexão pode ser implementada em hardware, o que significa que ela pode usar classificação de alta velocidade baseada em hardware e mecanismos de comutação de rótulo. Portanto, uma rede orientada à conexão pode ser escalada para redes de taxas de dados mais altas. Como são criadas por software, sobreposições são flexíveis e fáceis de mudar. Além disso, se os sites são conectados pela Internet global, uma topologia de sobreposição arbitrária pode ser imposta ou alterada rapidamente. Mais importante ainda, um conjunto de topologias pode ser imposto simultaneamente, o que permite que o tráfego seja segregado em várias redes virtuais.

Além de exigir sobrecarga de processamento, sobreposições podem resultar em roteamento ineficiente. Para entender o porquê, observe que a abstração sobreposição oculta a arquitetura e os custos da rede subjacente. Por exemplo, suponha que uma empresa configure túneis de sobreposição entre seis sites e cada túnel atravessa a Internet global entre os sites. A empresa pode usar uma estimativa dos custos da Internet para configurar métricas de roteamento artificiais que direcionam o tráfego ao longo de rotas preferidas. No entanto, o sistema de roteamento da Internet e o sistema de roteamento de sobreposição operam independentemente. Se o custo de roteamento de uma rede subjacentes aumentar, o sistema de sobreposição não vai saber da mudança. Consequentemente, o encaminhamento de sobreposição continuará a seguir o caminho original. O ponto importante é descrito a seguir.

Embora as redes orientada à conexão e as tecnologias de roteamento de sobreposição possam, cada uma, dar aos gerentes de rede controle sobre o tráfego, cada abordagem tem algumas desvantagens.

28.5 SDN: uma nova abordagem híbrida

Surge a pergunta: podemos combinar tecnologias de rede de conexão orientada e tecnologias de sobreposição de roteamento para superar suas fraquezas individuais e aproveitar os pontos fortes de ambas as abordagens? Para casos especiais, a resposta é sim. A combinação, que é conhecida como *rede definida por software* (*Software Defined Networking – SDN*), usa as ideias a seguir.

- Para evitar a sobrecarga que surge a partir da execução da classificação em software, usar hardware de classificação de alta velocidade.
- Para evitar o gargalo que resulta da realização de encaminhamento de pacotes em software, usar hardware de encaminhamento de alta velocidade.
- Para dar confiabilidade aos gerentes e permitir engenharia de tráfego, evitar o uso de protocolos de roteamento para definir as rotas para todo o tráfego e, ao contrário, possibilitar aos gestores especificarem como lidar com cada caso.
- Para dimensionar ao tamanho da Internet, permitir aplicativos de gestão, em vez de seres humanos, para configurar e controlar os dispositivos de rede.

As próximas seções examinam cada uma das ideias básicas e, então, descrevem uma tecnologia específica que as incorpora.

28.6 Separação de dados e de controle

Conceitualmente, um dispositivo de rede, como um roteador ou comutador, pode ser dividido em duas partes: os mecanismos que permitem aos gerentes configurarem e controlarem o dispositivo e os mecanismos que lidam com pacotes. Para capturar a dicotomia, usamos os termos *plano de controle* e *plano de dados*. O plano de dados lida com todo o processamento de pacotes e encaminhamento. O plano de controle fornece uma interface de gerenciamento. A Figura 28.1 ilustra a divisão conceitual.

Como a figura indica, as conexões ao longo das quais os pacotes chegam e partem têm uma capacidade muito maior do que a conexão de gerenciamento usada para controle. Usamos os termos *caminho de dados* como referência ao percurso de alta capacidade para o tráfego de pacotes e *controle de caminho* como referência ao caminho de menor capacidade que um gerente utiliza para controlar o dispositivo.

Infelizmente, quando criam dispositivos de rede, os fornecedores costumam seguir uma abordagem altamente integrada em que plano de controle e o plano de dados do aparelho estão fortemente acoplados. O dispositivo exporta uma interface de gerenciamento que permite que um gerente controle funções específicas. Por exemplo, a interface de um fornecedor em um comutador VLAN permite que um gerente especifique um conjunto de VLANs e associe uma determinada porta com uma das VLANs. Da mesma forma, a interface de um fornecedor em um roteador

permite que um gerente preencha as entradas da tabela de encaminhamento. No entanto, um gerente não pode configurar regras de classificação específicas ou controlar como pacotes individuais são manipulados.

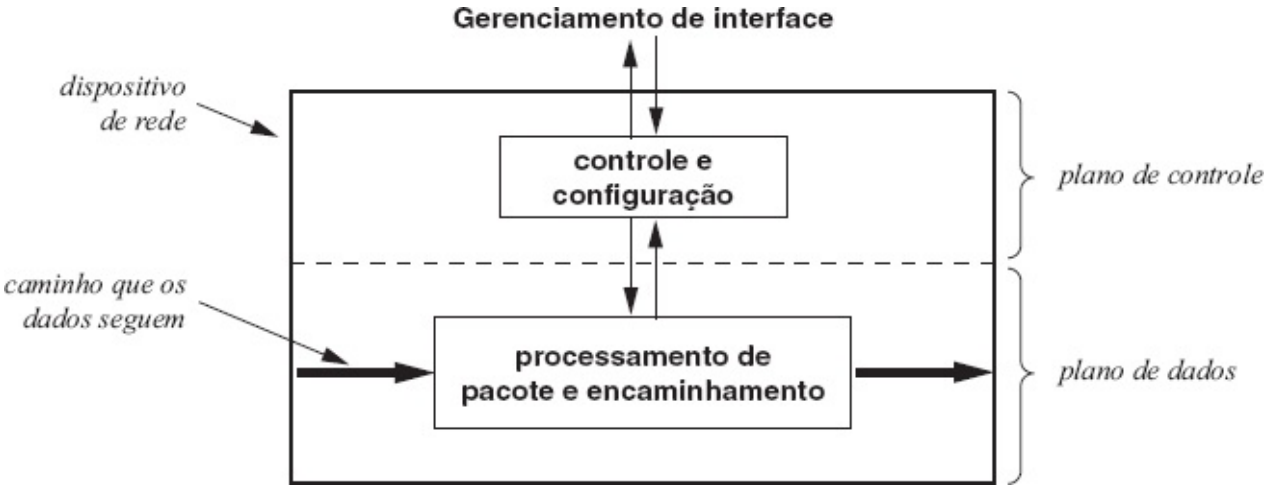


Figura 28.1 Ilustração das duas partes conceituais de um dispositivo de rede: os planos de controle e de dados.

Para alcançar uma solução híbrida, temos de encontrar uma maneira de separar o plano de dados e o plano de controle. Ou seja, precisamos de uma forma de substituir o sistema de plano de controle do fornecedor com uma versão personalizada. Nosso novo sistema de controle deve ter acesso direto ao hardware do plano de dados, e deve ser capaz de configurar o processamento e encaminhamento de pacotes. A Figura 28.2 ilustra o conceito: um novo sistema de controle adicionado a um dispositivo de rede.

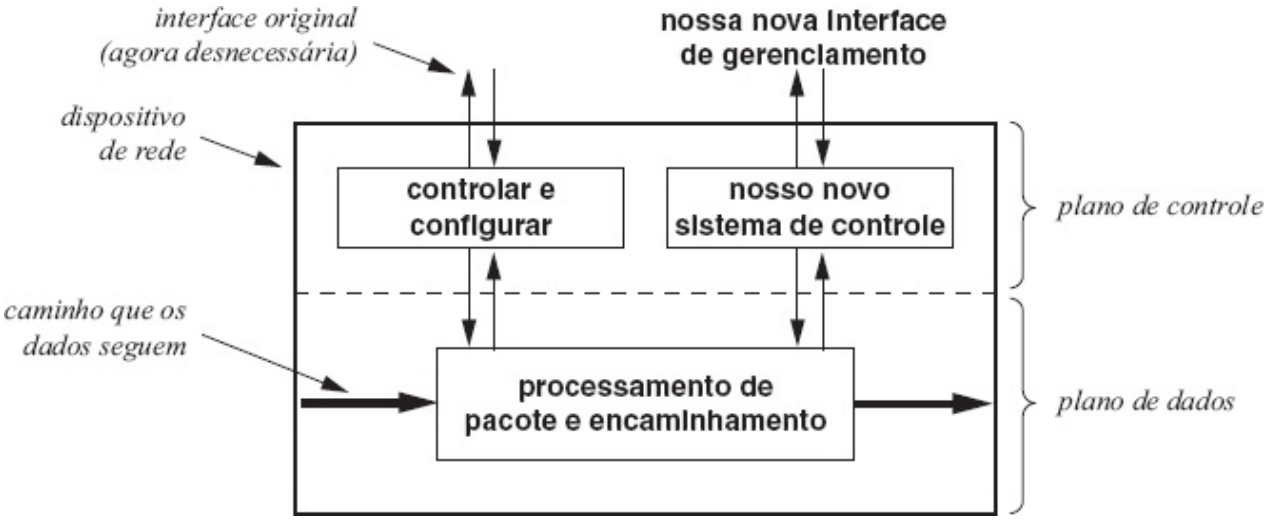


Figura 28.2 A organização conceitual de um dispositivo com um novo sistema de controle que substitui o sistema do fornecedor.

Em teoria, é possível a instalação de um novo sistema de controle em um dispositivo de rede, sem adicionar mais hardware. Para entender o porquê, observe que a maioria dos sistemas de controle é implementada em software: um processador embutido executa o programa de controle a partir da ROM. Assim, o sistema de controle pode ser alterado através do carregamento do novo software na ROM. Na prática, substituir o software de controle do fornecedor é insustentável por duas razões.

Primeiro, a tecnologia SDN depende de configuração convencional e de encaminhamento para ser posta em prática inicialmente. Em segundo lugar, uma vez que muitas execuções de software de controle de tarefas são específicas para o hardware subjacente, o plano de controle deve ser especializado para cada dispositivo de hardware. O ponto é importante é descrito a seguir.

Embora a tecnologia SDN precise de uma nova funcionalidade de plano de controle em dispositivos de rede, substituir completamente o software de controle do fornecedor é impraticável.

28.7 A arquitetura SDN e controladores externos

A abordagem que tem sido adotada pelo SDN separa software de controle dos dispositivos de rede subjacente. Em vez de reescrever completamente o software de controle dos fornecedores, o SDN utiliza uma abordagem de aumento: o software SDN é executado em um sistema externo e um pequeno módulo é adicionado a um dispositivo que permite que o sistema SDN externo configure o hardware subjacente. O sistema externo, normalmente um PC convencional, é chamado um *controlador*. A Figura 28.3 ilustra a arquitetura.

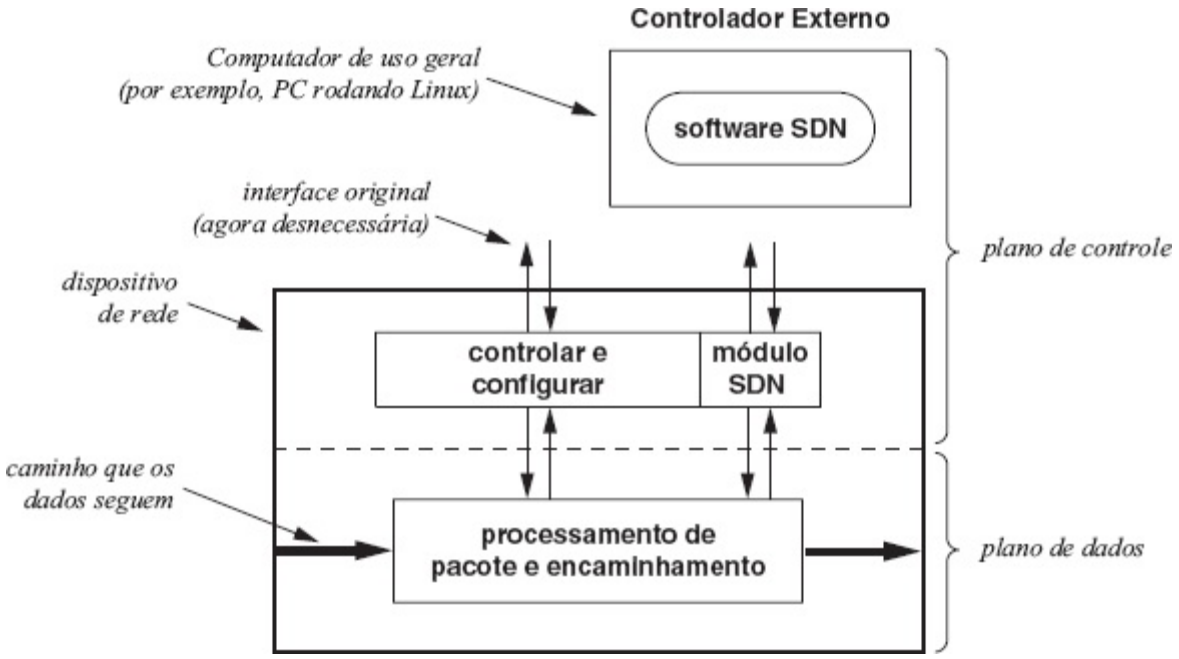


Figura 28.3 A arquitetura básica SDN com um controlador externo configurando o hardware de classificação e encaminhamento em um dispositivo de rede.

Como a figura mostra, a adição de um controlador externo estende a funcionalidade do plano de controle externamente ao dispositivo. Um novo *módulo* SDN deve ser adicionado ao plano de controle de um dispositivo a fim de permitir que o controlador externo configure o plano de dados. A figura mostra o módulo SDN como pequeno, porque o código não contém inteligência nem fornece uma interface de gerenciamento convencional. Em vez disso, o módulo de SDN meramente aceita comandos de baixo nível a partir do controlador externo e passa cada comando através da unidade de processamento de plano de dados. O ponto é que um módulo SDN é minimalista – ele contém complexidade e funcionalidade substancialmente menores do que um mecanismo de plano de controle típico.

Ao mover as funções complexas do plano de controle para um controlador externo, a abordagem SDN dá mais controle aos gerentes. Um gerente pode configurar quaisquer regras de classificação e encaminhamento que o software SDN permitir, mesmo que elas sejam diferentes das regras de VLAN convencional ou sub-rede IP permitidas pelo software do fornecedor. Veremos que um gerente pode escolher como classificar e encaminhar cada pacote individual. Em essência, o SDN utiliza o hardware de classificação de dados e encaminhamento em um dispositivo de rede, mas ignora o plano de controle oferecido pelo fornecedor. A ideia

pode ser resumida da maneira a seguir.

> *A abordagem SDN separa o processamento do plano de controle do processamento de plano de dados, movendo as funções de controle para um controlador externo. Como ele emite comandos de baixo nível que configuram o plano de dados do dispositivo, um controlador externo pode oferecer a um gerente mais controle do que o software de controle de plano do fornecedor.*

28.8 SDN através de vários dispositivos

A descrição anterior se concentra no mecanismo básico, explicando como a tecnologia SDN pode ser utilizada para configurar e controlar um único dispositivo de rede. Para fornecer recursos significativos para uma internet, a tecnologia deve ser estendida a vários dispositivos de rede (por exemplo, todos os dispositivos em uma intranet de um *campus* ou em uma rede de um ISP). A ideia importante é que queremos todos os controladores conectados a uma rede, o que permitirá que o software aplicativo de gerenciamento em execução nos controladores se comunique com o conjunto de dispositivos e o coordene.

Duas questões fundamentais se referem a uma arquitetura global: quantos controladores são necessários e como eles devem ser interligados? Como se pode imaginar, o número de controladores externos necessários depende do tipo de dispositivos de rede que estão sendo controlados e do software SDN. Se os dispositivos de rede forem pequenos (por exemplo, modems) e o computador utilizado como um controlador for poderoso, um único controlador pode lidar com vários dispositivos. No entanto, se um determinado dispositivo exige que o sistema de controle lide com muitas exceções ou faça alterações frequentes na configuração, um controlador pode ser capaz de lidar só com um dispositivo de rede. Por enquanto, vamos supor que cada dispositivo de rede tem um controlador dedicado; seções posteriores consideram uma extensão em que um controlador pode lidar com vários dispositivos.

Em termos de comunicação entre controladores, vamos imaginar que existe uma rede de gerenciamento separada para conectar o conjunto de controladores. Controladores usam a rede de gerenciamento para se comunicarem entre si, o que significa que podemos criar software aplicativo de gerenciamento que funciona nos controladores e coordena todo o

conjunto de dispositivos. Para permitir que um técnico humano defina políticas e execute outras tarefas, assumiremos que a rede de gerenciamento inclui também um computador de um gerente. A Figura 28.4 ilustra uma versão idealizada de uma rede de gerenciamento que interconecta os controladores.

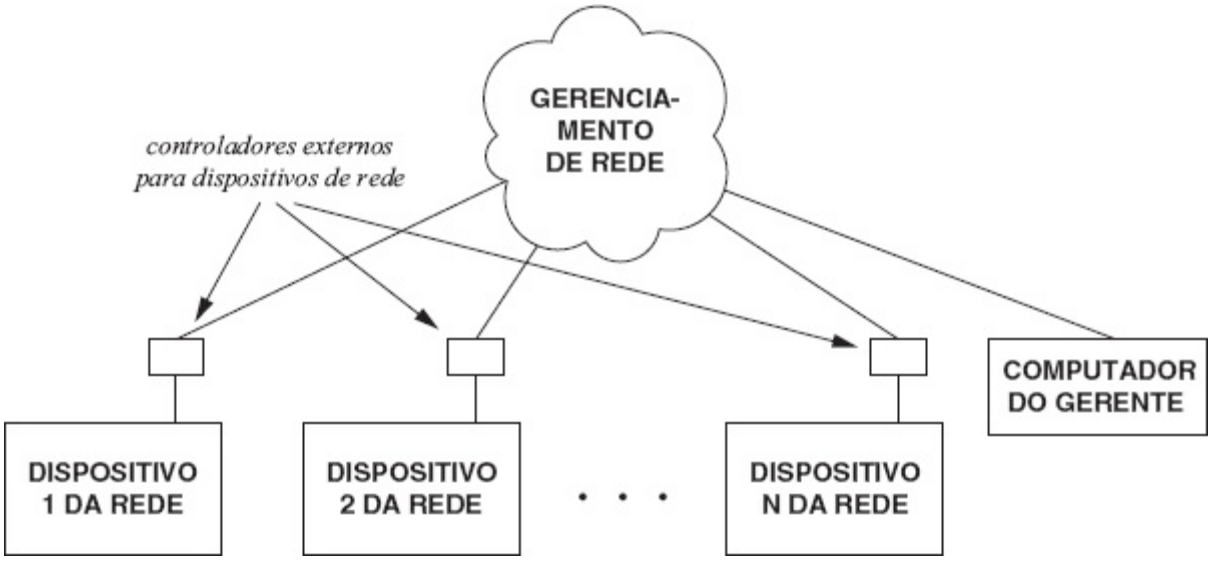


Figura 28.4 Uma interconexão idealizada de controladores SDN em uma rede de gerenciamento separada. Aplicações de gerenciamento nos controladores usam a rede de gerenciamento para se coordenarem uma com a outra.

Como só mostra uma rede de gestão, a figura omite uma parte importante da arquitetura: as redes de dados que estão sendo controladas. Sem ver qualquer uma das redes de dados, pode ser difícil entender por que os controladores precisam ser conectados. A resposta é que os dispositivos de rede que estão sendo controlados compartilham dados das redes. Por exemplo, suponha que o dispositivo 1 e o dispositivo 2 são ambos comutadores VLAN que têm uma conexão direta de dados. Suponha ainda que os dois comutadores estejam colocalizados em um laboratório de uma grande universidade onde eles se conectam um conjunto de computadores. Se um gerente configura uma VLAN para o laboratório, ela deve abranger ambos os comutadores e as configurações devem estar coordenadas. Quando a abordagem SDN é usada, aplicações de gerenciamento em execução nos dois controladores devem se coordenar; para isso, as aplicações de gerenciamento se comunicam através da rede de gestão.

28.9 Implementação SDN com comutadores

convencionais

Talvez o aspecto mais interessante da tecnologia SDN surge a partir da integração do gerenciamento e da rede de dados. O SDN adota a mesma abordagem que o SNMP: o tráfego de gerenciamento viaja ao longo dos mesmos fios do tráfego de dados.* Ou seja, em vez de usar uma rede fisicamente separada, o sistema de gestão utiliza a rede que está sendo gerenciada.

Para entender o paradigma SDN, considere a conexão física entre um comutador Ethernet e o controlador SDN para o comutador. Em vez de usar uma interface de hardware especializada, o controlador pode se conectar a uma porta Ethernet padrão no comutador. Claro que o comutador deve ser configurado para reconhecer o controlador como privilegiado para impedir que aceite comandos SDN a partir de um computador qualquer. A Figura 28.5 ilustra o arranjo mais simples possível: uma conexão direta entre um controlador e um comutador.

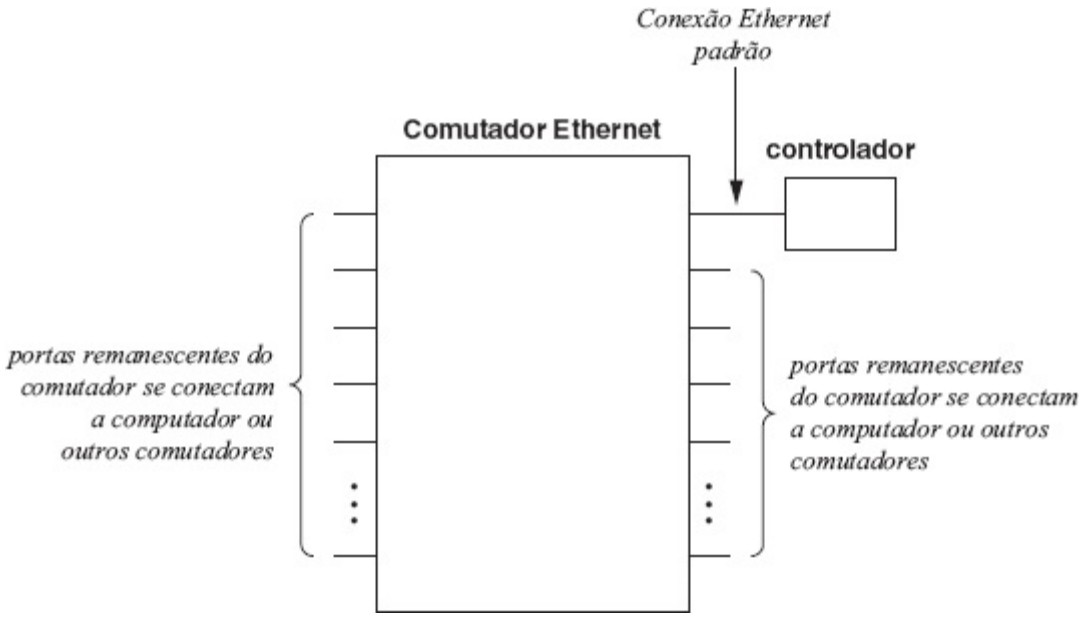


Figura 28.5 Conexão entre um comutador Ethernet e o controlador SDN para o comutador.

A ideia pode ser generalizada. Observe que a maioria das intranets contém vários comutadores. Se imaginarmos uma intranet convencional, as Camadas de encaminhamento 2 e 3 nos comutadores são organizadas de forma a permitir que um controlador conectado a um comutador se comunique com outros comutadores. Por exemplo, a cada comutador será atribuído um endereço IP e o encaminhamento será configurado para que

um computador possa enviar um datagrama IP a qualquer comutador. O SDN assume que tal configuração tenha sido posta em prática antes de o software SDN assumir o controle. Assim, um controlador pode utilizar IP para alcançar qualquer dispositivo de rede. Em essência, a rede de gestão na Figura 28.4 é uma sobreposição virtual em uma intranet convencional em vez de uma rede física separada. Tal como acontece com o SNMP, os gerentes e os aplicativos de gerenciamento usando SDN devem ter o cuidado de preservar a conectividade no gerenciamento de sobreposição – alterações incorretas nas regras de encaminhamento podem deixar um controlador incapaz de se comunicar com um ou mais comutadores.

28.10 Tecnologia OpenFlow

Várias questões se colocam. Exatamente que configuração e recursos de controle deve oferecer um comutador a um controlador SDN externo? Ao enviar um pedido SDN ao comutador, que formato um controlador deve usar? Como um comutador pode distinguir entre solicitações SDN destinadas a ele próprio e outros pacotes que o controlador está enviando (por exemplo, os pacotes enviados para outros controladores)? Em suma, que protocolo deve ser utilizado entre um controlador e um comutador?

A resposta às perguntas encontra-se em um protocolo conhecido como OpenFlow.** Originalmente criado na Universidade de Stanford como uma maneira para os pesquisadores fazerem experiências com novos protocolos de rede, o Open-Flow ganhou maior aceitação. Muitos fornecedores de comutadores concordaram em adicionar capacidade OpenFlow a seus produtos, e implantações maiores dele estão sendo usadas.

Uma vez que o plano de controle, na maioria dos comutadores, executa um sistema operacional, um módulo Open-Flow pode ser adicionado ao interruptor facilmente. O módulo OpenFlow funciona exatamente como o *módulo* SDN mostrado na Figura 28.3* – a maior parte da inteligência está localizada no controlador externo, e o módulo do comutador atua apenas como um intermediário que traduz mensagens provenientes do controlador externo em comandos que são transmitidos para o plano de dados do hardware.

Usamos o termo comutador *OpenFlow* para fazer referência a um comutador que aceita o protocolo OpenFlow. Open-Flow não é um padrão IETF. Em vez disso, a Especificação do Comutador OpenFlow, o documento-padrão central para OpenFlow, é mantido pelo Consórcio OpenFlow:

O OpenFlow fornece uma tecnologia de virtualização de rede. Um comutador OpenFlow pode ser configurado para lidar com o tráfego de rede especializado (incluindo protocolos experimentais fora do padrão) e tráfego de rede de produção simultaneamente. Ele permite a coexistência, sem interferência, de vários tipos de tráfego. Veremos que a presença de uma rede de produção é crucial para OpenFlow, pois permite a um controlador se comunicar com um comutador. Mais importante, mesmo que um comutador tenha regras de encaminhamento convencionais, o OpenFlow pode estabelecer exceções. Por exemplo, se as regras normais de encaminhamento enviam tráfego para IP de destino X para uma determinada porta do comutador, o OpenFlow permite a um administrador especificar que o tráfego IP para X que se origina a partir de IP de origem Y deve ser encaminhado para outra porta do comutador.

28.11 OpenFlow básico

Existem duas versões do protocolo OpenFlow. Um relatório escrito em 2008 descreve a ideia básica e especifica como um comutador OpenFlow básico opera. A especificação OpenFlow, lançada na versão 1.1, expande o modelo, preenche os detalhes do protocolo e inclui funcionalidades adicionais. Nossa investigação de OpenFlow escrevendo o modelo expandido e funcionalidades mais avançadas. Em ambas as versões, básica e avançada, o OpenFlow especifica três aspectos da tecnologia descritos a seguir.

- A comunicação usada entre um controlador e um comutador.
- O conjunto de itens que podem ser configurados e controlados em um comutador.
- O formato de mensagens que um controlador e um comutador usam para se comunicar.

Comunicação. O OpenFlow especifica que um controlador e um comutador usam TCP para se comunicar. Além disso, o OpenFlow especifica que a comunicação deve ocorrer ao longo de um canal de comunicação seguro. Apesar de ser permitida uma conexão TCP, o uso de SSL (descrita no Capítulo 29) é recomendado como uma forma de garantir a confidencialidade de todas as comunicações. É importante lembrar que o OpenFlow não exige uma conexão física direta entre um controlador e um

dispositivo de rede. Em vez disso, ele assume que uma rede de produção estável permanecerá ativa e que um controlador sempre será capaz de usar a rede de produção para se comunicar com o(s) dispositivo(s) de rede que está(ao) sendo controlado(s). O uso do TCP em SSL significa que um único controlador pode estabelecer comunicação com vários dispositivos de rede, mesmo que o controlador não tenha uma conexão física com cada dispositivo.

Itens que podem ser configurados. O OpenFlow especifica que um comutador OpenFlow Tipo 0 (configuração mínima) tem uma tabela de *fluxo* que implementa classificação** e encaminhamento. A parte de classificação de uma tabela de fluxo contém um conjunto de padrões que são confrontados aos pacotes. Na maioria dos comutadores, a correspondência de padrão é implementada com hardware TCAM, mas o OpenFlow permite que um fornecedor escolha uma implementação. Em adição a um padrão, assume-se que cada entrada na tabela contém uma ação que especifica como processar um pacote que combina com o padrão e as *estatísticas* relacionadas à entrada. As estatísticas incluem uma contagem de pacotes que combinam com a entrada, uma contagem de octetos em pacotes que combinam com a entrada e uma estampa de tempo que especifica a última vez em que houve correspondência na entrada. As estatísticas podem ser acessadas por um controlador OpenFlow, sendo úteis para a tomada de decisões de engenharia de tráfego.

Formato das mensagens. Uma seção posterior descreve o formato das mensagens utilizadas com a versão 1.1 do Open-Flow; por enquanto, é suficiente saber que a especificação define o formato exato da mensagem e uma representação para os itens de dados. Por exemplo, o OpenFlow especifica que os inteiros são enviados em uma grande fila indiana.

28.12 Campos específicos em um padrão OpenFlow

Lembre-se, do Capítulo 17, de que os mecanismos de classificação especificam combinações de bits nos cabeçalhos dos pacotes. Assim, é possível especificar valores para campos de cabeçalho arbitrários. Para poupar custos, alguns comutadores só fornecem hardware de classificação para casos específicos (por exemplo, o campo de tag VLAN e endereços IP, mas não cabeçalhos da camada de transporte). Para acomodar comutadores que não oferecem padrão de correspondência arbitrário, o OpenFlow define um conjunto mínimo de requisitos para um comutador Tipo 0. A Figura 28.6 lista os campos a que um comutador OpenFlow deve ser capaz de

corresponder.

Campo	Significado
Ethersrc	**Endereço de origem Ethernet 48-bit**
Etherdst	**Endereço de destino Ethernet 48-bit**
EtherType	**Campo tipo Ethernet 16-bit**
VLAN id	**VLAN tag 12-bit no pacote**
IPv4src	**Endereço de origem IPv4 32 bits**
IPv4dst	**Endereço IPv4 de destino para IPv4 32 bits**
Proto	**Campo protocolo IPv4 8 bits**
TCP/UDP/SCTP src	**Porta origem TCP/UDP/SCTP 16 bits**
TCP/UDP/SCTP dst	**Porta destino TCP/UDP/SCTP 16 bits**

Figura 28.6 Campos nos cabeçalhos dos pacotes que podem ser usados para classificação em comutador OpenFlow Tipo 0.

Os leitores podem se surpreender ao ver que muitos dos campos na Figura 28.6 estão relacionados a protocolos convencionais. Ou seja, os campos permitem ao OpenFlow confrontar o tráfego TCP passando por IPv4 e Ethernet. O conjunto de campos limita experiências? Sim: os campos significam que um comutador OpenFlow Tipo 0 não pode especificar encaminhamento especial para tráfego *ping*, e nem sequer distinguir entre solicitações e respostas ARP. No entanto, mesmo um comutador OpenFlow Tipo 0 permite que pesquisadores usem um tipo de Ethernet não atribuído ou estabeleçam encaminhamento especial para todo o tráfego que chega através de uma determinada porta do comutador. Assim, muitos ensaios são possíveis.

28.13 Ações que OpenFlow pode tomar

Como a Figura 28.7 lista, um comutador OpenFlow Tipo 0 define três ações básicas que podem ser associados com um padrão de classificação:

Ação	Efeito
1	**Encaminhar o pacote para uma determinada porta do comutador ou um determinado conjunto de portas do comutador.**
2	**Encapsular o pacote e enviar para o controlador externo para processamento.**
3	**Descartar o pacote sem qualquer processamento posterior.**

Figura 28.7 Ações possíveis que um comutador OpenFlow tipo 0 pode tomar quando um pacote corresponde a uma das regras de classificação.

A Ação 1 é o caso mais comum. Quando um comutador é inicializado, o software de controle do fornecedor o configura com regras de encaminhamento de tal forma que todos os pacotes que chegam a ele serão encaminhados. Assim, na maioria dos casos, o OpenFlow só tem que configurar exceções. A ideia de reencaminhamento para um conjunto de portas do comutador é usada para implementar broadcast e multicast.

A Ação 2 se destina a permitir que um controlador externo lide com pacotes para os quais nenhum encaminhamento tenha sido estabelecido. Por exemplo, considere como o OpenFlow pode ser utilizado para estabelecer o encaminhamento por unidade de fluxo. O OpenFlow começa especificando a Ação 2 como um padrão para todo o tráfego TCP. Quando o primeiro pacote de uma nova conexão TCP chega, o comutador segue a Ação 2 encapsulando o pacote e encaminhando o resultado para o controlador externo. O controlador pode escolher um caminho para o fluxo TCP, configurar uma regra de classificação no comutador e, em seguida, encaminhar o pacote para o comutador para continuar o processamento (ou seja, para ser encaminhado de acordo com a nova regra de classificação). Todos os pacotes subsequentes na conexão TCP seguem a nova regra de classificação.

A Ação 3 se destina a permitir ao OpenFlow lidar com problemas, como um ataque de negação de serviço ou broadcast excessivo de um determinado host. Quando um problema é detectado, o controlador externo pode identificar a origem e configurar a Ação 3 para ela, o que faz com que o comutador descarte todos os pacotes dessa origem.

Dissemos que, além de encaminhamento experimental, um comutador OpenFlow também suporta uma rede de produção. Surge a pergunta: como o comutador lida com o tráfego de produção? Há duas possibilidades: VLANs especiais são configuradas para OpenFlow e tráfego de produção

ocorre em outras VLANs, ou o OpenFlow inclui uma quarta ação que faz com que um pacote possa ser encaminhado de acordo com as regras definidas para a rede de produção. O Open-Flow permite qualquer abordagem.

28.14 Extensões OpenFlow e adições

A versão 1.1 da especificação OpenFlow estende o modelo básico e adiciona funcionalidade considerável. As adições podem ser classificadas em cinco categorias, que são:

- tabelas de fluxo múltiplo organizadas em uma pipeline;
- disponibilidade de campos de cabeçalho de pacote adicionais para confronto;
- um campo usado para passar informação ao longo da pipeline;
- novas ações que fornecem significativa funcionalidade;
- uma tabela de grupo que permite que um conjunto de ações seja executado.

Uma pipeline de tabelas de fluxo. Na versão 1.1, o modelo subjacente de um comutador foi alterado. Em vez de uma única tabela de fluxo, assume-se que um comutador OpenFlow tenha uma ou mais tabelas de fluxo dispostas em uma pipeline. A confrontação sempre começa com a primeira tabela de fluxo na pipeline. Como antes, cada entrada em uma tabela de fluxo especifica uma ação. Uma ação possível especifica que o processamento deve continuar saltando até o i-ésimo fluxo da tabela, onde i é mais distante ao longo da pipeline. Um salto nunca pode especificar uma tabela de fluxo anterior – cada salto deve mover-se para a frente, o que significa que nunca poderá haver um loop. Quando o pacote chega a uma tabela de fluxo que não especifica um salto, o comutador executa uma ou mais ações, tal como encaminhar o pacote.

Campos adicionais de cabeçalho de pacote. A Versão 1.1 inclui novos protocolos e os campos de cabeçalho que correspondem a cada um. O MPLS é um protocolo novo significativo porque quer dizer que o OpenFlow pode ser utilizado para configurar caminhos MPLS. A Figura 28.8 lista os campos disponíveis para uso com classificação (chamado *campos de correspondência* na especificação).

Como mostra a figura, o OpenFlow não é completamente geral porque não permite que se confrontem todos os campos de cabeçalho possíveis.

Mais importante ainda, o OpenFlow nem sempre permite a um controlador explorar o hardware do comutador subjacente. Para ver o porquê, lembre-se, do Capítulo 17, de que o mecanismo de classificação em muitos comutadores permite que se faça o confronto em campos de bits arbitrários. Espera-se que versões emergentes de OpenFlow tirem vantagem dessa capacidade; em vez de especificar campos de cabeçalho bem conhecidos, um controlador pode especificar cada padrão de classificação como uma tripla:

(posição do bit, tamanho, padrão),

na qual *posição do bit* especifica uma posição arbitrária em um pacote, *tamanho* especifica o tamanho de um campo de bits começando no deslocamento especificado e o *padrão* é uma sequência de bits de *tamanho* que é para ser confrontada.

Comunicação Intrapipeline. O campo denominado *metadados*, na Figura 28.8, não faz parte do pacote. Em vez disso, destina-se a ser utilizado dentro da pipeline. Por exemplo, um estágio da pipeline pode calcular um endereço IPv4 do próximo salto que precisa para passar para uma fase posterior da pipeline junto com o pacote. O OpenFlow não especifica o conteúdo do campo de *metadados* – a pipeline deve ser organizada de modo que seus sucessivos estágios conheçam o conteúdo e o formato da informação que é passada.

Campo	Significado
IngressPort	Porta do comutador pela qual o pacote chegou
Metadata	Campo de 64 bits de metadados usados na pipeline
Ether src	Endereço de origem Ethernet 48-bit
Ether dst	Endereço de destino Ethernet 48-bit
Ether Type	Campo tipo Ethernet 16-bit
VLAN id	VLAN tag 12-bit no pacote
VLAN priority	Número MPLS prioritário VLAN 3-bit
MPLS label	Rótulo MPLS 20-bit
MPLS class	Classe de tráfego MPLS 3-bit
IPv4 src	Endereço de origem IPv4 32-bit
IPv4 dst	Endereço de destino IPv4 32-bit
IPv4 Proto	Campo protocolo IPv4 8-bit
ARP opcode	Opcode ARP 8-bit
IPv4 tos	Bits tipo de serviço IPv4 8-bit
TCP/UDP/SCTP src	Porta origem TCP/UDP/SCTP 16-bit
TCP/UDP/SCTP DST	Porta destino TCP/UDP/SCTP 16-bit
ICMP type	Campo tipo ICMP 8-bit
ICMP code	Campo código ICMP 8-bit

Figura 28.8 Campos disponíveis para uso com a versão 1.1 do OpenFlow.

Novas ações. Na versão 1.1, as ações não são executadas imediatamente quando ocorre uma coincidência. Em vez disso, um conjunto de ações *a ser executado* é acumulado como um pacote de recursos ao longo da pipeline. Cada estágio da pipeline pode adicionar ou remover ações do conjunto. Quando um pacote chega a um estágio de pipeline que não especifica um salto para outra fase (ou seja, o estágio final do pipeline), o OpenFlow executa todas as ações que se acumularam. O OpenFlow requer que o conjunto de ações contenha uma ação de *saída* que determina a disposição final do pacote (por exemplo, a porta de saída pela qual encaminhar).

Além de mudar a forma como as ações são executadas, a versão 1.1 define um conjunto de ações necessárias que cada comutador OpenFlow deve suportar e uma longa lista de ações opcionais recomendada, mas não obrigatória. Por exemplo, a especificação inclui ações que manipulam o TTL em um cabeçalho de datagrama. Outra ação permite ao OpenFlow encaminhar um pacote para a pilha de protocolos do comutador local. Muitas das novas ações se destinam a suportar MPLS. Em particular, um comutador pode precisar receber um datagrama IP de entrada, encapsulá-lo em um cabeçalho shim MPLS, colocar o resultado em um frame Ethernet e encaminhá-lo. Da mesma forma, um comutador pode precisar descapsular um datagrama quando um pacote MPLS chega e, portanto, o datagrama pode ser enviado sem o cabeçalho shim MPLS. Finalmente, porque a versão 1.1 inclui filas QoS, ações foram definidas permitindo a um comutador colocar um pacote em uma fila particular.

Uma tabela de grupo. Uma tabela de grupo adiciona flexibilidade ao paradigma de encaminhamento e lida com vários casos. Por exemplo, uma entrada em uma tabela de grupo pode especificar como encaminhar pacotes, e várias regras de classificação podem direcionar os fluxos para a entrada. Alternativamente, uma entrada em uma tabela de grupo pode especificar o encaminhamento a *qualquer* porta do comutador em um determinado conjunto. A seleção é realizada de uma maneira de dispositivo específico (por exemplo, o comutador calcula um valor hash dos campos de cabeçalho e usa o valor para selecionar uma das portas especificadas). A ideia é que um conjunto de portas do comutador, todas conectando ao mesmo próximo salto, pode agir como uma única conexão de alta capacidade.

Em cada entrada da tabela de grupo, há quatro itens: um *identificador* de 32 bits que identifica o grupo, um *tipo*, um conjunto de *contadores* que coleta estatísticas, e um conjunto de *planos de ação* (*Action Buckets*) em que cada um especifica uma ação. A Figura 28.9 lista os possíveis tipos de grupos.

Um tipo rápido de failover fornece uma forma limitada de execução condicional em que um controlador pode configurar um conjunto de planos (buckets), sendo que cada um especifica uma ação de encaminhamento. Cada plano está associado a um *liveness test*. Por exemplo, o comutador pode monitorar portas de saída e declarar que uma porta não está mais viva se um carrier foi perdido (por exemplo, o dispositivo está desligado). Uma vez que um determinado plano já não está mais ativo, uma rápida seleção

failover vai descartá-lo e tentar uma alternativa. A ideia é evitar o atraso que haveria se cada perda de liveness fizesse o comutador informar ao controlador externo e este tivesse que reconfigurar o encaminhamento.

Tipo	Significado
all	**Executa todos os planos de ação para o grupo (por exemplo, para lidar com broadcast).**
select	**Executa um plano de ação (por exemplo,usar um algoritmo hash ou round-robin para seleção).**
indirect	**Executa o único plano de ação definido para o grupo (projetado para permitir múltiplas entradas de tabela de fluxo para indicar para um único grupo).**
fast fail-over	**Executa o primeiro plano de ação ativo (ou descarta o pacote se não houver plano ativo).**

Figura 28.9 Os quatro tipos de grupo OpenFlow e o significado de cada um.

28.15 Mensagens OpenFlow

Uma mensagem de OpenFlow começa com um cabeçalho de tamanho fixo que compreende 16 octetos. A Figura 28.10 ilustra o formato da mensagem.

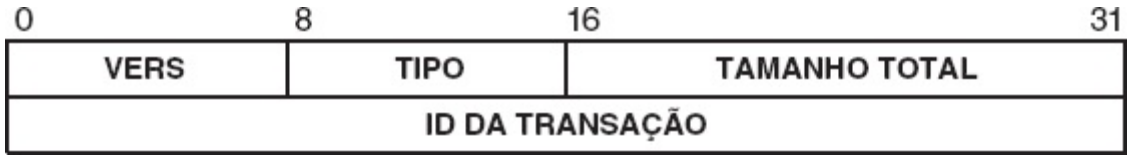


Figura 28.10 O cabeçalho de tamanho fixo usado para todas as mensagem OpenFlow.

O campo VERS é um número de versão (por exemplo, o valor 0x02 especifica versão 2). O campo TIPO especifica o tipo da mensagem que segue. O OpenFlow define 24 tipos de mensagem. O campo TAMANHO TOTAL especifica o tamanho total da mensagem, incluindo o cabeçalho, medido em octetos. O ID DA TRANSAÇÃO é um valor único que permite que o controlador case respostas com pedidos.

O formato do restante da mensagem além do cabeçalho é determinado pelo tipo de mensagem. Embora um exame de todos os formatos de mensagem esteja fora do escopo de nossa revisão, deve-se notar que

existem várias categorias de mensagens: controlador para comutador, assíncronas (o comutador informa o controlador quando um evento ocorreu) e simétricos (por exemplo, uma requisição de eco e resposta). O leitor deve consultar a especificação OpenFlow para mais informações.

28.16 Usos do OpenFlow

Como o OpenFlow pode ser usado? O OpenFlow permite que um gerente configure encaminhamento como uma função de origem, de destino ou de campos de tipo em um cabeçalho de pacote. Assim, são possíveis várias configurações. Alguns exemplos a seguir ilustram algumas das possibilidades.

- Protocolo experimental usado entre dois hosts.
- VLAN de Camada 2 que cruza atravessa uma área remota.
- Encaminhamento IP com base na origem.
- Conexão VPN sob demanda entre dois sites.

Como o OpenFlow pode usar o campo tipo de Ethernet na tomada de decisões de encaminhamento, este pode permitir que dois hosts troquem pacotes Ethernet que usam um protocolo de camada 3 diferente do padrão. O OpenFlow também pode reconhecer o ID VLAN atribuído aos pacotes, e usar ID VLAN para criar uma VLAN que estende um par de comutadores através de uma área remota. Ao contrário de Ethernet convencional ou encaminhamento de IP, o OpenFlow é capaz de examinar os campos de endereço de origem em um pacote, e a combinação de endereços de origem e destino pode ser usada na escolha de uma rota. Assim, o tráfego para um determinado destino a partir do host de origem A pode ser enviado por um caminho diferente do que o tráfego a partir do host de origem *B*. Como exemplo final, considere um túnel VPN. Como a versão 1.1 permite o encapsulamento, o OpenFlow pode criar um túnel VPN quando o primeiro pacote de uma conexão TCP aparece.

A lista não é exaustiva porque muitos arranjos são possíveis. No entanto, a chave para entender o OpenFlow não está em pensar em configurações ou regras de encaminhamento. Em vez disso, o ponto importante é que o OpenFlow permite que pesquisadores criem software de gerenciamento de rede personalizado que possa coordenar as ações de vários controladores. Assim, o OpenFlow pode ser usado para fazer vários comutadores agirem de uma forma consistente.

28.17 OpenFlow: entusiasmo, publicidade e limitações

A comunidade científica tem abraçado o OpenFlow com entusiasmo selvagem. Muitos pesquisadores universitários estão ansiosos para adotá-lo. Aqueles que não têm acesso ao hardware do comutador estão realizando experiências com emuladores de software. Reuniões de pesquisa inteiras e conferências são dedicadas a trabalhos e apresentações sobre o OpenFlow. Empresas iniciantes estão sendo formadas. Na empolgação, muitas das reivindicações tendem ao exagero. Um pesquisador orgulhosamente anunciou que o OpenFlow iria "quebrar as correntes" que estão nos prendendo ao gerenciamento de rede fornecido pelo vendedor.

Apesar da badalação, vimos que o OpenFlow não cumpre totalmente a meta SDN de tornar todos os dispositivos totalmente configuráveis. Em vez disso, ele seleciona um pequeno subconjunto de possíveis dispositivos, funções e capacidades. Os próximos parágrafos destacam algumas limitações.

Dispositivos limitados. O OpenFlow não permite que um gerente controle dispositivos arbitrários. Embora tenha sido utilizado com um punhado de dispositivos, tais como pontos de acesso, o trabalho preliminar no OpenFlow tem sido dirigido aos comutadores.

Somente Ethernet. Como vimos, a versão 1.1 da especificação OpenFlow concentra-se em frames Ethernet. Em particular, o OpenFlow define padrões de correspondência para campos em cabeçalhos de frame Ethernet, mas não para outros frames. Embora a Ethernet esteja generalizada, nenhum sistema global está completo sem Wi-Fi ou sem o framing usado em circuitos digitais.

Foco no IPv4. Originalmente, a versão 1.1 do OpenFlow se concentrou exclusivamente no IPv4 e nos protocolos de suporte associados, tais como ARP e ICMP. A especificação foi mais tarde estendida para incluir correspondência para IPv6 e seus protocolos associados (por exemplo, descoberta de vizinho). O foco no IPv4 foi especialmente surpreendente, dado o propósito original do OpenFlow: incentivar a investigação. Versões emergentes incluem suporte para mais protocolos, incluindo o IPv6.

Dadas as limitações, muitos fornecedores de rede têm tido uma visão mais sóbria do OpenFlow. Eles apontam que a falta de generalidade limita a aplicabilidade do OpenFlow. No entanto, muitos fornecedores de comutador concordaram em adicionar um módulo OpenFlow em seus produtos. Consequentemente, a comoção que o envolve permanece elevada.

28.18 Software Defined Radio (SDR)

A ideia básica em rede definida por software – a separação de dados e planos de controle – também foi aplicada a redes sem fio. O resultado é conhecido como *Rádio definido por software* (*Software Defined Radio – SDR*). Um dos componentes-chave no SDR é um dispositivo de rádio flexível (ou seja, um chip). Ao contrário de um projeto convencional, em que detalhes como a frequência de utilização e da modulação do sinal estão no hardware do equipamento, um rádio SDR permite que os detalhes sejam especificados enquanto o aparelho está em funcionamento. Mais importante, a configuração pode ser alterada facilmente, e o rádio pode lidar com múltiplas frequências ao mesmo tempo. Assim, é possível configurar o dispositivo de rádio para mudar de uma frequência para outra (Frequency Hopping) ou utilizar duas frequências ao mesmo tempo. Um par de rádios SDR pode pesquisar um conjunto de frequências, encontrar uma que não esteja em uso e concordar em usá-la.

Uma utilização óbvia de um chip de rádio configurável é que um vendedor pode usar o mesmo chip em vários produtos e simplesmente configurar cada um de acordo com as necessidades. No entanto, a principal vantagem da SDR é que um sistema de rádio pode ser criado para permitir que softwares aplicativos configurem o chip de forma dinâmica. Ou seja, o software aplicativo em execução em um dispositivo de rádio pode detectar as frequências que estão sendo usadas e, então, ajustar o chip de acordo com elas. Assim, é possível que dois dispositivos SDR encontrem maneiras de comunicar que maximizem o rendimento e minimizem a interferência com outros rádios.

Uma das limitações da tecnologia SDR é a limitação de antenas. Em SDN, a configuração meramente escolhe caminhos para os pacotes, mas o SDR lida com parâmetros da Camada 1. Quando a frequência muda, a antena necessária para transmitir ou receber radiação eletromagnética também muda. Projetos experimentais SDR têm usado uma variedade de técnicas, inclusive limitando as frequências a uma faixa específica, usando antenas múltiplas que cobrem, cada uma, um conjunto de frequências, e usando antenas em uma nova forma que permita adicionar a radiação a partir de várias pequenas antenas para alcançar a mesma capacidade de uma antena grande.

28.19 Resumo

O Software Defined Networking separa processamento de plano de controle de processamento de plano de dados e muda o processamento de controle

para um controlador externo (por exemplo, um PC). Em essência, a inteligência está localizada no controlador externo; um módulo minimalista em um dispositivo de rede aceita solicitações do controlador e configura o plano de dados do dispositivo em conformidade.

A principal tecnologia SDN é conhecida como OpenFlow. O OpenFlow, que foi projetado para permitir que os pesquisadores fizessem experiências com novos protocolos e aplicativos de gerenciamento de rede, usa os mecanismos de classificação e encaminhamento no plano de dados de um comutador Ethernet. O relatório original descreve uma versão básica do OpenFlow que lida com Ethernet, IPv4 e TCP; a versão 1.1 da especificação OpenFlow estende o modelo para ter uma pipeline de tabelas de fluxo, muitas novas ações e mensagens e uma Tabela de Grupo que permite que várias portas de saída operem como um pacote paralelo (paralell bundle) e rápido failover. Embora a comunidade de pesquisa esteja entusiasmada, vendedores apontam limitações do OpenFlow.

EXERCÍCIOS

28.1 Se a sua organização usa uma tecnologia SDN, descobrir o porquê.

28.2 Uma pipeline de tabelas de fluxo pode ser implementada utilizando-se metadados e iterativamente buscando uma única tabela. Mostre que qualquer processamento de pacote que pode ser alcançado por uma pipeline também pode ser tratado por pesquisa iterativa.

28.3 Leia atentamente a especificação OpenFlow. Como você a usaria para criar uma tabela de encaminhamento IP convencional?

28.4 Suponha que dois pesquisadores que têm computadores host conectados a um comutador OpenFlow querem fazer uma experiência com um protocolo diferente do padrão da Camada 3. Quais são as possíveis maneiras de configurar o comutador para suportar a experiência? Quais são as vantagens e desvantagens de cada uma?

28.5 Leia o tutorial no site do OpenFlow e aprenda como ele lida com ARP.

28.6 Leia atentamente a especificação OpenFlow. Um conjunto de controladores pode ser organizado em uma hierarquia? Se sim, qual é a razão? Se não, por quê?

28.7 Qual é o propósito do tipo *indireto* em uma entrada de tabela de grupo?

28.8 Enquanto um pacote viaja ao longo de uma pipeline OpenFlow, cada

estágio da pipeline pode gravar uma ou mais ações a serem tomadas no pacote ou pode remover uma ou mais ações que foram especificadas em etapas anteriores. Encontre um exemplo em que remover uma ação é útil.

28.9 O OpenFlow inclui uma ação local, que entrega um pacote para a pilha de protocolos do comutador local, e uma ação NORMAL, que faz com que o pacote possa ser encaminhado de acordo com as regras de encaminhamento tradicionais no comutador. Se o encaminhamento tradicional inclui a entrega para a pilha do comutador, por que ambas as ações são necessárias?

28.10 A versão 1.1 do OpenFlow permite uma entrada de tabela de fluxo para verificar os bits Tipo de Serviço (Type-Of-Service – TOS) em um cabeçalho IP? A versão 1.1 permite uma ação que define os bits TOS? Explique.

28.11 Consulte o site da *Fundação Open Networking*

www.opennetworking.org

para encontrar a versão mais recente da especificação do protocolo OpenFlow. Faça uma lista dos novos recursos mais significativos que foram adicionados.

---

* O Capítulo 27 cobre o SNMP e explica como o tráfego de gerenciamento é executado na rede de dados que está sendo gerenciada.

** O web site *http://www.openflow.org/wk/index.php/OpenFlow_Tutorial* contém um tutorial, e *www.opennetworking.org* tem os documentos padrão do OpenFlow.

* A Figura 28.3 pode ser encontrada na página 403.

** Ver Capítulo 17 para uma discussão sobre classificação.

CAPÍTULO 29

Segurança na Internet e projeto de firewall (IPsec, SSL)

CONTEÚDOS DO CAPÍTULO

29.1 Introdução

A segurança em um ambiente de internet é importante e difícil. Importante na medida em que as informações tem um valor significativo – elas podem ser compradas ou vendidas diretamente ou, então, usadas indiretamente para criar artefatos valiosos. E difícil porque implica entender quando e como os usuários, computadores, serviços e redes participantes podem confiar um no outro, além de entender os detalhes técnicos do hardware e dos protocolos de rede. Portanto, um único ponto fraco pode comprometer a segurança da rede inteira. Mais importante, como o TCP/IP admite uma expressiva diversidade de usuários, serviços e redes, e como uma rede pode espalhar-se por muitos limites políticos e organizacionais, os indivíduos e as organizações participantes também podem não concordar com um nível de confiança ou com políticas para lidar com os dados.

Este capítulo considera duas técnicas fundamentais que formam a base para a segurança da internet: segurança do perímetro e criptografia. Explicando: a segurança do perímetro permite a uma organização determinar os serviços e as redes que estarão disponíveis aos de fora. A criptografia, por sua vez, trata dos outros aspectos da segurança. Vamos começar revendo alguns conceitos básicos e a terminologia.

29.2 Protegendo recursos

Os termos *segurança de rede* e *segurança da informação* referem-se, em grande parte, ao fato de que a segurança na informação e os serviços disponíveis em uma rede não podem ser acessados por usuários não autorizados. A segurança implica proteção, incluindo a garantia de integridade dos dados, impedimento de acesso não autorizado aos recursos computacionais, impedimento de escutas ou grampos e impedimento de interrupção de serviço. Claro está que, assim como nenhuma propriedade física é absolutamente segura contra o crime, nenhuma rede é também completamente segura. Portanto, as organizações esforçam-se por proteger as redes pelo mesmo motivo pelo qual se esforçam para proteger prédios e escritórios: medidas de segurança básica podem desencorajar o crime, tornando-o significativamente mais difícil.

Fornecer segurança às informações exige que os recursos físicos e abstratos sejam protegidos. Os recursos físicos incluem dispositivos de armazenamento passivo, tais como discos, além de dispositivos ativos, como os computadores dos usuários e os smartphones. Em um ambiente de rede, a segurança física estende-se aos cabos, comutadores e roteadores que formam a infraestrutura da rede. Na realidade, ainda que raramente mencionado, a segurança física desempenha um importante papel em um plano de segurança geral. Uma efetiva segurança física pode eliminar sabotagem (por exemplo, a desativação de um roteador para fazer com que os pacotes sejam encaminhados por um caminho alternativo menos seguro).

A proteção de um recurso abstrato, como a informação, é geralmente mais difícil do que oferecer segurança física, tendo em vista ser a informação esquiva. A segurança da informação envolve muitos aspectos da proteção conforme descritos a seguir.

- *Integridade de dados.* Um sistema seguro precisa proteger as informações contra mudança não autorizada.
- *Disponibilidade de dados.* O sistema precisa garantir que estranhos não possam impedir o acesso legítimo aos dados (por exemplo, nenhum estranho deverá impedir que os clientes acessem um site).
- *Privacidade ou confidencialidade.* O sistema deverá evitar que estranhos façam cópias dos dados enquanto passam por uma rede, ou mesmo que entendam o conteúdo se cópias forem feitas.
- *Autorização.* Mesmo que a segurança física normalmente classifique as pessoas e os recursos em categorias gerais (por exemplo, os não

empregados são proibidos de usar determinado corredor), a segurança para a informação normalmente precisa ser mais restritiva (por exemplo, algumas partes do registro dos funcionários estão disponíveis apenas ao pessoal do escritório, enquanto outras estão disponíveis apenas à chefia do empregado, e outras, apenas ao setor de folha de pagamento).

- *Autenticação.* O sistema precisa permitir que duas entidades de comunicação validem a identidade uma da outra.
- *Impedimento de replay.* Visando evitar que estranhos capturem cópias dos pacotes e as utilizem mais tarde, o sistema precisa impedir que seja aceita uma cópia retransmitida de um pacote.

29.3 Política de informação

Antes que uma organização possa impor a segurança da rede, ela deve avaliar os riscos e desenvolver uma política clara em relação ao acesso e à proteção da informação. A política especifica quem receberá o acesso a cada parte da informação, as regras que um indivíduo precisa seguir na disseminação da informação para outros e a declaração de como a organização reagirá às violações.

Uma política de informação se inicia com as pessoas porque:

> *os humanos são, em geral, o ponto mais suscetível de qualquer esquema de segurança. Um trabalhador que seja malicioso, descuidado ou desavisado acerca da política de informação de uma organização pode comprometer a segurança.*

29.4 Segurança da Internet

A segurança da Internet é difícil porque os datagramas que trafegam da origem ao destino normalmente passam por muitas redes intermediárias e por roteadores que não pertencem nem são controlados pelo emissor ou pelo receptor. Portanto, como os datagramas podem ser interceptados ou comprometidos, o conteúdo pode não ser confiável. Como exemplo, considere um servidor que tenta usar a *autenticação da origem* para verificar se as requisições foram originadas de clientes válidos. A autenticação da origem requer que o servidor examine o endereço IP de origem em cada datagrama que chega e só aceite requisições de

computadores em uma lista autorizada. Por ser facilmente quebrada, a autenticação da origem é *fraca*. Em particular, um roteador intermediário pode observar o tráfego que atravessa de/para o servidor e registrar o endereço IP de um cliente válido. Mais tarde, o roteador intermediário pode preparar uma requisição que tenha o mesmo endereço de origem (e interceptar a resposta). O ponto importante é este a seguir.

> *Um esquema de autorização que usa o endereço IP de uma máquina remota para autenticar sua identidade não é suficiente em uma internet desprotegida. Um impostor que consiga o controle de um roteador intermediário pode obter acesso personificando um cliente autorizado.*

Uma autenticação forte exige *criptografia*. Escolhas cuidadosas de um algoritmo de criptografia e chaves associadas podem tornar praticamente impossível que máquinas intermediárias decodifiquem mensagens ou preparem mensagens que sejam válidas.

29.5 IP Security (IPsec)

O *IETF* criou um conjunto de protocolos que oferece comunicação segura pela Internet. Coletivamente conhecidos como *IPsec* (abreviação de IP security), eles oferecem serviços de autenticação e privacidade na camada IP e podem ser usados com IPv4 e IPv6. E, mais importante, em vez de especificar completamente a funcionalidade do algoritmo de criptografia a ser usado, o IETF escolheu tornar o sistema flexível e extensível. Por exemplo, uma aplicação que emprega IPsec pode escolher se usará uma facilidade de autenticação que valida o emissor ou, então, uma facilidade de criptografia que também garante que o payload permanecerá confidencial. As opções podem ser assimétricas em cada direção (por exemplo, uma extremidade pode optar por usar autenticação quando envia datagramas, e o outro terminal pode optar por enviar datagramas sem autenticação). Além do mais, o IPsec não restringe o usuário a uma criptografia específica ou um algoritmo de autenticação. Em vez disso, ele fornece uma estrutura geral que permite que cada par de extremidades em comunicação escolha algoritmos e parâmetros, tais como o tamanho de uma

chave. Para garantir a interoperabilidade, o IPsec inclui um conjunto de algoritmos de criptografia que todas as implementações precisam reconhecer. O ponto importante é descrito a seguir.

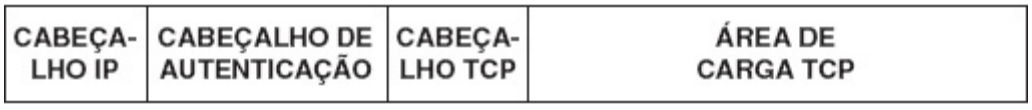
> *O IPsec não é um protocolo de segurança isolado. Em vez disso, ele oferece um conjunto de algoritmos de segurança e também uma estrutura geral que permite a um par de entidades de comunicação usar quaisquer algoritmos que ofereçam segurança apropriada para a comunicação.*

29.6 Cabeçalho de autenticação IPsec

O IPsec segue a abordagem básica que tem sido adotada para o IPv6: um cabeçalho de autenticação (*Authentication Header – AH*) separado para transportar informações de autenticação. Curiosamente, o IPsec aplica a mesma abordagem para o IPv4. Desse modo, em vez de modificar o cabeçalho IPv4, o IPsec insere um cabeçalho extra no datagrama. Pensamos em inserir um cabeçalho porque a autenticação pode ser adicionada como um último passo, após o datagrama ter sido criado. A Figura 29.1 ilustra um cabeçalho de autenticação sendo inserido em um datagrama que transporta o TCP.

Como a figura mostra, o IPsec insere o cabeçalho de autenticação imediatamente após o cabeçalho IP original, mas antes do cabeçalho de transporte. Para o IPv4, o IPsec modifica o campo de PROTOCOLO (protocol) no cabeçalho IP para especificar que o payload é autenticação; para o IPv6, o IPsec usa o PRÓXIMO campo de CABEÇALHO. Em ambos os casos, o valor 51 indica que o item a seguir do cabeçalho IP é um cabeçalho de autenticação.

(a)



(b)

Figura 29.1 Ilustração de (a) um datagrama IP e (b) o mesmo datagrama após um cabeçalho de autenticação IPsec. A abordagem é utilizada para IPv4 e IPv6.

Um cabeçalho de autenticação não tem um tamanho fixo. Em vez disso, começa com um conjunto de campos de tamanho fixo que descreve as informações de segurança que estão sendo realizadas, seguido por uma área de tamanho variável, que depende do tipo de autenticação a ser utilizado. A Figura 29.2 ilustra o formato do cabeçalho de autenticação.

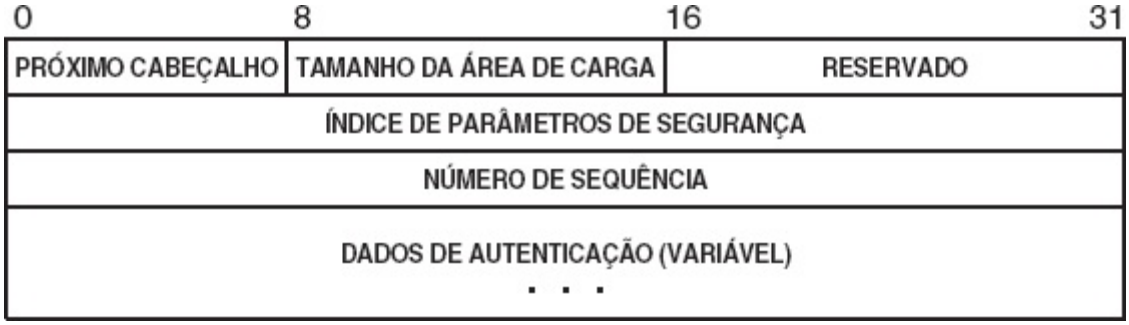


Figura 29.2 O formato do cabeçalho de autenticação IPsec. O mesmo formato é usado com IPv4 e IPv6.

Como um receptor determina o tipo de informação transportada no datagrama? O cabeçalho de autenticação tem seu próprio campo de CABEÇALHO PRÓXIMO que especifica o tipo – o IPsec registra o valor do protocolo original no campo de CABEÇALHO PRÓXIMO do cabeçalho de autenticação. Quando um datagrama chega, o receptor usa as informações de segurança do cabeçalho de autenticação, a fim de verificar o remetente, e usa o campo de CABEÇALHO PRÓXIMO no cabeçalho de autenticação para desmultiplexar ainda mais o datagrama.

A ideia de inserir um cabeçalho extra e o conceito de um campo de CABEÇALHO PRÓXIMO em cada cabeçalho foram originalmente projetados para uso em IPv6. Quando o IPsec foi retroequipado do IPv6 para o IPv4, os designers optaram por manter a abordagem geral, ainda que não seguisse o modelo habitual IPv4.

Curiosamente, o campo TAMANHO DO PAYLOAD (PAYLOAD LEN) não especifica o tamanho da área payload final no datagrama. Em vez disso, ele especifica o tamanho do próprio cabeçalho de autenticação. Desse modo, um receptor será capaz de saber onde o cabeçalho de autenticação termina, mesmo que ele não entenda o esquema de autenticação específico que está sendo usado. O campo NÚMERO DE SEQUÊNCIA (SEQUENCE

NUMBER) contém um número de sequência exclusivo para cada pacote enviado; o número começa com zero quando determinado algoritmo de segurança é selecionado e aumenta monotonicamente. O campo ÍNDICE DE PARÂMETROS DE SEGURANÇA (SECURITY PARAMETERS INDEX) especifica o esquema de segurança utilizado, e o campo DADOS DE AUTENTICAÇÃO (AUTHENTICATION DATA) contém os dados para o esquema de segurança selecionado.

29.7 Associação de segurança

A fim de entender o motivo para usar um índice de parâmetro de segurança, observe que um esquema de segurança define os detalhes que oferecem muitas variações possíveis. Por exemplo, o esquema de segurança inclui um algoritmo de identificação, uma chave (ou chaves) que o algoritmo utiliza, um tempo de vida durante o qual a chave permanecerá válida, um tempo de vida durante o qual o destino concorda em usar o algoritmo e uma lista de endereços de origem autorizada a usar o esquema. Além disso, observe que a informação pode não caber no cabeçalho.

Para economizar espaço no cabeçalho, o IPsec providencia que cada receptor colete todos os detalhes sobre um esquema de segurança em uma abstração conhecida como associação de segurança (*Security Association – AS*). Cada SA recebe um número, conhecido como índice de parâmetros de segurança (*security parameters index*), por meio do qual ela é identificada. Assim, antes que um emissor possa usar o IPsec para se comunicar com o receptor, ele precisa saber o valor de índice para determinada SA. O emissor, então, coloca o valor no campo ÍNDICE DE PARÂMETROS DE SEGURANÇA de cada datagrama de saída.

Os valores de índice não são especificados globalmente. Em vez disso, cada destino cria tantas SAs quantas precisar e atribui um valor de índice a cada uma. O destino pode não só especificar um tempo de vida para cada SA, mas também reutilizar os valores de índice quando uma SA se tornar inválida. Consequentemente, o índice não pode ser interpretado sem consultar o destino (por exemplo, o índice 1 pode ter significados diferentes para dois destinos). Podemos fazer o resumo a seguir.

O destino usa o índice de parâmetros de segurança a fim de identificar a associação de segurança para um pacote. Os valores não são globais; uma combinação de endereço de destino e índice de parâmetros de

segurança é necessária para identificar uma SA.

29.8 Payload de segurança no encapsulamento IPsec

Para lidar com a privacidade e também a autenticação, o IPsec utiliza um *Encapsulating Security Payload* (*ESP*), que é mais complexo do que um cabeçalho de autenticação. Em vez de inserir um cabeçalho extra, o ESP requer um emissor para substituir o IP payload com uma versão criptografada de carga útil. O receptor decodifica o payload e recria o datagrama original.

Tal como ocorre com a autenticação, o IPsec define o PRÓXIMO CABEÇALHO (IPv6) ou campo de PROTOCOLO (IPv4) no cabeçalho de IP, para indicar que foi usado ESP. O valor escolhido é 50. Um cabeçalho ESP tem um campo de PRÓXIMO CABEÇALHO que especifica o tipo do payload original. A Figura 29.3 ilustra como o ESP modifica o datagrama.

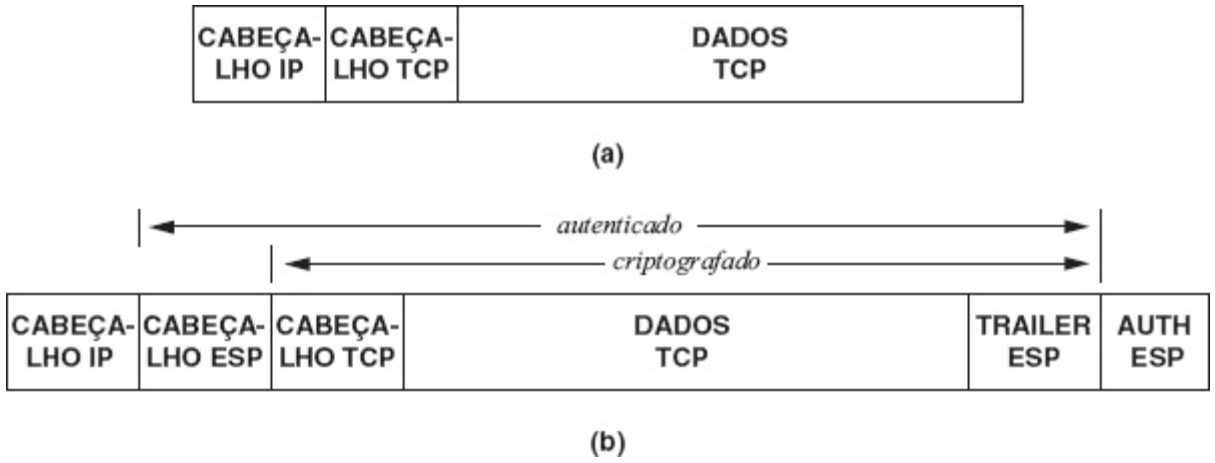
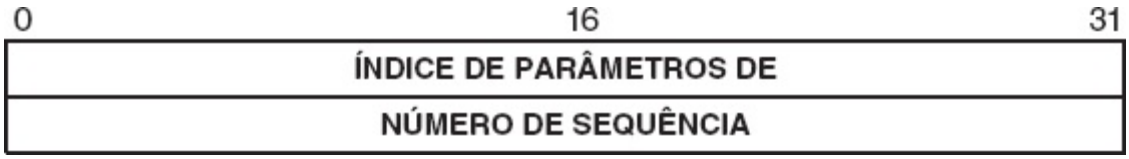


Figura 29.3 (a) Um datagrama e (b) o mesmo datagrama usando IPsec Encapsulating Security Payload. Na prática, a criptografia significa que os campos não são facilmente identificáveis.

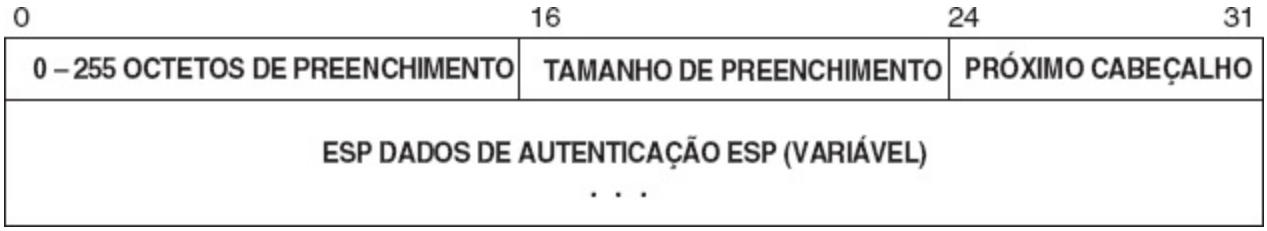
Como mostra a figura, o ESP acrescenta três áreas adicionais ao datagrama. O CABEÇALHO ESP vem imediatamente após o cabeçalho IP e precede o payload criptografado. O TRAILER ESP é criptografado junto com o payload. Por fim, um campo AUTENT ESP de tamanho variável vem imediatamente após a seção criptografada. Por que a autenticação está presente? A ideia é que o ESP não é uma alternativa para autenticação, mas deve ser um complemento. Portanto, a autenticação é uma parte necessária do ESP.

Embora represente com precisão o uso do IPsec com IPv4, a Figura 29.3 ilustra um conceito importante em IPv6: múltiplos cabeçalhos. No caso mais simples, um datagrama IPv6 pode ser estruturado exatamente como na figura, com uma base de cabeçalho IPv6 seguida por um cabeçalho TCP e TCP payload. No entanto, o conjunto de cabeçalhos opcionais IPv6 inclui cabeçalhos salto a salto, processados por roteadores intermediários. Por exemplo, o datagrama pode conter um cabeçalho de rota de origem que especifica um conjunto de pontos intermediários ao longo de um caminho até o destino. Se o ESP criptografasse todo o datagrama após o cabeçalho IPv6 de base, as informações salto a salto estariam indisponíveis para os roteadores. Portanto, o ESP é aplicado apenas aos itens que seguem os cabeçalhos salto a salto.

Os cabeçalhos ESP utilizam muitos dos itens encontrados no cabeçalho de autenticação, mas reorganizam sua ordem. Por exemplo, o CABEÇALHO ESP consiste em oito octetos que identificam os índices de parâmetros de segurança e um número de sequência.



O ESP TRAILER consiste em um preenchimento opcional, um campo de tamanho de preenchimento, TAMANHO DE PREENCHIMENTO (PAD LENGTH), e um plano de PRÓXIMO CABEÇALHO, que é seguido por uma quantidade variável de dados de autenticação



O preenchimento é opcional; ele pode estar presente por três razões. Primeiro, alguns algoritmos de descriptografia exigem zeros após uma mensagem criptografada. Segundo, observe que o campo PRÓXIMO CABEÇALHO aparece alinhado à direita dentro de um campo de 4 octetos. O alinhamento é importante porque o IPsec exige que os dados de autenticação que vêm após o término sejam alinhados no início de um limite de 4 octetos. Assim, o preenchimento pode ser necessário para garantir o alinhamento. Terceiro, alguns sites podem decidir acrescentar

quantidades aleatórias de preenchimento a cada datagrama, de modo que os bisbilhoteiros em pontos intermediários ao longo do caminho não possam usar o tamanho de um datagrama para adivinhar sua finalidade.

29.9 Campos de autenticação e cabeçalho mutável

O mecanismo de autenticação IPsec foi projetado para garantir que um datagrama que chega seja idêntico ao datagrama enviado pela origem. Porém, essa garantia é impossível. Para entender por que, lembre-se de que o IP é uma camada de máquina para máquina, porque o princípio de camadas só se aplica por um salto. Em particular, cada roteador intermediário decrementa o limite de salto (IPv6) ou o campo de tempo de vida e recalcula o checksum.

O IPsec usa o termo campos mutáveis para se referir aos campos do cabeçalho IP que são alterados em trânsito. Para evitar que tais mudanças causem erros de autenticação, o IPsec omite especificamente os campos mutáveis do cálculo de identificação (por exemplo, o endereço de origem e o tipo de protocolo).

29.10 Tunelamento IPsec

Lembre-se de que, conforme visto no Capítulo 19, a tecnologia VPN usa a criptografia junto com o tunelamento IP-em-IP para manter privativas as transferências entre sites. O IPsec é projetado especificamente para acomodar um túnel criptografado. Em particular, o padrão define versões tuneladas do cabeçalho de autenticação e do payload de segurança de encapsulamento. A Figura 29.4 ilustra o layout dos datagramas no modo de tunelamento.

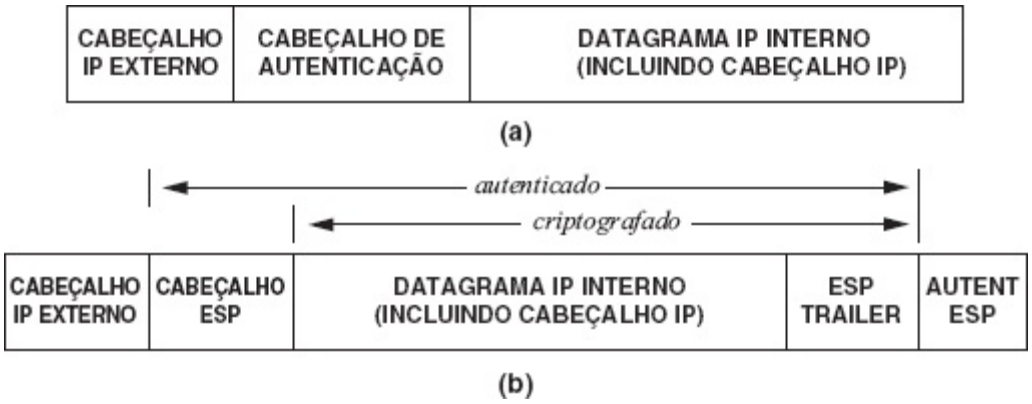


Figura 29.4 Ilustração do modo de tunelamento IPsec para (a) autenticação e (b) encapsulamento do payload de segurança. O datagrama interno inteiro é protegido.

29.11 Algoritmos de segurança necessários

O IPsec define um conjunto mínimo de algoritmos que são obrigatórios (ou seja, que todas as implementações precisam fornecer). Em cada caso, o padrão define usos específicos. A Figura 29.5 lista os algoritmos exigidos.

Autenticação	
HMAC com MD5	**RFC 2403**
HMAC com SHA-1	**RFC 2404**
Encapsulamento de Segurança de Carga (Encapsulating Security Payload)	
DES no modo CBC	**RFC 2405**
HMAC com MD5	**RFC 2403**
HMAC com SHA-1	**RFC 2404**
Autenticação Nula	
Criptografia Nula	

Figure 29.5 Os algoritmos de segurança que são obrigatórios para IPsec.

29.12 Camada de socket seguro (SSL e TLS)

Em meados da década de l990, quando se tornou evidente que a segurança era importante para o comércio na Internet, vários grupos propuseram mecanismos de segurança para uso com a Web. Embora não adotada formalmente pelo IETF, uma das propostas tornou-se de fato um padrão.

Conhecida como *Secure Sockets Layer* (*SSL*), a tecnologia foi desenvolvida originalmente pela Netscape, Inc. Como o nome indica, o SSL reside na mesma camada da API socket. Quando um cliente usa SSL para contatar um servidor, o protocolo SSL permite que cada lado se autentique com o outro. Os dois lados, então, negociam para selecionar um algoritmo de criptografia que ambos admitam. Finalmente, o SSL permite que os dois lados estabeleçam uma conexão criptografada (ou seja, uma conexão que usa o algoritmo de criptografia escolhido para garantir a privacidade). O IETF usou o SSL como base para um protocolo conhecido como *Transport Layer Security* (*TLS*). O SSL e o TLS estão tão relacionados que ambos utilizam a mesma porta bem conhecida, e a maioria das implementações do SSL admite TLS.

29.13 Firewalls e acesso à Internet

Mecanismos que controlam o *acesso à internet* tratam do problema de filtragem de uma rede em particular, ou de uma organização, a partir da comunicação indesejada. Esses mecanismos podem ajudar a evitar que estranhos obtenham e alterem informações ou que interrompam a comunicação na intranet de uma organização. O controle de acesso bem-sucedido requer uma combinação cuidadosa de restrições na topologia da rede, estágios de informações intermediárias e filtros de pacotes.

Uma técnica simples, conhecida como firewall de rede,* surgiu como base para o controle de acesso à internet. Uma organização coloca um firewall em sua conexão para a Internet global (ou para qualquer site externo não confiável). Um firewall particiona uma rede em duas regiões, conhecidas informalmente como interna e externa.

29.14 Conexões múltiplas e links mais fracos

Embora o conceito pareça simples, os detalhes complicam a construção do firewall. Primeiro, a intranet de uma organização pode ter várias conexões externas. A organização precisa formar um *perímetro de segurança* instalando um firewall em cada conexão externa. Para garantir que o perímetro seja eficaz, todos os firewalls precisam ser configurados de modo que usem exatamente as mesmas restrições de acesso. Caso contrário, pode ser possível contornar as restrições impostas por um firewall entrando na internet da organização por meio de outra.**

Podemos resumir conforme descrito a seguir.

> *Uma organização que possui várias conexões externas precisa instalar um firewall em cada uma, bem como coordenar todos os firewalls. Deixar de restringir o acesso de forma idêntica em todos os firewalls pode deixar a organização vulnerável.*

29.15 Implementando firewall e filtros de pacotes

Como um firewall deve ser implementado? Em teoria, um firewall simplesmente bloqueia toda a comunicação não autorizada entre os computadores na organização e os computadores fora dela. Na prática, os detalhes dependem da tecnologia da rede, da capacidade da conexão, da carga de tráfego e das políticas da organização. Nenhuma solução isolada funciona para todas as organizações – os sistemas de firewall são projetados

para serem configuráveis. Informalmente chamado de *filtro de pacotes*, o mecanismo requer que o administrador especifique como o roteador deve descartar cada datagrama. Por exemplo, o administrador pode decidir filtrar (ou seja, bloquear) todos os datagramas que vêm de uma origem e permitir os de outra, ou então um administrador poderia decidir bloquear todos os datagramas destinados a algumas portas TCP e permitir datagramas destinados a outras.

Para operar em velocidades de rede, um filtro de pacotes precisa de hardware e software otimizado para a tarefa. Muitos roteadores comerciais incluem hardware para a filtragem de pacotes de alta velocidade. Quando um administrador configura as regras de firewall, o filtro opera na velocidade do cabo, descartando pacotes indesejáveis, sem atrasar pacotes válidos.

Como o TCP/IP não dita um padrão para filtros de pacotes, cada fornecedor de roteador é livre para escolher as capacidades de seu filtro de pacotes, além da interface que um administrador utiliza para configurar o filtro. Alguns sistemas de firewall oferecem uma interface gráfica. Por exemplo, o firewall pode executar um servidor web que exibe páginas da web com opções de configuração. Um administrador pode usar um navegador convencional para acessar o servidor e especificar uma configuração. Outros sistemas de firewall usam uma interface de linha de comando.

Se um roteador oferece a funcionalidade de um firewall, a interface geralmente permite ao administrador que configure regras de filtragem separadas para cada interface. É importante ter uma configuração separada para cada interface porque uma interface de um roteador pode conectar-se a uma rede externa (por exemplo, um ISP), enquanto outras interfaces se conectam a redes internas. Assim, as regras para rejeitar pacotes variam entre as interfaces.

29.16 As regras de firewall e as 5-tuplas

Muitas regras de firewall concentram-se nos cinco campos encontrados nos cabeçalhos de protocolo que são suficientes para identificar uma conexão TCP. Na indústria, o conjunto é conhecido como as *5-tuplas* (*5-tuple*). A Figura 29.6 lista os campos.

IPsrc	**Endereço IP de origem**
IPdst	**Endereço IP de destino**
Proto	**Tipo de protocolo de transporte (por exemplo, TCP ou UDP)**
srcPort	**Número da porta de origem para o protocolo de transporte**
dstPort	**Número da porta de destino para o protocolo de transporte**

Figura 29.6 Campos de cabeçalho que compõem as 5-tuplas. Os cinco campos são suficientes para identificar uma conexão TCP individual.

Como isso se refere a datagramas IP, as 5-tuplas não incluem um campo do tipo Camada 2. Nesse sentido, nós tacitamente assumimos que o frame da Camada 2 especifica o pacote para ser um datagrama IP.

Um filtro de pacotes de firewall geralmente permite a um administrador especificar combinações arbitrárias de campo nas 5-tuplas. A Figura 29.7 ilustra um exemplo de especificação de filtro que se refere ao campo das 5-tuplas.

Na figura, o administrador escolheu bloquear os datagramas que chegam destinados a FTP (TCP porta 21), TELNET (TCP porta 23), WHOIS (UDP porta 43), TFTP (UDP porta 69) e FINGER (TCP porta 79). Além disso, os blocos de filtro de datagramas de saída se originam a partir de qualquer máquina com um endereço que corresponda ao prefixo IPv4 128.5.0.0/ 16 e que sejam destinados a um servidor de e-mail remoto (TCP porta 25).

Interface de chegada	IPsrc	IPdst	Proto	srcPort	dstPort
2	*	*	TCP	*	21
2	*	*	TCP	*	23
1	128.5.0.0/16	*	TCP	*	25
2	*	*	UDP	*	43
2	*	*	UDP	*	69
2	*	*	TCP	*	79

Figura 29.7 Um roteador com duas interfaces e um exemplo de especificação do filtro de datagramas.

29.17 Segurança e especificação de filtro de pacotes

Embora a configuração de filtro de exemplo na Figura 29.7 especifique uma

pequena lista de serviços a serem bloqueados, essa técnica não funciona bem para um firewall efetivo. Existem três motivos apresentados a seguir.

1. O número de portas bem conhecidas é grande e cresce rapidamente. Assim, a listagem de cada serviço exige que um administrador atualize a lista continuamente. Infelizmente, um erro de omissão pode tornar o firewall vulnerável.
2. Grande parte do tráfego em uma internet não vai de/para uma porta bem conhecida. Além dos programadores, que podem escolher números de porta para suas aplicações cliente-servidor privativas, serviços como *Remote Procedure Call* (*RPC*) e aplicações de compartilhamento de arquivos atribuem portas dinamicamente.
3. A listagem de portas de serviços bem conhecidos deixa o firewall vulnerável ao *tunelamento*. O tunelamento pode contornar a segurança se um host ou roteador no interior concordar em aceitar datagramas encapsulados de fora, remover uma camada de encapsulamento e encaminhar o datagrama adiante, para o serviço que de outra forma seria restringido pelo firewall. O problema é significativo porque um malware (software malicioso) inadvertidamente instalado no computador de um usuário pode explorar os pontos fracos do firewall.

Como um firewall pode usar um filtro de pacotes de forma eficaz? A resposta está na reversão da configuração do filtro: em vez de especificar que os datagramas devem ser bloqueados, um firewall deve ser configurado para bloquear todos os datagramas, exceto aqueles que o administrador admite. Ou seja, o administrador precisa especificar os hosts e as portas de protocolo para os quais a comunicação externa foi aprovada. Por padrão, toda a comunicação é proibida. Antes de habilitar qualquer porta, o administrador deve examinar cuidadosamente a política de informação da organização e, então, determinar se ela permite a comunicação. Na prática, os filtros de pacotes em muitos produtos comerciais permitem ao administrador especificar um conjunto de datagramas que admitirá, em vez de um conjunto que bloqueará. Nós podemos resumir o assunto da maneira a seguir.

Para ser eficaz, um firewall que usa a filtragem de datagramas deve restringir o acesso a todas as origens IP, destinos IP, protocolos e portas de protocolo, exceto para aqueles computadores, redes e serviços que a

organização decide tornar disponíveis externamente. Na verdade, um filtro de pacotes que permite a um administrador especificar quais datagramas serão admitidos, em vez de quais datagramas serão bloqueados, pode facilitar a especificação dessas restrições.

29.18 A consequência do acesso restrito para clientes

Talvez pareça que nossa regra simplista de bloquear todos os datagramas que chegam para uma porta de protocolo não conhecido resolva muitos problemas de segurança, evitando que estranhos acessem servidores arbitrários na organização. Esse firewall possui uma consequência interessante: ele também impede que um computador qualquer dentro do firewall se torne um cliente que acessa um serviço fora do firewall. Para entender por que, lembre-se de que, embora cada servidor opere em uma porta bem conhecida, um cliente não faz isso. Quando um aplicativo cliente inicia a execução, ele requisita ao sistema operacional selecionar um número de porta de protocolo que não está entre as portas bem conhecidas nem atualmente em uso no computador do cliente. Ao tentar se comunicar com um servidor fora da organização, um cliente gerará um ou mais datagramas e os enviará ao servidor. Cada datagrama que sai tem a porta de protocolo do cliente como porta de origem e a porta de protocolo bem conhecida do servidor como a de destino. Assumindo que o servidor externo foi aprovado, o firewall não bloqueará os datagramas quando eles saírem. Ao gerar uma resposta, entretanto, o servidor reverte as portas de protocolo, o que significa que a porta do cliente se torna a porta de destino. Quando uma resposta chegar ao firewall, as próprias regras dele bloquearão o pacote, tendo em vista que nenhuma regra foi criada para aprovar a porta de destino. Assim, podemos destacar uma ideia importante a seguir.

Se o firewall de uma organização restringe os datagramas que chegam exceto para portas que correspondem a serviços que a organização torna disponíveis externamente, uma aplicação arbitrária qualquer dentro da organização não pode se tornar um cliente de um servidor fora dela.

29.19 Stateful Firewalls

Como quaisquer clientes dentro da organização podem ter permissão para acessar serviços na Internet sem admitir datagramas que chegam, destinados a quaisquer portas de protocolo? A resposta está em uma técnica conhecida como firewall com estado (stateful firewall). Essencialmente, o firewall observa as conexões de saída e adapta as regras de filtro de acordo, para acomodar respostas.

Como um exemplo de um stateful firewall, suponha que um cliente no interior de uma organização forme uma conexão TCP com um servidor Web. Se o cliente tiver o endereço IP de origem I_1 e a porta TCP de origem P_1 e conectar-se a um servidor Web na porta 80 com endereço IP I_2, o segmento SYN de saída que inicia a conexão passará pelo firewall, que registra as 5-tuplas:

$$(I_1, I_2, \text{TCP}, P_1, 80)$$

Quando o servidor retornar um SYN+ACK, o firewall combinará as duas extremidades com a tupla que foi armazenada, e o segmento que entra será admitido.

É interessante que um stateful firewall não permite aos clientes dentro da organização iniciarem conexões com quaisquer destinos. Em vez disso, todas as ações no stateful firewall são controladas por um conjunto de regras de filtro de pacote. Assim, um administrador de firewall ainda pode escolher se deve permitir ou negar o trânsito para determinado pacote. No caso de um pacote que tem permissão para passar pelo firewall, a regra do filtro pode especificar ainda mais se deve registrar informações de estado que permitirão que uma resposta seja retornada.

Como o estado deve ser gerenciado em um stateful firewall? Existem duas técnicas gerais: um firewall pode usar o estado flexível definindo um timer que remove a informação de estado inativo depois de um período de timeout, ou então pode haver o monitoramento de conexão, em que o firewall observa pacotes no fluxo e remove informações de estado quando o fluxo termina (por exemplo, quando um FIN é recebido em uma conexão TCP). Mesmo que um stateful firewall tente monitorar as conexões, o estado flexível normalmente é um backup para tratar de casos como um fluxo UDP que não possui término explícito.

29.20 Proteção de conteúdo e proxies

Os mecanismos de segurança descritos anteriormente têm como foco o acesso. Outro aspecto de segurança, no entanto, focaliza o conteúdo.

Sabemos, por exemplo, que um arquivo importado ou uma mensagem de e-mail pode conter vírus. Em geral, esses problemas só serão eliminados por um sistema que examine o conteúdo que chega.

Uma abordagem, conhecida como *inspeção profunda de pacotes* (*Deep Packet Inspection* – DPI), analisa a carga útil dos pacotes de entrada. Ainda que possa capturar alguns problemas, a DPI nem sempre consegue lidar com situações em que o conteúdo é dividido em vários pacotes e estes chegam fora de ordem. Assim, criaram-se alternativas para extrair o conteúdo de uma ligação e, em seguida, examinar o resultado antes de permitir que ele passe para a organização.

Um mecanismo que examina o conteúdo precisa atuar como um proxy de aplicação. Por exemplo, uma organização pode executar um proxy HTTP que intercepta cada organização de saída, obtém uma cópia do item solicitado e a verifica. Se a cópia for considerada livre de vírus, o proxy a encaminha para o cliente. Por outro lado, se a cópia contém vírus, ao cliente é enviada a mensagem de erro. Observe que um proxy de aproximação pode ser transparente (por exemplo, com exceção de um atraso, o cliente não percebe que o proxy interceptou um pedido) ou não transparente (por exemplo, o cliente precisa ser configurado para usar um proxy específico).

Embora muitas organizações utilizem um stateful firewall para se protegerem contra sondas aleatórias de fora, poucas verificam o conteúdo. Assim, é típico encontrar uma organização que seja imune a ataques de fora, mas que seja que assolada por problemas de vírus de e-mail e cavalo de Troia quando um funcionário ingênuo importa um programa que pode penetrar em um firewall, formando uma conexão com o exterior.

29.21 Monitoração e logging

A monitoração é um dos aspectos mais importantes de um projeto de firewall. O administrador de rede responsável por um firewall precisa estar ciente das tentativas de burlar a segurança. Uma lista de tentativas fracassadas de invasão pode fornecer vários indícios sobre o ataque, como um endereço IP ou mesmo um padrão de como tais tentativas são feitas. Desse modo, um firewall deve relatar incidentes.

A monitoração pode ser *ativa* ou *passiva*. Na monitoração ativa, um firewall notifica o administrador sempre que houver um incidente. A principal vantagem desse tipo de monitoração é a velocidade – administrador descobre um problema em potencial imediatamente. A

principal desvantagem é que os monitores ativos normalmente produzem tanta informação que um administrador pode não compreendê-la ou não notar problemas. Assim, muitos administradores preferem a monitoração passiva, ou uma combinação de monitoração passiva com alguns incidentes de alto risco, também relatados por um monitor ativo.

Na monitoração passiva, um firewall faz um registro de cada incidente em um arquivo no disco. Um monitor passivo normalmente registra informações sobre o tráfego normal (por exemplo, estatísticas simples), além de datagramas que são filtrados. O administrador pode acessar o log a qualquer momento; a maioria dos administradores utiliza um programa de computador para isso. A principal vantagem da monitoração passiva vem do seu registro de eventos – o administrador pode consultar o log para observar tendências e, quando ocorrer um problema de segurança, rever o histórico de eventos que levaram a isso. Mais importante do que isso, um administrador pode analisar o log periodicamente (por exemplo, todos os dias) para determinar se as tentativas de acessar a organização aumentam ou diminuem com o tempo.

29.22 Resumo

Problemas de segurança surgem porque uma rede pode conectar organizações que não possuem confiança mútua. Várias tecnologias estão disponíveis para ajudar a garantir que as informações permaneçam seguras quando estão sendo enviadas por uma internet. O protocolo Secure Sockets Layer (SSL) acrescenta criptografia e autenticação à API socket. O IPsec permite a um usuário escolher entre dois esquemas básicos: um que oferece autenticação do datagrama e um que fornece autenticação mais privacidade. O IPsec modifica um datagrama inserindo um Authentication Header ou usando um En-capsulating Security Payload, que insere um cabeçalho e um trailer que criptografa os dados que estão sendo enviados. Além disso, oferece uma estrutura geral que permite que cada par de entidades em comunicação escolha um algoritmo de criptografia. Como a segurança normalmente é usada com o tunelamento (por exemplo, em uma VPN), o IPsec define um modo de túnel seguro.

O mecanismo de firewall é usado para controlar o acesso à internet. Uma organização coloca um firewall em cada conexão externa para garantir que a intranet que usa permaneça livre de tráfego não autorizado. Um firewall contém um filtro de pacotes que um administrador precisa configurar para especificar quais pacotes podem passar em cada direção. Um stateful firewall pode ser configurado para que o firewall permita pacotes de resposta automaticamente quando uma conexão com o exterior tiver sido

estabelecida.

EXERCÍCIOS

29.1 Leia a descrição de um filtro de pacotes para um roteador comercialmente disponível. Que recursos ele oferece?

29.2 Colete um log de todo o tráfego que entra no seu site. Analise o log para determinar a porcentagem de tráfego que chega ou é destinado a uma porta de protocolo bem conhecida. Os resultados o surpreendem?

29.3 Se o software de criptografia estiver disponível no seu computador, meça o tempo exigido para criptografar um arquivo de l0 Gbyte, transferi-lo para outro computador e decriptografá-lo. Compare o resultado com o tempo exigido para a transferência se nenhuma criptografia for usada.

29.4 Procure determinar com os usuários no seu local se eles enviam informações confidenciais no e-mail. Os usuários estão cientes de que as mensagens são transferidas em textos simples e, portanto, qualquer um observando o tráfego da rede pode ver o conteúdo delas?

29.5 Pode um firewall ser combinado com NAT? Quais são as consequências?

29.6 Os militares só liberam informações para aqueles que "precisam saber". Esse esquema funcionará para todas as informações na sua organização? Sim ou não? Por quê?

29.7 Cite dois motivos para que o grupo de pessoas que administra as políticas de segurança de uma organização deva estar separado do grupo de pessoas que administra os sistemas de computador e de rede da organização.

29.8 Algumas organizações utilizam firewalls para isolar grupos de usuários internamente. Cite exemplos de maneiras como os firewalls internos podem melhorar o desempenho da rede e exemplos de maneiras como os firewalls internos podem degradar o desempenho da rede.

29.9 Se a sua organização utiliza IPsec, descubra quais algoritmos estão sendo usados. Qual é o tamanho da chave?

29.10 O IPsec pode ser usado com NAT? Explique.

---

* O termo firewall é derivado da arquitetura de prédios em que um firewall é uma

partição espessa e à prova de fogo, tornando uma parte da construção impenetrável ao fogo.

** A ideia bem conhecida de que a segurança é tão forte quanto o ponto mais fraco foi denominada axioma do elo mais fraco, em referência ao fato de que o adágio de uma cadeia é apenas tão forte quanto seu elo mais fraco.

CAPÍTULO 30

Sistemas embarcados conectados (A Internet das coisas)

CONTEÚDOS DO CAPÍTULO

30.1 Introdução

Os primeiros capítulos descrevem os princípios, protocolos e arquitetura por trás do TCP/IP e da Internet global. Os últimos capítulos se concentram em protocolos de aplicação e paradigmas que os aplicativos usam para se comunicar através de uma internet. A maioria das aplicações da Internet, que incluem e-mail, navegação na web, download de música, programas de chat e até mesmo de gerenciamento de rede, tem uma coisa em comum: um ser humano inicia a comunicação. De fato, o início da Internet tem sido descrito como concentrado em fornecer mecanismos de comunicação que os seres humanos utilizam.

Este capítulo se concentra em uma nova e importante área que está emergindo como parte da Internet: a comunicação entre sistemas inteligentes embarcados. A ideia é que os sistemas de computação podem ser incorporados em muitos dispositivos e, portanto, serão capazes de se comunicar com outros dispositivos. Claro, alguns dos dispositivos embarcados fornecerão dados para os seres humanos verem e usarem, e, desse modo, a interação humana está incluída. No entanto, a ênfase principal será nos sistemas que podem interagir com o seu ambiente, em vez de computadores convencionais que armazenam dados e executam aplicativos. Pesquisadores e profissionais que trabalham com o tema cunharam os termos *Internet das coisas* (*Internet of Things – IoT*)* e *aplicativos Machine to Machine* (*M2M*) para captar a ideia.

O capítulo começa apresentando exemplos de sistemas inteligentes embarcados e a forma como eles usam a comunicação. Em seguida, discute uma tecnologia particular em detalhes: uma pilha de protocolos para uma rede sem fio que se destina a aplicações de smart grid. O uso de IPv6 e algumas das consequências do projeto IPv6 são de particular relevância para a discussão.

30.2 Sensoreamento, monitoramento e controle

Usamos o termo *sistema embarcado* como referência a um sistema computacional que é uma parte integrante de outro mecanismo ou dispositivo. A principal diferença entre um sistema embutido e um sistema de computador convencional decorre de suas conexões externas. Um sistema de computador convencional lida com informação: pode armazenar, acessar e manipular dados. Em contraste, um sistema embarcado pode *sentir* e *controlar* o mundo físico em torno dele.

Como exemplo, considere um termostato utilizado para controlar um sistema de aquecimento e de ar condicionado. Um termostato moderno (chamado de "termostato inteligente") constitui um sistema embarcado: o termostato contém um processador embutido que usa um software para executar todas as suas funções. Um usuário pode configurar o termostato para usar a hora do dia a fim de alterar as configurações. O termostato tem conexões com uma variedade de sensores, que podem incluir: um sensor de temperatura interna, um sensor de temperatura externa, um sensor que detecta o fluxo de ar (isto é, se o ventilador está funcionando) e um sensor ligado aos comandos que permitem ao usuário ajustar a temperatura desejada. Sistemas mais avançados têm sensores para a umidade relativa do ar. Além disso, o termostato apresenta conexões com controles que permitem ao processador ligar ou desligar o aquecedor ou o ar-condicionado, regular a velocidade do ventilador e o controle das funções de umidificar e desumidificar.

A maioria dos usuários de computador já está familiarizada com sistemas embarcados. Por exemplo, uma impressora conectada a um computador incorpora um sistema embarcado. Quando um computador envia um documento para a impressora, o sistema incorporado da impressora controla os motores e os mecanismos existentes nela, levando-os a alimentar folhas de papel, mover o mecanismo de jato de tinta e aspergir gotas de tinta. A impressora também contém sensores que podem detectar um congestionamento de papel ou baixos suprimentos de tinta.

A General Electric, a maior empresa industrial nos Estados Unidos, vai

muito mais além dos que itens para o mercado consumidor. A G.E. produz motores de aeronaves, turbinas para usinas de energia, locomotivas para ferrovias, equipamentos médicos de imagens e maquinaria pesada para transporte de pessoas, além de aquecer casas e fornecer energia a instalações fabris. Usando a expressão Internet industrial, a G.E. lançou uma importante iniciativa para incorporar sistemas embarcados de comunicação tanto em suas fábricas como em seus produtos.

30.3 Conservação e captação de energia

Alguns sistemas embarcados, tais como o sistema de controle embutido em uma impressora, se conectam a uma fonte confiável de energia contínua. No entanto, muitos desses sistemas dependem de energia temporária, e são projetados para conservar energia. Por exemplo, os celulares funcionam com baterias e sensores ambientais localizadas em locais remotos (por exemplo, um deserto) podem utilizar fotocélulas.

Como um caso especial, alguns sistemas embarcados são concebidos para captarem energia a partir do ambiente que os rodeia. Por exemplo, um sensor no oceano pode usar o movimento das ondas para gerar energia, e um sensor perto de uma fonte de água quente pode utilizar a energia térmica. A captação de energia ainda inclui a energia cinética que os seres humanos geram apenas abrindo uma porta ou acionando um interruptor de luz. Um sistema integrado que utiliza a energia captada pode precisar operar periodicamente – o sistema pode precisar acumular energia até que uma carga suficiente esteja disponível (por exemplo, para operar um transmissor de rádio).

30.4 Um mundo de dispositivos inteligentes incorporados

Para entender a Internet das coisas, temos de imaginar que sistemas embarcados estarão em toda parte: casas, prédios de escritórios, veículos, shoppings e nas esquinas das ruas. Considere veículos como um exemplo. Além de sistemas embarcados que fornecem informações, entretenimento e navegação, cada carro terá um sistema que verificará a distância para os carros ao redor, perceberá objetos na estrada e alertará para mudanças no pavimento (por exemplo, um pequeno desnível ou calombo devido à construção). Sistemas embarcados serão capazes de sentir as pistas e alertar o motorista caso o carro desvie para fora da sua faixa ou quando houver congestionamento de tráfego à frente. Se o sistema for inteligente, poderá monitorar outros carros e aplicar os freios, se necessário. Além disso, um sistema inteligente pode identificar o motorista (por exemplo, por meio do

reconhecimento facial), monitorar hábitos de condução do indivíduo e usá-los para ajustar os tempos de reação.

Em edifícios de escritórios, sistemas embarcados podem sentir a presença de pessoas e ajustar luzes, aquecimento ou arrefecimento. Também podem ser alterados os sistemas de aquecimento e refrigeração quando as janelas estiverem. Mais importante, através da adição de inteligência para sistemas embarcados, o sistema poderá ser capaz de aprender padrões. Por exemplo, se um funcionário entrar e sair do seu escritório com frequência durante o dia de trabalho, o sistema poderá aprender a não desligar o aquecimento até que ele esteja ausente por um tempo mais longo. Da mesma forma, se um funcionário tender a trabalhar até mais tarde, o sistema vai aprender o padrão e controlar o escritório de acordo com isso. Assim, se um empregado com horários flexíveis for para casa às três horas a cada dia, um sistema inteligente poderá aprender o padrão e desligar a iluminação e o aquecimento mais cedo.

30.5 A importância da comunicação

Por que a ênfase está se deslocando para sistemas embarcados inteligentes que podem se comunicar? Há muitas vantagens: a comunicação permite controle, coordenação e transferência de dados. Pequenos sistemas embarcados podem conseguir muito mais trabalhando em conjunto com os sistemas embarcados nas proximidades ou acessando informações remotas.

Como exemplo, considere um conjunto de sensores utilizados para avaliar a tensão em pontes. Medir a tensão é importante para entender quanto tráfego uma ponte pode transportar com segurança. Medições de tensão podem indicar quando um reforço é suficiente ou se a ponte deve ser substituída. Para medir a tensão, os engenheiros colocam pequenos sensores movidos a bateria em vários pontos ao longo da ponte. Sem comunicação, cada sensor deve armazenar medições, juntamente com uma estampa de tempo. Periodicamente, os engenheiros coletam os dados armazenados e resetam o sensor para coletar mais. Se cada sensor incluir um rádio, o conjunto de sensores poderá formar uma rede sem fio. A comunicação em rede permite que os sensores se coordenem para que tomem cada leitura simultaneamente. Também permite que enviem os resultados para um ponto de coleta comum, possivelmente um local longe da ponte. Embora coletar leituras eletronicamente seja muito mais conveniente do que um método manual, o aspecto mais importante de comunicação é a nova funcionalidade que permite. No caso de sensores, fazer o upload dos dados em tempo real faz com que seja possível detectar situações perigosas e desviar o tráfego antes que ocorra um desastre.

Como um segundo exemplo de comunicação, considere *medidores inteligentes* usados por empresas prestadoras de serviços, como água e energia elétrica. A abordagem tradicional utilizada para avaliar o uso consiste na colocação de um medidor na parte de fora da casa de cada cliente e no envio de uma pessoa para registrar a leitura de consumo a cada mês. Um medidor inteligente incorpora comunicação sem fio, o que significa que a empresa concessionária pode ler o medidor de um local remoto. Mesmo que a rede sem fio só chegue até a rua, o medidor pode ser lido a partir de um veículo passando pela área, o que reduz consideravelmente o custo de leitores de consumo.

30.6 Exemplo: anúncios eletrônicos em shopping centers

Muitos shoppings têm agora grandes painéis de exibição de vídeo que mostram anúncios. E estes mudam continuamente – lojas no shopping pagam uma taxa para mostrar imagens de seus produtos, fixas ou em vídeos cheios de movimento. Eles também anunciam descontos e apresentam promoções especiais de curto prazo (por exemplo, para os próximos 30 minutos, se você comprar o produto Y na loja X e mencionar este anúncio, você receberá um segundo grátis). A comunicação é necessária para baixar os anúncios de forma dinâmica, porque o conteúdo e os horários podem mudar a qualquer momento – o gerenciamento precisa da capacidade de controlar qual anúncio é exibido em uma determinada tela e quanto tempo o anúncio permanece visível.

Onde está localizado o controlador que a administração usa e qual a tecnologia de rede necessária para conectá-lo aos painéis individuais? A resposta é interessante e um pouco surpreendente: cada tela tem uma pilha de protocolos TCP/IP e uma conexão com a Internet global. Há duas razões. Em primeiro lugar, uma conexão com a Internet permite que o controlador possa estar em qualquer lugar. Em particular, o shopping pode terceirizar funções de TI, colocando o controlador "na nuvem". Mais importante, ter uma conexão com a Internet significa que o conteúdo não precisa ficar no mesmo host físico do controlador.

Permitir que o conteúdo esteja separado do controlador é importante porque muitas lojas em um shopping são parte de uma rede nacional (ou internacional). Por exemplo, a Apple Computer pode ter lojas em vários shoppings. Filiais não produzem vídeos promocionais. Em vez disso, a matriz cria toda a publicidade e a torna disponível para as filiais. O conteúdo de vídeo para anúncios reside no servidor da matriz. Ter acesso à Internet significa que o controlador só precisa especificar um programa de

itens a serem exibidos, juntamente com a URL de cada item. Como cada um tem acesso à Internet, os sistemas de visualização podem baixar uma cópia de cada item que lhes for atribuído para mostrar (possivelmente através de um cache local para melhorar o desempenho).

30.7 Coleta de dados de sistemas embarcados

O exemplo dos shopping centers ilustra outro recurso importante dos sistemas em rede: a capacidade de coletar dados. Embora não seja óbvio para os clientes, alguns dos monitores em shopping centers são equipados com uma câmera, e o software é programado para operar a câmera. Quando alguém se aproxima da tela, o sistema utiliza a câmera para detectar sua presença e aplicar algoritmos de reconhecimento facial. Uma vez que as faces tenham sido identificadas, o sistema utiliza as características faciais (por exemplo, a distância entre os olhos) para analisar o indivíduo. Com alta precisão, o software no sistema integrado é capaz de dizer se o indivíduo na frente da câmera é homem ou mulher, bem como sua idade aproximada. Em vez de simplesmente seguir um cronograma predeterminado de anúncios para mostrar, o sistema pode utilizar as características do indivíduo para escolher um anúncio apropriado. Assim, a um homem de meia-idade pode ser mostrado um anúncio de um carro esportivo no lugar do um anúncio de um vestido.

Embora a câmera possa ser usada para selecionar os anúncios, o sistema também pode usá-la para coletar e relatar dados sobre a interação. Por exemplo, o sistema pode enviar estatísticas sobre quantas pessoas assistiram a um determinado anúncio, seu sexo e idade, e por quanto tempo uma pessoa ou grupo permaneceu na frente da tela. Supermercados começam a usar a mesma abordagem: eles estão instalando câmeras com capacidade de processamento em freezers e em outros locais da loja. As câmeras registram se os clientes param para olhar um determinado produto em exposição, quanto tempo cada pessoa fica parada olhando e quantos clientes, finalmente, escolhem o produto ou apenas seguem em frente. Em cada caso, a informação é enviada a um servidor para análise.*

Podemos fazer o resumo a seguir.

Adicionar capacidade de comunicação a um sistema inteligente integrado permite que a informação flua em duas direções: dados podem ser baixados para controlar as ações do sistema e o sistema pode fazer upload de dados coletados a partir de sensores.

30.8 Rede sem fio e IEEE 802.15.4

Muitos dos nossos exemplos usam a tecnologia de rede sem fio. As redes sem fio funcionam extremamente bem para sensores ambientais que estão implantados ao longo de uma vasta área (por exemplo, sensores em uma ponte ou sensores de terremoto implementados por quilômetros em torno de uma falha). No entanto, a rede sem fio também é usada para implementações semipermanentes em uma área pequena. Por exemplo, o uso de redes sem fio nos displays eletrônicos em shoppings significa que uma unidade de exibição pode ser movida sem a instalação de uma nova conexão de rede. As unidades são grandes e pesadas, o que significa que a instalação é semipermanente (por exemplo, o visor permanece estacionário durante semanas). O uso de wireless torna a relocação muito mais fácil do que em uma rede cabeada.

Que tecnologias de rede sem fio devem ser usadas para conectar um sistema inteligente integrado? A resposta depende de vários fatores, incluindo a distância geográfica entre os nós da rede, as taxas de dados desejados e a energia a ser usada. A energia é especialmente importante por duas razões. Em primeiro lugar, muitos sistemas de sensores embutidos funcionam com energia de bateria. Para maximizar a vida útil da bateria, o consumo de energia deve ser minimizado (notamos que, entre outras coisas, reduzir a utilização global de energia pode significar a redução da quantidade de energia usada pelo transmissor de rádio, e pode significar a redução da memória ou da velocidade do processador). Em segundo lugar, mesmo que o sistema integrado tenha uma fonte de energia, pode ser necessário limitar a intensidade do sinal para evitar a interferência com outros dispositivos ou outras transmissões.

Várias tecnologias de rede sem fio de baixa potência foram padronizadas, incluindo a tecnologia bluetooth, que é bem conhecida. Para o nosso exemplo, vamos usar uma tecnologia de rede de área pessoal de baixa potência definida pelo padrão IEEE 802.15.4. O padrão IEEE especifica as camadas física e MAC da rede, e outros grupos definiram protocolos de camada superior para uso na rede. Várias versões do 802.15.4 foram produzidas; elas diferem nas faixas de frequências e técnicas de modulação usadas. Para os nossos propósitos, é necessário apenas compreender que a taxa de dados é relativamente baixa (um máximo de 250 Kbps), o MTU é extremamente pequeno (127 octetos) e a distância é limitada (um máximo de 10 metros com uma antena convencional e energia por bateria). As próximas seções consideram uma aplicação de tecnologia sem fio 802.15.4 e camadas superiores de uma pilha de protocolos.

30.9 Uma rede de malha para Smart Grid Sensors

Uma aplicação de tecnologia sem fio 802.15.4 surge de um esforço para adicionar gerenciamento de computador inteligente para o sistema de distribuição de energia elétrica. Conhecido como *Smart Grid*, o plano global inclui a colocação de sensores em todos os dispositivos que usam eletricidade. Além de sistemas de grande porte, tais como aqueles usados para aquecimento e arrefecimento, os desenvolvedores imaginam sensores em aparelhos de cozinha (por exemplo, fornos, geladeiras, máquinas de lavar louça, e até mesmo torradeiras), sistemas de computadores e sistemas de entretenimento (por exemplo, televisores e aparelhos de som). As empresas de serviços querem cobrar mais nos horários de pico, e os sensores irão se comunicar com a empresa concessionária para obter informações sobre preços e alertar os usuários quando estes forem elevados. Como alternativa, os sensores serão capazes de desabilitar certos usos durante o horário de pico.

O projeto mais óbvio para um sistema de sensores consiste em colocar uma estação-base em cada residência e usar uma tecnologia sem fio que permite que essa estação se comunique com cada sensor. A abordagem é conhecida como uma *topologia hub-and-spoke*. Como vimos, tecnologias (Wi-Fi) 802.11 usam uma abordagem hub-and-spoke. No entanto, tal concepção não funciona bem para todas as situações. Em particular, tubos metálicos e outros obstáculos que interferem com os sinais sem fio podem fazer com que seja impossível chegar a todos os locais a partir de um único ponto, especialmente para aparelhos portáteis que podem ser movidos de uma sala para outra. Por isso, os desenvolvedores imaginam um sistema adaptativo, em que o conjunto de nós sensores constitui automaticamente uma *rede de malha auto-organizada* (geralmente abreviado como *malha*). Cada nó na malha desempenha duas tarefas: comunicação para dispositivos a que se conectam e encaminhamento para outros nós.

Cada residência irá conter um *roteador de borda* que conecta os sensores na residência com o mundo exterior. Quando um nó é inicializado, ele se junta à malha e tenta estabelecer uma conexão com o roteador de borda. Se pode alcançar o roteador de borda, o nó se comunica diretamente com ele. Se não pode alcançá-lo diretamente, o nó procura por um nó vizinho próximo que tenha um caminho para o roteador de borda. Em essência, o vizinho se compromete a agir como um roteador que encaminha pacotes.

Se vários vizinhos têm um caminho para o roteador de borda, um nó

aplica um algoritmo de seleção para escolher um. O algoritmo de seleção pode levar em conta várias métricas, incluindo a qualidade do sinal de rádio (ou seja, o nível de interferência), a latência, capacidade do link e capacidade de nós de encaminhamento intermediários. Uma vez que um nó tenha um caminho para o roteador de borda, ele informa aos vizinhos e se compromete a encaminhar os seus pacotes. Assim, se for possível criar uma rede que proporcione conectividade a cada nó, os nós formarão uma rede de malha automaticamente. Podemos fazer o resumo a seguir.

Em vez de usar uma poderosa estação de base única que possa alcançar o sensor em cada dispositivo elétrico, está sendo concebido um sistema em que um conjunto de nós sensores se auto-organizam para formar uma rede de malha; cada nó na malha concorda em enviar pacotes para outros nós.

30.10 Uma árvore de encaminhamento para uma malha de sensores

Nossa descrição faz a seleção de caminhos em uma malha parecer trivial. De fato, uma malha oferece muitos caminhos possíveis. Além disso, a distinção entre roteamento (escolhendo caminhos entre as redes) e encaminhamento (escolhendo caminhos dentro de uma rede) é turva porque podemos pensar em cada link de rádio como uma rede. Mais importante, a conectividade é entre pares de nós. A agregação de roteamento que ocorre a partir de um prefixo IP não se aplica porque a malha não funciona como uma única rede na qual todos os nós se conectam. Se nós da malha são móveis, o encaminhamento deve mudar dinamicamente. Apesar da complexidade potencial do roteamento de malha, precisamos encontrar uma solução que funcione para os casos básicos. Portanto, vamos assumir uma topologia de malha estática e considerar apenas o encaminhamento entre o roteador de borda e nós individuais.

Se cada nó sensor escolhe um caminho para alcançar o roteador de borda, os caminhos de encaminhamento na rede formam uma *árvore* (ou seja, um grafo sem ciclos). A Figura 30.1 ilustra um conjunto de nós e uma árvore de encaminhamento possível. Como a figura indica, um roteador de borda é geralmente maior do que os outros nós da rede (ou seja, tem mais poder de processamento e memória). Veremos que é esperado que um

roteador de borda possa executar servidores que prestam serviços para a malha como um todo.

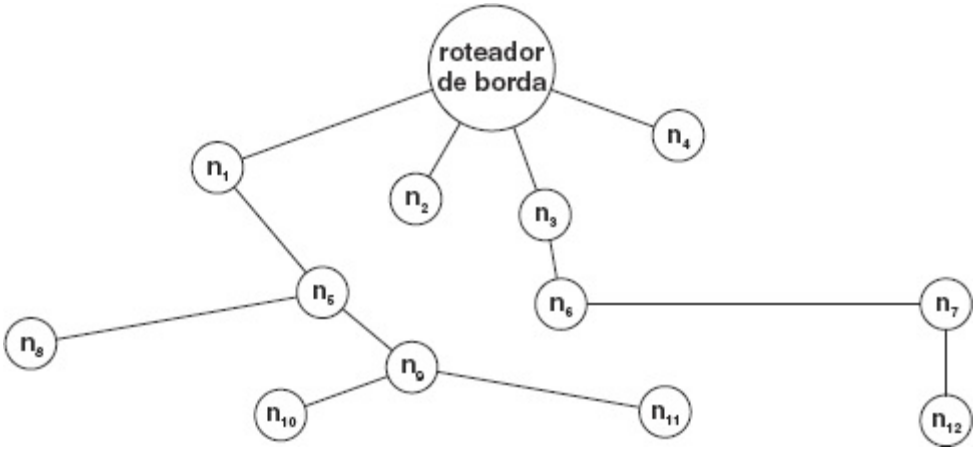


Figura 30.1 Um exemplo de árvore encaminhamento aplicada a um conjunto de nós sensores. Se cada nó escolher um caminho para o roteador de borda isso resultará em uma árvore.

30.11 Usando protocolos de camadas 2 e 3 em uma malha

Lembre-se do Capítulo 4, em que duas abordagens foram concebidas para criar uma árvore de encaminhamento através de uma malha, a saber:

- mesh-under: encaminhamento usa protocolos da Camada 2;
- route-over: encaminhamento usa protocolos da Camada 2.

Em qualquer abordagem, quando um nó entra na malha, deve escolher a forma de se conectar na árvore. Existem duas etapas. Na primeira, um nó deve encontrar o conjunto de nós vizinhos que pode ser alcançado diretamente e avaliar a qualidade do sinal de rádio para cada um. Na segunda etapa, o nó deve optar por se comunicar com o roteador de borda diretamente ou usar um dos vizinhos para encaminhar pacotes. Note que a qualidade de um sinal é de fundamental importância – mesmo que um nó possa alcançar o roteador de borda diretamente, ele pode escolher um caminho indireto, se a qualidade do sinal da conexão direta for muito baixa.

Mesh-under. Com a abordagem *mesh-under*, que é favorecida pelo IEEE, um nó utiliza protocolos de Camada 2 para formar uma árvore de encaminhamento. A ideia é semelhante à da ponte e dos protocolos árvore estendidos utilizados para Ethernet. Por exemplo, um nó usa broadcast de Camada 2 para descobrir quais vizinhos estão dentro do alcance de rádio. Cada vizinho responde, o que permite que o par saiba que podem alcançar

um ao outro e a qualidade do sinal. O roteador de borda também usa broadcast de Camada 2 para anunciar a si mesmo. Nós na malha repassam informações sobre quais nós eles podem alcançar. Eventualmente, cada nó na malha está ciente de outros nós e de como alcançá-los, bem como sobre a forma de encaminhar pacotes por broadcast. Assim, dado o endereço da Camada 2 do roteador de borda, nós na malha saberão como encaminhar pacotes (ou seja, eles terão formado uma árvore de encaminhamento).

Route-over. Com a abordagem de *route-over*, que é favorecida pela IETF, um nó utiliza protocolos de Camada 3 para identificar vizinhos e formar uma árvore de encaminhamento. É claro que o hardware subjacente não entende endereços IP ou o formato do datagrama IP. Assim, comunicação de Camada 3 usa pacotes da Camada 2. Por exemplo, considere encontrar vizinhos. O software gera um datagrama de Camada 3 com um endereço de broadcast local e esse o datagrama é enviado via broadcast de hardware.

Em termos de uma pilha de protocolos IP, a principal diferença entre mesh-under e route-over surge no encaminhamento IP. A abordagem mesh-under trata todo o conjunto de nós como uma rede. O IP atribui um prefixo para a rede e o encaminhamento IP faz com que qualquer datagrama destinado a um nó na rede passe para a interface. A Camada 2 aceita o datagrama do IP e utiliza a informação que foi reunida para encaminhar o datagrama.

A abordagem route-over torna o IP ciente de que alguns nós da rede são alcançáveis diretamente e outros não. Na terminologia do IPv6, dizemos que um nó está *on link* ou *off link*. Para lidar com os nós que não são diretamente alcançáveis, o IP usa o *roteamento de origem*. Ou seja, o IP deve compreender a topologia da malha e ser capaz de especificar um caminho (por exemplo, ir ao nó 9, então ao nó 5, em seguida, ao nó 1, e finalmente para o roteador de borda). As próximas seções descrevem a pilha do protocolo ZigBee que usa a abordagem route-over e avalia alguns dos problemas que surgem.

30.12 A pilha de protocolos IPv6 ZigBee

A ZigBee Alliance e o IETF estão explorando o uso de IPv6 em um projeto de malha que segue a abordagem route-over. A Figura 30.2 lista três grupos de trabalho IETF relacionados ao esforço ZigBee.

Nome	Propósito

6lowpan	**Camada shim IP-over-802.15.4**
roll	**Protocolo de roteamento RPL para redes de malha**
coap	**Protocolo RESTful Lightweight para dispositivos restringidos**

Figura 30.2 Grupos de trabalho IETF relacionados ao esforço ZigBee.

A ideia básica do projeto route-over ZigBee é utilizar o IPv6 quando possível e introduzir as modificações necessárias. Um protocolo foi criado para comprimir datagramas IPv6 e enviá-los através de um link de rádio 802.15.4. Uma forma modificada de Descoberta de Vizinho IPv6 é usada para encontrar os endereços IP dos vizinhos diretamente alcançáveis, e um protocolo foi inventado para permitir que os vizinhos troquem características. Além disso, um novo protocolo de roteamento é usado para reunir informações sobre conectividade em toda a malha. Finalmente, um cabeçalho de rota de origem IPv6 é usado para encaminhar cada datagrama salto a salto através da malha. As próximas seções descrevem alguns dos protocolos básicos.

30.12.1 IPv6 em redes sem fio de baixa potência (6LoWPAN)

O esforço 6LoWPAN (IPv6 em redes de área pessoais sem fio de baixa potência – *IPv6 over Low-power Wireless Personal Area Networks*) define a transmissão de IPv6 através de um link de rádio 802.15.4. O principal problema é um conflito entre a exigência de IPv6 para um MTU de pelo menos 1280 octetos e o protocolo 802.15.4 que especifica um máximo de 127 octetos. De fato, se a encriptação AES-CCM-128 é usada, o tamanho da carga disponível se reduz a 81 octetos. Para enviar um datagrama IPv6 nesse link, o 6LoWPAN introduz uma camada shim extra que realiza a compressão e a transmissão. A camada shim aceita um datagrama IPv6 de saída, comprime o cabeçalho, divide o datagrama em uma série de pedaços, que chamaremos *fraglets*, e envia cada fraglet em um pacote separado. No lado do receptor, a camada shim *6LoWPAN* aceita os fraglets recebidos, os recombina em um único datagrama, descomprime o cabeçalho e passa o resultado para a camada IP. Assim, o IPv6 é configurado para enviar e receber datagramas completos.

Note-se que o 6LoWPAN não usa fragmentação IPv6 convencional. Há duas razões para isso. Primeiro, o 6LoWPAN só precisa operar em um único

link. Assim, o protocolo é muito mais simples, porque todos os fraglets de um datagrama devem chegar em ordem. Em segundo lugar, a fragmentação do IPv6 não pode lidar com um MTU de 127 octetos.

30.12.2 Descoberta de vizinho (6LoWPAN)

Como vimos, a descoberta de vizinho IPv6 (IPv6-ND) inclui detecção de endereços duplicados. A premissa fundamental é que cada prefixo IPv6 mapeia para um domínio de broadcast; um nó pode usar uma transmissão broadcast para chegar a todos os outros nós que compartilham o prefixo. Em uma rede de malha, no entanto, um broadcast só pode chegar a alguns dos nós na malha. Dizemos que alguns nós estão *off link*. Como resultado, a detecção de endereços duplicados convencional, que assume que todos os nós são alcançáveis através de broadcast ou multicast, não vai funcionar corretamente.

O 6LoWPAN-ND define várias mudanças e otimizações de IPv6-ND que são destinados especificamente para redes sem fio de baixa potência com perdas. Em geral, o 6LoWPAN-ND evita todos os mecanismos de inundação de pacotes por toda a malha. Em vez de exigir que cada nó realize a detecção de endereço duplicado, o 6LoWPAN-ND usa uma abordagem de *registro* em que um nó na malha pode registrar seu endereço com um software que é executado no roteador de borda. Cada nó registra seu endereço e o software no roteador de borda sinaliza qualquer duplicata (lembre-se de que um roteador de borda tem poder de processamento e memória necessária para lidar com serviços em toda a rede, como o registro de endereço). Finalmente, o 6LoWPAN-ND permite que os nós *durmam* (ou seja, entrem em *stasis state*, ou estágio de êxtase, para economizar energia). Quando o nó desperta, ele renova seu registro de endereço.

30.12.3 Estabelecimento de Link Mesh (MLE)

O IPv6 foi concebido com o pressuposto de que os links tenham sido configurados antes de executar o software IP. Em particular, a conexão de rádio entre dois nós não pode ser avaliada pelo envio de um pacote em uma direção: é necessária uma troca em duas direções. Além disso, o IPv6 considera que as trocas sejam autenticadas, o que significa que os links já devem estar operacionais. Para permitir um route-over mesh IPv6, é requerido que pares de nós possam usar um protocolo de baixo nível para testar e configurar os links entre eles.

O ZigBee usa o protocolo *estabelecimento de link de malha* (*Mesh*

Link Establishment – MLE) para a configuração do link. O MLE permite que dois nós avaliem a qualidade do sinal durante a transmissão de pacotes, troquem informação de configuração (inclusive informação de endereço) e escolham um tipo de segurança para o link. Além disso, o MLE pode avaliar vários links que levam ao roteador de borda e escolher um para usar.

30.13 Encaminhamento em uma rota de malha ZigBee

Talvez a mais séria consequência de usar o IPv6 para implementar uma malha route-over surja a partir da necessidade de roteamento de origem. Sabemos que alguns nós da rede serão on link (ou seja, diretamente acessíveis), mas outros estarão off link (ou seja, só acessíveis indiretamente). Portanto, o IP deve usar mais do que o prefixo de rede para decidir como encaminhar um datagrama. Há duas maneiras de lidar com o problema: o *modo armazenamento* e o *modo não armazenamento*.

Modo armazenamento. Como o termo armazenamento indica, cada nó na rede armazena uma quantidade significativa de informação. Em particular, cada nó aprende e armazena a topologia de toda a rede (ou seja, que pares de nós podem se comunicar). Quando se necessita enviar um datagrama, o software IP consulta a informação armazenada para obter um caminho através da malha para o destino. O remetente coloca o caminho em um cabeçalho de rota de origem e envia o datagrama.

Modo não armazenamento. No modo não armazenamento, cada nó na rede só aprende duas coisas: o conjunto de vizinhos diretamente alcançáveis e a identificação de um vizinho que leva ao roteador de borda. No entanto, o roteador de borda (que se presume ter significativamente mais capacidade de memória e de processamento do que nós individuais) aprende a topologia completa da rede. Quando um nó tem um datagrama para enviar, o encaminha para o roteador de borda que, por conhecer a topologia, usa o roteamento de origem para enviar o datagrama ao seu destino.

Pode parecer que o modo não armazenamento desperdiça recursos de rede porque um datagrama enviado a partir de um nó para outro vai primeiro para o roteador de borda e, em seguida, ao seu destino. No entanto, o modo de não armazenamento foi selecionado para permitir que nós individuais fossem extremamente pequenos (tem memória limitada). Embora se espere que as redes devam ter apenas alguns nós (talvez duas dezenas), o tamanho é importante. Mesmo que uma rede contenha apenas alguns nós, a malha de nós não terá memória suficiente para armazenar

tabelas de encaminhamento para toda a malha; para uma malha maior, o modo de armazenamento implica que cada nó deve armazenar tabelas extremamente grandes. Usar o modo de não armazenamento significa que cada nó na rede só precisa manter os seguintes itens na memória:

- uma lista de todos os vizinhos diretamente alcançáveis, incluindo o endereço MAC de cada um;
- a identidade do vizinho que foi escolhido como o caminho para o roteador de borda.

30.13.1 Protocolo de roteamento para redes de baixa potência e com perdas (RPL)

O IETF definiu um protocolo que pode ser usado com o IPv6 em uma malha route-over. Conhecido como protocolo de roteamento para redes de baixa potência e com perdas (*Routing Protocol For Low-Power And Lossy Networks – RPL*), o protocolo permite que os nós anunciem conexões diretas e aprendam sobre conexões. No modo não armazenamento, RPL propaga informações de conexão "para cima" para o roteador de borda. O roteador de borda executa uma versão especial do software RPL que reúne as informações, o que significa que ele aprende a topologia da malha inteira.

Uma vez que aprende a topologia, o roteador de borda calcula uma árvore de encaminhamento. Em vez de uma árvore e um grafo não direcionado, o RPL faz cada link direcionado e a direção é para a raiz (ou seja, em direção ao roteador de borda). Assim, o RPL chama o grafo da topologia de malha um *Destination-Oriented Directed Acyclic Graph* (*DODAG*). A Figura 30.3 ilustra a forma DODAG da árvore da Figura 30.1.*

A representação DODAG é meramente um detalhe do protocolo. Quando um roteador de borda precisa enviar um datagrama para um dos nós, a rota de origem usada no cabeçalho lista os nós abaixo da árvore (ou seja, no sentido inverso das setas na figura).

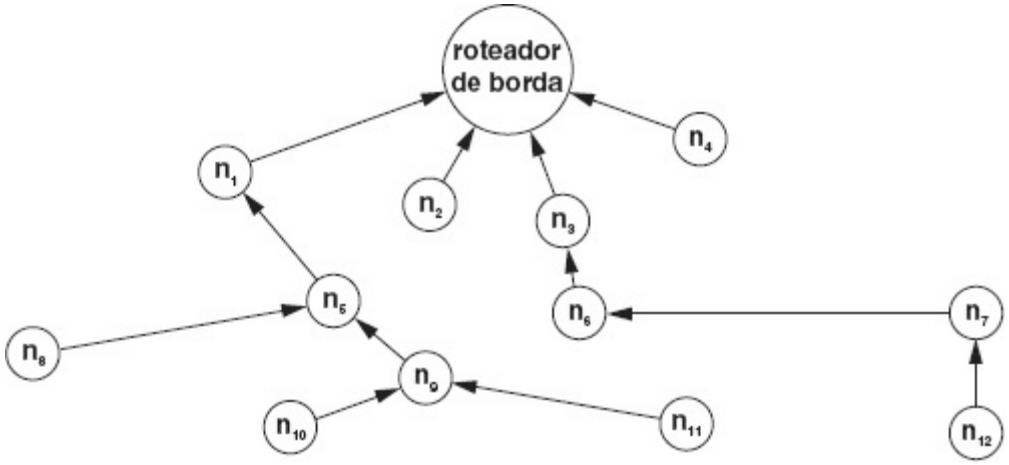


Figura 30.3 O DODAG que o RPL define para a árvore na figura 30.1.

O RPL separa nós na árvore em três tipos: *raiz* (o roteador de borda funciona como a raiz do DODAG), *folha* (ou seja, um nó que tem apenas uma conexão) e *intermediário* (um nó que tem pelo menos duas conexões e encaminha data-gramas). Como nós intermediários encaminham pacotes, os padrões ZigBee usam o termo roteador IP ZigBee, roteador ZIP abreviado. Nós folha não executam RPL. Em vez disso, cada nó folha conecta a um roteador ZIP, que lida com toda a responsabilidade de encaminhamento para um nó folha. Ou seja, um roteador ZIP informa a um roteador de borda sobre cada nó folha a que se conecta.

O RPL é muito mais complexo do que o indicado aqui. Por exemplo, o RPL também pode ser utilizado com o modo de armazenamento; mensagens devem especificar o modo como está sendo utilizado. Além disso, o RPL tem a capacidade de distribuir a informação para baixo na árvore. Por exemplo, é possível utilizar RPL para informar aos nós sobre prefixos IP que estão sendo usados.

30.13.2 Outros protocolos na especificação ZigBee

Além dos principais protocolos definidos acima, a especificação ZigBee inclui muitos outros protocolos. O ZigBee usa IPv6 e ICMPv6, TCP, UDP, *Protocolo para levar Autenticação para Acesso à Rede* (*Protocol for Carrying Authentication for Network Access – PANA*), *DNS multicast* (*mDNS*), *DNS Service Discovery* (*DNS-SD*) e *Transport Layer Security* (*TLS*), que é usado em conjunto com a PANA, EAP, e EAP-TLS para

autenticação. Finalmente, os aplicativos que estejam em conformidade com a *Energy Smart Profile 2.0* podem usar HTTP. A Figura 30.4 resume os principais protocolos.

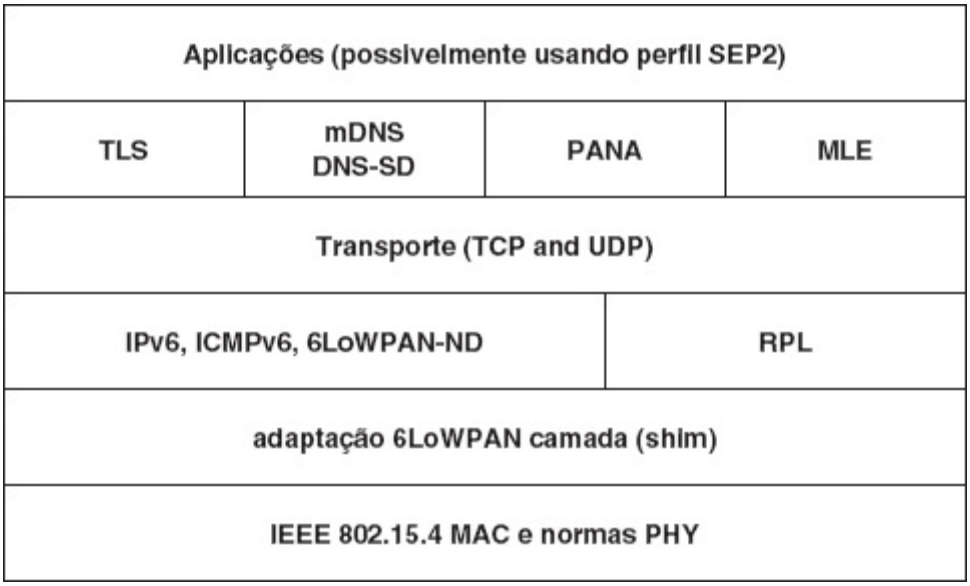


Figura 30.4 A camada dos principais protocolos usados em uma rede de malha ZigBee.

30.14 Avaliação de uso de rota IPv6 para uma malha

Muitas perguntas surgem sobre a abordagem route-over e o uso de IPv6 para uma malha de nós sensores de baixa potência. Um projeto route-over é melhor do que um projeto mesh-under? Quanto de sobrecarga de protocolo adicional é necessário para route-over? Uma implementação route-over exigirá mais memória do que a implementação mesh-under? Se assim for, quanto mais? Faz sentido usar o IPv6 em uma rede de malha que tem um MTU extremamente pequeno e links extremamente lentos? Se RPL usa o modo não armazenamento, o que é a sobrecarga em termos de encaminhamento de pacote adicional? Quanto do IPv6 e do protocolo de suporte IPv6 deve mudar para torná-los operacionais em um ambiente de malha?

Já vimos que a especificação IPv6 proíbe o uso de IPv6 em uma rede com um MTU menor que 1280. Assim, uma camada shim 6LoWPAN é necessária para permitir que o IPv6 use uma tecnologia que tem um MTU de apenas 127 (menos de 127, se uma autenticação de cabeçalho é usada). Outra complicação com MTU pequeno surge porque redes 802.15.4 podem ter perdas. Embora a especificação 6LoWPAN reconheça o comportamento de perdas, ela depende da utilização de fraglets, o que significa que a

probabilidade de perda de datagramas (ou atraso para retransmissão) será muito maior do que a probabilidade de perda de um único pacote.

Outro problema relacionado com MTU surge porque um roteador de borda deve encaminhar datagramas que chegam de fora da malha. O padrão IPv6 especifica um link MTU mínimo de 1280. Suponha que o roteador de borda anuncie um MTU de 1280 em links externos. Quando um datagrama chega a partir do exterior, deve ser encapsulado antes da transmissão através da malha. Infelizmente, o cabeçalho adicional aumenta o tamanho do datagrama para 1280 +, o que faz com que seja maior do que o MTU do 6LoWPAN de 1280. Uma solução define que o 6LoWPAN MTU deve ser 1280 +, mas exige que o roteador de borda imponha um MTU de 1280 para fontes externas. A desvantagem da incorporação de restrições especiais no código IP surge da falta de generalidade – o uso de técnicas não padronizadas para lidar com uma malha torna a pilha de protocolos quebradiça. Por exemplo, se alguém inventa um novo mecanismo de descoberta de caminho MTU, o novo mecanismo não pode ser integrado ao roteador de borda até que tenha sido modificado para atender as regras especiais de MTU.

Desenvolvedores perceberam que protocolos IPv6 convencionais não podem ser usados para configurar um link de rádio ou para realizar uma avaliação de sinais de mão dupla, então eles criaram MLE. Eles também tiveram que substituir a Descoberta de Vizinho IPv6, porque nós na malha compartilham um prefixo IP, mas não compartilham um único domínio de broadcast (ou seja, alguns nós estão off link). Outras dúvidas sobre protocolos surgiram porque o RPL e o ICMPv6 ambos suportam propagação para baixo de informações de rede (por exemplo, prefixos de endereços). Qual deve ser usado?

Uma ineficiência final no uso de IPv6 surge a partir de uma decisão de projeto fundamental: um cabeçalho de datagrama IPv6 não pode ser modificado. Sabemos que, para transitar na malha, um datagrama deve incluir um cabeçalho de rota de origem e também transportar informações RPL. No entanto, essa informação só é relevante dentro da malha. Mais importante, a informação não pode ser enviada para fora da malha porque o datagrama poderia passar através de outra rede ZigBee – os cabeçalhos extras devem ser removidos para evitar serem mal interpretados. Da mesma forma, se um datagrama passa do exterior para dentro da malha, devem ser adicionados os cabeçalhos apropriados. Como consequência do projeto IPv6, os cabeçalhos não podem ser mudados. Portanto, a única opção é encapsulamento: cada datagrama IPv6 enviado através da malha deve ser encapsulado num datagrama IPv6 externo que tem os cabeçalhos

apropriados. Em redes normais, o encapsulamento IP em IP adiciona uma pequena sobrecarga. Para uma rede 802.15.4, no entanto, o cabeçalho adicional é enorme em relação ao MTU. Como resultado, a sobrecarga aumenta significativamente. Podemos resumir a seguir.

Embora seja possível usar um paradigma route-over e IPv6 para uma malha ZigBee, fazer isso significa criar alguns protocolos alternativos e incorrer em sobrecarga significativa.

30.15 Resumo

A Internet, em seu início, se concentrou em conectar usuários a recursos remotos. Uma nova tendência se concentra em uma Internet das coisas, em que sistemas embarcados inteligentes que detectam e controlam seu ambiente usam tecnologia de Internet para se comunicar. Exemplos incluem sensores em veículos, residências, prédios de escritórios, shopping centers e na infraestrutura civil.

Um consórcio de fornecedores conhecido como o ZigBee Alliance está explorando o uso da tecnologia de rede sem fio IEEE 802.15.4 com uma malha de sensores Smart Grid. Uma rede Zig-Bee tem um roteador de borda que se conecta ao exterior; outros nós da malha se auto-organizam para se conectarem e estabelecerem o encaminhamento.

Em termos de protocolos, o ZigBee Alliance está trabalhando com o IETF em uma abordagem route-over que usa IPv6. Em princípio, um sistema de route-over usa encaminhamento de Camada 3. Na prática, o IPv6 e os protocolos de suporte associados são insuficientes para tecnologia de malha sem fio de baixa potência, com perdas que tem um MTU baixo e uma baixa taxa de dados. Consequentemente, o trabalho se concentrou na substituição de muitas partes do IPv6, adicionando uma camada shim para acomodar MTU pequeno, e criando um novo protocolo de roteamento de malha. Mesmo com as mudanças, o projeto do IPv6 introduz muitas ineficiências.

EXERCÍCIOS

30.1 Suponha que seu carro tenha um sistema embarcado que pode reconhecer o seu rosto e configurar o veículo de acordo com você. Quais parâmetros você escolheria para definir?

30.2 Um dos primeiros usos de sistemas embarcados conectados surgiu de caixas registradoras. Quando um consumidor faz uma compra, a

caixa registradora o identifica (por exemplo, a partir de um ID cliente frequente ou cartão de crédito) e envia detalhes da compra de volta a um servidor. Encontre exemplos de outros sistemas embarcados conectados que os consumidores utilizam.

30.3 Câmeras estão sendo colocadas em supermercados para monitorar os hábitos dos compradores. Se a loja também instalar câmeras nos caixas, será possível conhecer os clientes. Quais são as consequências para a privacidade?

30.4 Se você tivesse que criar protocolos para uma rede de malha, você escolheria mesh-under ou route-over? Justifique sua escolha.

30.5 Se os aplicativos assumem um MTU de 1280 e um datagrama IPv6 original deve ser encapsulado em um datagrama IPv6 externo, quantos fraglets devem ser enviados para transmitir o datagrama?

30.6 Suponha que um datagrama seja dividido em K fraglets e a probabilidade de perder um determinado fraglet seja p (0 p 1). Qual é a probabilidade de que o datagrama seja perdido?

30.7 Quais são as vantagens de usar o IPv6 como a base para uma Internet das coisas? Quais são as desvantagens?

30.8 Considere a escolha de caminhos em uma rede mesh. Uma métrica possível escolhe um caminho com menor número de saltos. Explique por que uma métrica de contagem de saltos nem sempre funciona bem em uma rede mesh.

30.9 O trabalho em uma pilha ZigBee resultou em mais um protocolo de roteamento, RPL. Por que foi criado um novo protocolo em vez de serem usados protocolos de roteamento existentes (por exemplo, RIP, OSPF)?

30.10 Compare o modelo 6LoWPAN-ND de registro de endereço com o uso de DHCP para atribuição de endereço. Quais são as diferenças? Em que circunstâncias cada um é preferível?

---

* Apesar de soar estranha, a expressão *Internet das coisas* parece ser amplamente aceita. No entanto, o departamento de marketing de um fornecedor usa uma variação: a *Internet de tudo*.

* Os exemplos apresentados aqui não são meras especulações – esses sistemas existem e estão sendo implantados.

* A Figura 30.1 pode ser encontrada na página 430.

APÊNDICE 1

Glossário de termos e abreviações de interligação de redes

Terminologia TCP/IP

Como a maioria dos grandes empreendimentos, o TCC/IP e a Internet também criaram sua própria linguagem. Com uma curiosa mistura de jargão de redes, nomes de protocolo e abreviaturas, a terminologia é, ao mesmo tempo, difícil de aprender e também de lembrar. Para aqueles que não estão familiarizados, as discussões entre os conhecedores parecem tolices desprovidas de sentido, recheadas de siglas a cada oportunidade. Mesmo depois de já estarem há um tempo convivendo com os termos, os leitores talvez ainda os achem difíceis de entender. E o problema se agrava devido ao fato de alguns termos serem vagamente definidos, surgindo em uma quantidade esmagadora, além da existência de vários subgrupos que tentam redefinir tais palavras e expressões de tempos em tempos.

Este glossário visa ajudar a resolver o problema, por meio de definições curtas relativas aos termos usados pelos protocolos e pela Internet. No entanto, o glossário não pretende ser um tutorial para iniciantes. Em vez disso, vamos nos concentrar em fornecer referências concisas àqueles que, geralmente, estão bem informados acerca da rede, para que assim possam procurar o significado de termos ou mesmo de siglas específicas com rapidez. Os leitores perceberão que as referências são substancialmente mais úteis depois que estudarem o texto.

Glossário de termos e abreviações em ordem alfabética

10/100/1000 hardware

Um termo aplicado a hardware Ethernet que negocia se a operação será em 10 Mbps, 100 Mbps ou 1 Gbps.

1000Base-T, 1000Base-X

Os nomes técnicos para Gigabit Ethernet sobre par trançado e fibra ótica.

10GigE

O nome usado para Ethernet que opera a 10 Gbps.

127.0.0.1

O endereço de *loopback* IP usado para testes. Os pacotes enviados para esse endereço são processados pelo protocolo local sem sequer serem enviados através de uma rede.

1280 (MTU)

O tamanho de datagrama mínimo que todos os hosts e roteadores IPv6 precisam manipular.

2822 mail

O formato-padrão para mensagens de e-mail (de RFC 2822).

3-way handshake

A técnica usada em TCP para iniciar ou terminar uma conexão com segurança.

5-layer reference model

A troca de três segmentos que o TCP usa para iniciar de modo confiável ou terminar elegantemente uma conexão.

5 tuplas (5-tuple)

Uma referência aos cinco pontos necessários para identificar uma conexão TCP endereços IP de origem e destino; portas protocolo de origem e destino e o tipo de carga útil.

576 (MTU)

O tamanho de datagrama mínimo que todos os hosts e roteadores IPv4 precisam manipular.

7-layer reference model

Um modelo de protocolo de estratificação antigo criado pela ISO. Ver *5-layer reference model.*

6LoWPAN

Um padrão utilizado para enviar IPv6 através de redes sem fio de baixa potência.

802.11

Um prefixo para padrões do IEEE que especificam redes sem fio. Os padrões específicos incluem 802.11a, 802.11b, 802.11g, 802.11i e 802.11n.

802.15.4

Um padrão IEEE para a tecnologia de rede sem fio de baixa potência.

802.3

O padrão do IEEE para Ethernet.

abertura ativa (active open)

A operação que um cliente realiza para estabelecer uma conexão TCP com um servidor em um endereço conhecido.

abertura passiva (passive open)

A operação que o servidor executa para esperar em uma porta conhecida por uma conexão TCP.

ABR (Available Bit Rate)

Uma designação ATM para serviço que não garante uma velocidade. Também *Area Border Router*, uma designação OSPF para um roteador que se comunica com outra área.

acessibilidade/alcançabilidade

Um destino é "alcançável" a partir de uma determinada origem se existir uma rota da origem até o destino.

ACK

Abreviatura para acknowledgement (reconhecimento).

acordo de peering

Um acordo de cooperação entre dois ISPs para trocar informações de acessibilidade e também pacotes de dados. Além do peering nos IXPs, os grandes ISPs normalmente possuem acordos de peering privativos.

ADSL (Asymmetric Digital Subscriber Line)

Uma tecnologia popular para enviar dados simultaneamente através dos mesmos fios usados para serviço telefônico.

AFRINIC

Veja *registro*.

agente

No gerenciamento de redes, um agente é o software servidor executado em um dispositivo de rede que está sendo gerenciado.

agregação de rota (route aggregation)

A técnica usada por protocolos de roteamento para combinar vários destinos que possuem o mesmo próximo salto em uma única entrada.

AH (Authentication Header)

Um cabeçalho usado pelo IPSec para garantir a autenticidade da origem de um datagrama.

AIMD (Additive Increase Multiplicative Decrease)

Uma referência ao mecanismo de controle de congestionamento do TCP em que a janela de congestionamento diminui rapidamente e aumenta lentamente.

algoritmo de Ford-Fulkerson

Um sinônimo para o algoritmo de vetor de distância que se refere aos pesquisadores que o descobriram.

algoritmo de Karn

Um algoritmo que permite que os protocolos de transporte distingam entre amostras de tempo de ida e volta válidas e inválidas e, portanto, melhorem as estimativas de ida e volta.

algoritmo de Nagle

Uma heurística de automedição que agrupa dados de saída para melhorar a vazão e evitar a síndrome da janela tola.

ANSI (American National Standards Institute)

Uma organização que define os padrões norte-americanos para o setor de processamento de informações.

anúncio de janela (window advertisement)

Um valor utilizado pelo TCP para permitir a um receptor informar a um emissor o tamanho de um buffer disponível.

anycast

Uma forma de endereço introduzida com o IPv6 em que um datagrama enviado para o endereço pode ser roteado para qualquer um de um conjunto de computadores. Um endereço anycast era originalmente chamado de um endereço de cluster.

API (Application Program Interface)

A especificação da interface usada por uma aplicação. A API de soquete é a mais utilizada para comunicação da Internet.

API de socket

A definição de uma interface entre programas de aplicativos e TCP/IP.

APNIC

Veja *registro*.

área

No OSPF, um grupo de roteadores que troca informações de roteamento.

ARIN

Veja *registro*.

armazenar e encaminhar (store-and-forward)

O modelo usado pelos sistemas de comutação de pacotes em que um pacote que chega é armazenado na memória até que possa ser encaminhado em direção ao seu destino.

ARP (Address Resolution Protocol)

O protocolo usado para ligar dinamicamente um endereço IPv4 a um endereço hardware (MAC). O ARP é usado através de uma única rede física e limita-se às redes que aceitam broadcast de hardware.

ARPA (Advanced Research Projects Agency)

A agência governamental que fundou a ARPANET e, posteriormente, a Internet. A ARPA também é conhecida como DARPA.

ARPANET

Uma rede de longa distância pioneira fundada pela ARPA e construída pela BBN. Ela operou de 1969 a 1990 como base para pesquisas iniciais de rede e como um backbone central durante o desenvolvimento da Internet. A ARPANET consistia em nós de comutação de pacotes individuais interconectados por linhas alugadas.

AS (Autonomous System)

Uma coleção de roteadores e redes que estão sob uma entidade administrativa e cooperam intimamente para propagar informações de acessibilidade de rede (e roteamento) entre eles mesmos usando um protocolo de gateway interior de sua própria escolha. Os roteadores

dentro de um sistema autônomo possuem um alto grau de confiança. Para que dois sistemas autônomos possam se comunicar, um roteador em cada sistema envia informações de acessibilidade para um roteador no outro.

AS (Security Association)

Usada com o IPSec para indicar uma vinculação entre um conjunto de parâmetros de segurança e um identificador transportado em um cabeçalho de datagrama. Um host escolhe vinculações SA; elas não são globalmente padronizadas. Veja *SPI*.

ASN.1 (Abstract Syntax Notation.1)

O protocolo padrão de apresentação ISO usado pelo SNMP para representar mensagens.

Associação de segurança (Security Association)

Veja *SA*.

ATM (Asynchronous Transfer Mode)

Uma tecnologia de rede orientada para conexão que usa pequenas células de tamanho fixo na camada inferior.

atraso (delay)

Uma das duas principais medidas de uma rede. O retardo se refere à diferença entre o momento em que um bit de dados é injetado em uma rede e o momento em que ele sai.

atualizações acionadas

Uma heurística usada com protocolos de vetor de distância, como RIP, para enviar uma atualização assim que as rotas mudam, sem esperar o próximo ciclo de atualização.

automedição

Característica de qualquer sistema que opera periodicamente sem exigir um relógio externo (por exemplo, usa a chegada de um pacote para disparar uma ação).

backoff exponencial binário (binary exponential backoff)

Uma técnica usada para controlar rapidamente contenção ou congestionamento de rede. Um emissor dobra o período de tempo que ele espera entre cada retransmissão sucessiva.

backoff exponencial

Veja *backoff exponencial binário*.

banda base (baseband)

Característica de qualquer tecnologia de rede, como Ethernet, em que todas as estações compartilham acesso com o meio. Compare com *banda larga*.

banda larga (broadband)

Característica de uma tecnologia de rede que multiplexa várias operadoras de rede independentes em um único cabo (normalmente usando multiplexação de divisão de frequência). Compare com *banda base*.

base64

Uma codificação usada com o MIME para enviar dados não textuais, como um arquivo binário, através de e-mail.

BCP (Best Current Practice)

Um rótulo dado a um subconjunto de RFCs que contém recomendações do IETF sobre o uso, a configuração ou o emprego de tecnologias de rede.

Bellman-Ford

Um sinônimo para vetor de distância.

BGP (Border Gateway Protocol)

O principal exterior gateway protocolo usado na Internet. Quatro versões principais do BGP apareceram, sendo o BGP-4 a mais moderna.

big endian

Um formato de armazenamento ou transmissão de dados binários em que o byte mais significativo (bit) vem primeiro. A ordem de byte de rede padrão do TCP/IP é o big endian. Compare com *little endian*.

bind

O nome do popular software DNS, originalmente derivado do software no BSD Unix.

Bps (bits por segundo)

Uma medida da velocidade de transmissão de dados.

broadcast and prune

Uma técnica usada no encaminhamento multicast baseado em dados, em que roteadores encaminham cada datagrama para cada rede até que aprendem que a rede não possui membros de grupo.

broadcast

Dizemos que um sistema de entrega de pacotes que entrega uma cópia de um determinado pacote para todos os hosts que se conectam a ele o difundirem (broadcast). O broadcast pode ser implementado por hardware (por exemplo, como na Ethernet) ou por software (por exemplo, broadcasting IP na presença de sub-redes). Veja *multicast* e *unicast*.

BSD UNIX (Berkeley Software Distribution UNIX)

Uma versão do sistema operacional UNIX modificada na U.C. Berkeley para conter protocolos TCP/IP.

buffer bloat

Um termo usado para criticar os vendedores que usam grandes buffers de pacote, mesmo em pequenos dispositivos (por exemplo, um modem a cabo ou um roteador sem fio). Grandes buffers podem causar longos atrasos que afetam a comunicação em tempo real.

cabeçalho

Informações no início de um pacote ou de uma mensagem que descrevem o conteúdo e especificam um destino.

Cabeçalho-base (base header)
O cabeçalho obrigatório encontrado no início de cada datagrama IPv6.

cabeçalho de extensão
Qualquer um dos cabeçalhos IPv6 opcionais que seguem o cabeçalho-base.

cabeçalho de extensão de fragmento
Um cabeçalho opcional usado pelo IPv6 para marcar um datagrama como um fragmento.

cabo de categoria 5
O padrão para cabeamento usado com Ethernet de par trançado. As variantes incluem Cat5, Cat5e e Cat6, sendo que cada uma fornece mais velocidade do que a categoria anterior.

camada 1
Uma referência à camada mais baixa em um modelo em camadas. A camada 1 inclui conexões físicas e voltagens em fios.

camada 2
Uma referência à segunda camada mais baixa em um modelo em camadas. A camada 2 inclui formato de quadro e endereçamento.

camada 3
Uma referência à camada IP, que fornece interligação à Internet.

camada 4
Uma referência à camada de transporte, que inclui TCP e UDP.

caminho mais curto primeiro (shortest path first)
Veja *SPF*.

CBT (Core Based Trees)
Um protocolo de roteamento multicast baseado em demanda que constrói árvores de encaminhamento compartilhado.

CCITT (Consultative Committee on International Telephony and Telegraphy)

O antigo nome da ITU.

célula cell

Pequeno pacote de tamanho fixo. Uma célula ATM contém 48 octetos de dados e 5 octetos do cabeçalho.

CGI (Common Gateway Interface)

Uma tecnologia que um servidor utiliza para criar uma página web dinamicamente quando a requisição chega.

checksum

Um pequeno valor inteiro enviado com um pacote e usado para detectar erros que podem resultar quando um pacote é transmitido de uma máquina para outra.

CIDR (Classless Inter-Domain Routing)

O padrão que especifica os detalhes do endereçamento sem classe e de um esquema de roteamento associado.

classe de endereço (class of address)

No endereçamento com classe, a classe de um endereço determinava o local do limite entre o prefixo da rede e o sufixo do host.

classe de tráfego

Uma referência a um conjunto de serviços disponíveis na interpretação DiffServ.

cliente-servidor (client-server)

O modelo de interação em um sistema distribuído no qual um programa cliente em um computador envia uma requisição para um programa servidor em outro computador e espera uma resposta.

CO

Veja *serviço orientado à conexão*.

codec (codificador/decodificador)

Um dispositivo de hardware usado para converter entre um sinal de áudio analógico e um fluxo de valores digitais.

codificação TLV (TLV encoding)

Qualquer formato de representação que codifica cada item com três campos: um tipo, um tamanho e um valor.

comutação IP

Originalmente, uma tecnologia de encaminhamento IP de alta velocidade, desenvolvida pela Ipsilon Corporation; agora o termo é geralmente usado em referência a qualquer uma das várias tecnologias semelhantes.

conexão

Uma abstração fornecida pelo protocolo. O TCP fornece uma conexão de um programa aplicativo para outro.

congestionamento

Uma situação em que o tráfego excede (temporariamente) a capacidade das redes ou roteadores.

contagem de salto (hop count)

Uma medida de distância entre dois pontos na Internet. Uma contagem de salto de n significa que n redes separam a origem e o destino.

contagem para o infinito (count to infinity)

Um nome informal para o problema da convergência lenta.

controle de fluxo (flow control)

Controle exercido por um receptor, normalmente em um protocolo de transporte, para limitar a velocidade em que um emissor transfere dados.

convergência lenta (slow convergence)

Um problema nos protocolos de vetor de distância em que dois ou mais roteadores formam um loop de encaminhamento que persiste até que os

protocolos de roteamento aumentem a distância até o infinito.

COPS (Common Open Policy Service)
Um protocolo usado com o RSVP para verificar se uma requisição atende às restrições de política.

correspondência de prefixo mais longo (longest-prefix match)
Veja *LPM*.

CoS (Class of Service)
A designação atribuída a um pacote que determina a sua prioridade e como ele é encaminhado.

CRC (Cyclic Redundancy Code)
Um valor pequeno e inteiro enviado com um pacote, que é usado para detectar erros de transmissão. Veja também *checksum*.

criptografia por chave pública (public key encryption)
Uma técnica de criptografia que gera chaves de criptografia em pares. Um dos pares precisa ser mantido secreto e o outro é publicado.

CR-LF (Carriage Return-Line Feed)
A sequência de dois caracteres usada para terminar cada linha de texto nos protocolos de camada de aplicação, como TELNET e SMTP.

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)
Uma característica do hardware de rede que opera permitindo que várias estações disputem o acesso a um meio de transmissão escutando para ver se o meio está desocupado, e enviando uma breve mensagem de reserva para informar outras estações antes de transmitir dados. As redes sem fio 802.11 utilizam o CSMA/CA.

CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
Uma característica do hardware de rede que opera permitindo que várias estações disputem o acesso a um meio de transmissão escutando para ver se o meio está desocupado, e um mecanismo que permite que o hardware detecte quando duas estações tentam transmissão

simultaneamente. A Ethernet usa o CSMA/CD.

CSU/DSU (Channel Service Unit/Data Service Unit)

Um dispositivo eletrônico que conecta um computador ou roteador a um circuito digital alugado pela companhia telefônica. Embora o dispositivo preencha duas listas, ele normalmente consiste em um único hardware físico.

dados fora de banda (out-of-band data)

Dados enviados fora do caminho de distribuição normal, normalmente usados para transportar indicadores de erro (por exemplo, dados urgentes do TCP).

dados urgentes

Um recurso que o TCP usa para enviar dados fora de banda.

DARPA (Defense Advanced Research Projects Agency)

A agência governamental que criou a ARPANET e, posteriormente, a Internet. A DARPA também é conhecida como ARPA.

data de época

Um ponto na história escolhido como a data a partir da qual o tempo é medido. O TCP/IP usa 1º de janeiro de 1900, Hora Universal (antigamente chamada Hora de Greenwich), como sua data de época.

datagrama

Veja *datagrama IP*.

datagrama IP

A unidade básica de informações passadas através de uma Rede TCP/IP, análoga a um pacote de hardware. Cada datagrama contém os endereços do emissor original e do destino final.

DCE (Data Communications Equipment)

Equipamento na rede, em oposição ao equipamento do usuário, que se conecta à rede. Veja também *DTE*.

demultiplexar

Separar uma entrada comum em várias saídas. A demultiplexação ocorre em cada camada da pilha de protocolos. Veja *multiplexar*.

desligamento elegante (graceful shutdown)

Um mecanismo de protocolo que permite a duas partes comunicando-se concordarem em terminar a comunicação sem confusão, mesmo que os pacotes subjacentes estejam perdidos, retardados ou duplicados. O TCP usa um handshake de três vias para garantir o desligamento elegante.

DHCP (Dynamic Host Configuration Protocol)

Um protocolo que um host usa para obter informações de configuração, incluindo um endereço IP, o endereço de um roteador e o endereço de um servidor de nome de domínio.

DiffServ (Differentiated Services)

A definição atualmente usada para tipo de serviço em um datagrama IP. O DiffServ usa 6 bits no cabeçalho IP.

dig

Uma aplicação usada por gerentes de rede para consultar o sistema de nome de domínio.

divisão identificador-localizador (identifier-locator split)

A ideia de que, para apoiar a mobilidade, um endereço IP deve ser dividido em uma parte que identifica um nó e outra que especifica onde o nó está atualmente localizado.

DNS (Domain Name System)

O sistema usado para traduzir nomes de máquina legíveis por humanos para endereços IP. O DNS usa uma hierarquia de servidores.

DNS-SD (Domain Name Service System Discovery)

Um protocolo que utiliza mDNS para encontrar um serviço.

DNSSEC (Domain Name System SECurity)

Um conjunto de extensões de segurança para o sistema de nome de

domínio.

DPI (Deep Packet Inspection)
Um mecanismo de segurança que considera artigos que se encontram na área de carga útil de um pacote, assim como o cabeçalho.

DRR (Deficit Round Robin)
Um algoritmo usado para formatação de tráfego.

DSL (Digital Subscriber Line)
Um conjunto de tecnologias usadas para fornecer serviço de dados de alta velocidade através dos fios de cobre que interligam escritórios de telefonia, residências locais ou empresas.

DTE (Data Terminal Equipment)
Equipamento que se conecta a uma rede em oposição ao equipamento que a forma. Veja também *DCE*.

DVMRP (Distance Vector Multicast Routing Protocol)
Um protocolo usado para propagar rotas de multicast.

E.164
Um formato de endereço especificado pela ITU e usado com a ATM.

ECN (Explicit Congestion Notification)
Um mecanismo que permite que um roteador informe sistemas finais quando um pacote encontra congestionamento.

EF (Expedited Forwarding)
A classe DiffServ para pacotes de alta prioridade.

EGP (Exterior Gateway Protocol)
Um termo aplicado a qualquer protocolo usado por um roteador em um sistema anônimo para anunciar acessibilidade de rede a um roteador em outro sistema autônomo. O BGP-4 é o exterior gateway protocol mais usado atualmente.

emenda de TCP (TCP splicing)

Uma técnica em que um sistema intermediário reescreve segmentos TCP para mapear de um espaço de sequência para outro. A técnica é utilizada em sistemas que estabelecem duas conexões TCP e, em seguida, mapear uma para dentro da outra de modo que nenhum ponto de extremidade está ciente da emenda.

encaminhamento (forwarding)

O processo de aceitar um pacote que chega, consultando um próximo salto em uma tabela de roteamento e enviando o pacote para o próximo salto.

encaminhamento de próximo salto (next-hop forwarding)

O modelo utilizado na Internet, no qual cada roteador só sabe encaminhar um pacote por um salto.

encapsulamento

A técnica usada pelos protocolos de camada em que um protocolo de nível inferior aceita uma mensagem de um protocolo de nível superior e a coloca na parte de dados do pacote de baixo nível.

endereçamento classful

O esquema de endereçamento IPv4 original em que os endereços de host eram divididos em três classes: A, B e C.

endereçamento classless

Uma extensão do esquema de endereçamento IPv4 original que ignora os limites de classe originais para aproveitar melhor o espaço de endereço.

endereçamento de sub-rede

Uma extensão do esquema de endereçamento IP que permite que um local use um único endereço de rede IP para várias redes físicas.

endereçamento de super-rede

Outro nome para *CIDR*.

endereçamento hierárquico (hierarchical addressing)

Um esquema de endereçamento no qual um endereço pode ser subdividido em partes, e cada uma identifica uma granularidade sucessivamente mais fina. Os endereços IPv4 usam uma hierarquia de dois níveis em que a primeira parte do endereço identifica uma rede e a segunda parte identifica um host específico nessa rede; a tecnologia de sub-rede introduziu um nível adicional da hierarquia de endereçamento.

endereço

Um valor inteiro usado para identificar um determinado computador. Cada pacote enviado para o computador contém o endereço.

endereço care-of

Um endereço IP temporário usado por um móvel enquanto este visita uma rede estrangeira.

endereço de broadcast direcionado (directed broadcast address)

Um endereço IPv4 que especifica "todos os hosts" em uma rede específica. O uso do broadcast direcionado não é aconselhável.

endereço de escopo

Um termo usado com IPv6 para se referir a um endereço que é limitado a um link, um site, ou a uma organização.

endereço de hardware

Um endereço arbitrário usado por uma rede física (por exemplo, o Ethernet usa um endereço de hardware de 48 bits). São sinônimos endereço físico e endereço MAC.

endereço de IP (IP address)

Um número binário atribuído a cada host na Internet. Um endereço IPv4 é de 32 bits; um endereço IPv6 é de 128 bits.

endereço de link-local (link-local address)

Um endereço usado com o IPv6 que só possui importância em uma rede única.

endereço de loopback

Um endereço de rede usado para teste, que faz com que pacotes saindo sejam distribuídos para a máquina local. O IP usa 127.0.0.0 como o prefixo de loopback.

endereço de rede 10

Uma referência geral a um endereço não roteável (ou seja, que está reservado para uso em uma intranet e não usado na Internet).

endereço físico

Um sinônimo para endereço MAC ou endereço de hardware.

endereço Internet

Veja endereço *IP*.

endereço não roteável (nonroutable address)

Qualquer endereço usando um prefixo reservado para uso em intranets. Os roteadores na Internet não encaminharão datagramas que contenham um endereço não roteável. Veja *endereço de rede 10*.

endereço reservado

Um sinônimo para endereço não roteável.

enfileiramento justo (fair queueing)

Uma técnica bem conhecida para a programação de tráfego que é usado em routers. Veja *WRR*.

entrega de melhor esforço (best-effort delivery)

Característica das tecnologias de rede que não oferece confiabilidade nos níveis de enlace. O IP funciona bem sobre hardware de distribuição de melhor esforço porque ele não considera que a rede subjacente fornece confiabilidade. O protocolo UDP fornece serviço de distribuição de melhor esforço para programas de aplicativos.

entrega não confiável

Característica de um mecanismo que não garante entregar dados sem

perda, dano, duplicação ou na mesma ordem em que foram enviados. Veja *entrega de melhor esforço*.

ESP (Encapsulating Security Payload)
Um formato de pacote usado pelo IPSec para enviar informações criptografadas.

estado de link (link-state)
Uma classe de protocolos de atualização de roteamento em que cada roteador participante difunde mensagens de estado e cada roteador usa o algoritmo SPF de Dijkstra para calcular as rotas mais curtas até cada destino. Veja *vetor de distância*.

Ethernet de par trançado (twisted pair Ethernet)
Esquema de cabeamento Ethernet que usa cabo de par trançado. Veja *GigE* e *cabo de categoria 5*.

Ethernet
Uma tecnologia de rede local popular originalmente inventada no Palo Alto Research Center da Xerox Corporation. A Ethernet é um sistema de entrega de melhor esforço que usa a tecnologia CSMA/CD.

EUI-64
Um padrão de endereçamento de 64 bits da camada 2 do IEEE.

Exterior Gateway Protocol
Veja *EGP*.

eXternal Data Representation
Veja *XDR*.

extinção de origem (source quench)
Uma mensagem ICMP enviada de volta para a origem quando datagramas excedem um roteador.

FDM (Frequency Division Multiplexing)

O método de passar múltiplos sinais independentes através de um único meio, atribuindo a cada um uma frequência de portadora única. O hardware para combinar sinais é chamado de multiplexador; o hardware para separá-los se chama demultiplexador. Veja também *TDM*.

FIB (Forwarding Information Base)

Os dados coletados por protocolos de roteamento e usados para construir uma tabela de encaminhamento.

filtro de pacote (packet filter)

Um mecanismo em um roteador, que pode ser configurado para rejeitar alguns tipos de pacotes e admitir outros. Os filtros de pacote são usados para criar um firewall de segurança.

fim a fim (end-to-end)

Característica de qualquer mecanismo que fornece comunicação da origem inicial até o destino final. Os protocolos como TCP são classificados como fim a fim.

FIN

Um tipo de segmento TCP. Para fechar uma conexão, cada lado precisa enviar um FIN.

Firewall

Um mecanismo colocado entre a intranet de uma organização e a Internet para fornecer segurança.

flat namespace

Característica de qualquer esquema de nomeação em que os nomes de objetos são selecionados a partir de um único conjunto de sequências (por exemplo, nomes de rua em uma típica cidade).

fluxo (flow)

Um termo usado para caracterizar uma sequência de pacotes enviada de uma origem a um destino. Algumas tecnologias de rede definem um fluxo separado para cada par de aplicações em comunicação, enquanto outras definem um único fluxo para incluir todos os pacotes entre um

par de hosts. Veja *5 tuplas*.

fragmentação

O processo de dividir um datagrama IP em partes menores. Um roteador fragmenta um datagrama quando a MTU da rede é menor do que o datagrama; o destino final remonta os fragmentos.

Frame Relay

O nome de uma antiga tecnologia de rede orientada à conexão oferecida pelas companhias telefônicas.

Frame

Um termo para um pacote de camada 2 derivado dos primeiros protocolos que adicionaram marcadores especiais do início e no fim do frame durante a transmissão de um pacote.

FTP (File Transfer Protocol)

Um protocolo de camada de aplicação TCP/IP que usa TCP para transferir arquivos de um computador para outro.

FTP anônimo (anonymous FTP)

Uma sessão de FTP que usa o nome de login anonymous para acessar arquivos públicos. Um servidor que permite FTP anônimo normalmente permite a senha guest.

full duplex

Característica de uma tecnologia que permite transferência de dados simultânea nos dois sentidos.

FYI (For Your Information)

Um subconjunto de RFCs que contém tutoriais ou informações gerais sobre tópicos relacionados ao TCP/IP ou à Internet conectada.

gated (gateway daemon)

Um programa executado em um roteador e que interconecta vários protocolos de roteamento, normalmente um IGP e um EGP.

gateway

Qualquer mecanismo que conecta dois ou mais sistemas heterogêneos e traduz entre eles. Originalmente, os pesquisadores usavam o termo gateway IP; os fornecedores adotaram o termo roteador IP.

Gateway IP

O termo originalmente usado para IP router (IP roteador).

Gbps (Gigabits por segundo)

Uma medida da velocidade de transmissão de dados igual a 10^9 bits por segundo. Veja também *Kbps* e *Mbps*.

gerenciamento de rede

Veja *MIB* e *SNMP*.

gerente de área (area manager)

Uma pessoa a cargo de uma área IETF. O conjunto de gerentes de área forma o IESG.

GigE (Gigabit Ethernet)

Uma tecnologia Ethernet que opera em 1 Gbps.

GRE (Generic Routing Encapsulation)

Um esquema para informações de encapsulamento no IP que inclui IP-em-IP como uma possibilidade.

grupo de trabalho

Um grupo de pessoas no IETF trabalhando em um determinado protocolo ou projeto.

grupo todos os roteadores (all routers group)

O bem conhecido grupo IP multicast que inclui todos os roteadores na rede local.

grupo todos os sistemas (all systems group)

O bem conhecido grupo IP multicast que inclui todos os hosts e

roteadores na rede local.

H.323
Uma recomendação da ITU para um conjunto de protocolos usados para telefonia IP.

half duple
Característica de uma tecnologia que só permite transmissão de dados em um sentido de cada vez. Compare com *full duplex*.

HELO
O comando na troca inicial do protocolo SMTP.

histórico
Uma classificação do IETF usada para desencorajar o uso de um protocolo. Basicamente, um protocolo declarado histórico é obsoleto.

horizonte dividido (split horizon)
Uma heurística usada pelos protocolos de vetor de distância, como o RIP, para evitar loops de encaminhamento.

host
O termo do TCP/IP para um sistema de computador de usuário final. Além dos sistemas de desktop, os hosts incluem dispositivos como impressoras, laptops e pequenos sistemas embutidos. Compare com *roteador IP*.

host multihome
Um host TCP/IP que possui conexões a duas ou mais redes físicas.

HTML (HyperText Markup Language)
O formato de documento padrão usado para páginas web.

HTTP (HyperText Transfer Protocol)
O protocolo usado para transferir documentos web de um servidor para um navegador.

HTTPS (HyperText Transfer Protocol with Security)
Uma versão segura do HTTP.

IAB (Internet Architecture Board)
Um pequeno grupo de pessoas que define política e orientação para o TCP/IP e a Internet em geral. O IAB era antigamente chamado de Internet Activities Board. Veja *IETF*.

IANA (Internet Assigned Numbers Authority)
O grupo responsável pela atribuição de valores usados em protocolos.

ICANN (Internet Corporation For Assigned Names and Numbers)
Um grupo responsável por definir política e coordenar registros que atribuem endereços IP.

ICMP (Internet Control Message Protocol)
Uma parte integrante do Internet Protocol (IP) que manipula mensagens de erro e controle (por exemplo, ping). Os roteadores e hosts usam ICMP para enviar relatórios de problemas sobre datagramas de volta à origem que enviou o datagrama.

ICMPv6 (Internet Control Message Protocol version 6)
A versão do ICMP que foi definida para uso com o IPv6.

IEEE (Institute of Electrical and Electronic Engineers)
Uma comunidade de profissionais que publica normas para muitas redes de área local, como Ethernet.

IESG (Internet Engineering Steering Group)
Um comitê formado pelo presidente do IETF e os gerentes de área, que convoca grupos de trabalho e aprova RFCs.

IETF (Internet Engineering Task Force)
Um grupo de pessoas subordinadas ao IAB, as quais trabalham no projeto e engenharia do TCP/IP e da Internet. O IETF é dividido em áreas, que, por sua vez, são subdivididas em grupos de trabalho.

IGMP (Internet Group Management Protocol)

Um protocolo que os hosts utilizam para manter os roteadores locais informados de sua associação em grupos multicast.

IGP (Interior Gateway Protocol)

O termo genérico aplicado a qualquer protocolo usado para propagar informações de acessibilidade de rede e roteamento dentro de um sistema autônomo.

IMAP (Internet Message Access Protocol)

Um protocolo usado para transferir mensagens de e-mail entre a caixa de correio de um usuário e um agente que o usuário executa para ler e-mail. Veja *SMTP*.

início lento (slow start)

Um esquema de prevenção de congestionamento no TCP em que o protocolo aumenta seu tamanho de janela à medida que ACKs chegam.

Interior Gateway Protocol

Veja *IGP*.

Internet

Uma coleção de redes de dados ao redor do mundo, que são interconectadas por roteadores e usam protocolo TCP/IP para atuar como uma única e gigantesca rede virtual.

Internet das coisas

Uma referência a uma mudança das aplicações iniciais da Internet, que proveu desde a comunicação humana a novos serviços que conectam sistemas integrados, tais como sensores.

Internet draft

A minuta do documento gerado pelo IETF; se aprovado, o documento se tornará um RFC.

Internet Protocol

Veja *IP*.

interoperabilidade

A capacidade que software e hardware em múltiplas máquinas de diversos fornecedores possuem de se comunicar de modo eficiente, seguro e expressivo.

intranet

Uma rede corporativa particular consistindo em hosts, roteadores e redes que usam tecnologia TCP/IP. A maioria das intranets corporativas está conectada à Internet.

intrServ (Integraded services/serviços integrados)

O esforço original para adicionar QoS.

IP (Internet Protocol)

O protocolo no grupo TCP/IP que define o serviço de distribuição de pacotes de melhor esforço e sem conexão, o qual forma a base para a Internet. O termo pode referir-se à versão 4 ou 6 (IPv4 ou IPv6).

IP móvel

Uma tecnologia desenvolvida pelo IETF para permitir que um computador viaje para um novo local enquanto mantém seu endereço IP original.

IP-em-IP

O encapsulamento de um datagrama IP dentro de outro, para transmissão através de um túnel. O IP-em-IP é usado com VPNs e IPv6.

IPng (Internet Protocol – the Next Generation)

Um termo mais antigo usado para IPv6.

IPsec (IP security)

Um padrão de segurança usado com IPv4 ou IPv6 para permitir que um emissor autentique ou criptografe um datagrama.

IPv4 (Internet Protocol version 4)

A referência à versão original do IP definido em 1978 e utilizado como base para a Internet global. IPv4 está sendo substituído pelo IPv6.

IPv6 (Internet Protocol version 6)

A referência à versão de IP definido na década de 1990 como um sucessor para o IPv4.

IPv6-ND (IPv6 Neighbor Discovery)

Um protocolo de apoio que um nó IPv6 utiliza para identificar os outros nós conectados à mesma rede e aprender seus endereços MAC. Veja *ARP*.

IRTF (Internet Research Task Force)

Um grupo relativamente inativo, paralelo ao IETF.

IS-IS (Intermediate System to Intermediate System)

Um protocolo de roteamento projetado para OSI, mas mais tarde modificado para uso com TCP/IP.

ISO (International Organization for Standardization)

Um corpo internacional que projeta, discute, propõe e especifica padrões para protocolos de rede. A ISO é mais conhecida por seu modelo de referência de sete camadas que descreve a organização conceitual dos protocolos.

isócrono

Característica de um sistema de rede que não introduz jitter. O sistema de telefonia convencional é isócrono.

ISP (Internet Service Provider)

Qualquer organização que venda acesso à Internet, seja conectividade permanente, seja acesso discado.

ITU (International Telecommunication Union)

Uma organização internacional que define padrões para interconexão de equipamento telefônico.

IXP (Internet eXchange Point)

Um local físico em que os ISPs se conectam. Veja *NAP* e *acordo de peering*.

janela (window)

Veja *janela deslizante*.

janela deslizante (sliding window)

Característica de um protocolo que permite a um emissor transmitir mais de um pacote de dados antes de receber um reconhecimento. O número de pacotes ou bytes pendentes é conhecido como tamanho de janela; aumentar o tamanho de janela aumenta a vazão.

janela fechada (closed window)

Uma situação no TCP em que um receptor enviou um anúncio de janela de tamanho zero porque nenhum espaço de buffer adicional estava disponível. Quando o espaço se torna disponível, o receptor abre a janela novamente.

janela zero

Veja *janela fechada*.

jitter

Um termo usado para descrever variação indesejada no atraso, causada quando um pacote em uma sequência precisa ser atrasado mais do que outro. O jitter pode ocorrer quando roteadores em uma rede de comutação de pacotes experimentam congestionamento.

Kbps (Kilobits por segundo)

Uma medida da taxa de transmissão de dados equivalente a 10^3 bits por segundo. Veja também *Gbps* e *Mbps*.

keepalive

Uma pequena mensagem enviada periodicamente entre duas entidades em comunicação para garantir que a conectividade de rede permaneça

intacta e que os dois lados ainda estejam respondendo.

LAN (Local Area Network – rede local)

Qualquer tecnologia de rede física projetada para abranger curtas distâncias (até alguns quilômetros). Veja *WAN*.

LATNIC

Veja *registro*.

limite de salto (hop limit)

O nome IPv6 para o campo cabeçalho de datagrama que o IPv4 chama de tempo de vida. O limite de salto evita que datagramas sigam um loop de encaminhamento infinitamente.

link-state

Uma classe de protocolos de roteamento de atualização em que cada participantes transmite mensagens de status do roteador e cada roteador usa o algoritmo SPF de Dijkstra para calcular rotas mais curtas para cada destino. Ver *vetor de distância*.

link-status

Um sinônimo de link-state.

LIS (Logical IP Subnet)

Um termo usado quando o IP é executado sobre ATM para descrever uma rede virtual que possui um único prefixo IP.

little endian

Um formato de armazenamento ou transmissão de dados binários em que o byte menos significativo (bit) vem primeiro. Veja *big endian*.

loop de encaminhamento (forwarding loop)

Uma condição de erro em que, em um ciclo de roteadores, cada um tem o próximo roteador no ciclo como o caminho mais curto até um determinado destino. Também conhecido como loop de roteamento.

loop de roteamento (routing loop)

Uma condição de erro em que, em um ciclo de roteadores, cada um tem o próximo roteador no ciclo como o caminho mais curto até um determinado destino. Também conhecido como loop de encaminhamento.

LPM (Longest-Prefix Match)

A técnica usada ao pesquisar uma tabela de roteamento, que escolhe uma entrada com o prefixo mais longo que corresponde ao endereço de destino.

LSP (Label Switched Path)

Um caminho através de uma série de LSRs que executam o MPLS.

LSR (Loose Source Route; também Label Switching Router)

Loose Source Route é uma opção IP que contém uma lista dos endereços de roteador que o datagrama precisa visitar na ordem. O datagrama também pode ser encaminhado para paradas intermediárias adicionais. Um Label Switching Router é um roteador que tem recursos para encaminhar tráfego MPLS. Normalmente, esse roteador também manipula encaminhamento de IP convencional e pode ser configurado para transferir pacotes entre o MPLS e o IP convencional.

MABR (Multicast Area Border Router)

O termo do MOSPF para um roteador multicast que troca informações de roteamento com outra área.

MAC (Media Access Control)

Uma referência geral aos protocolos de hardware de nível inferior usados para acessar uma determinada rede. O termo endereço MAC frequentemente é usado como um sinônimo de endereço físico.

mail exchanger

Um computador que aceita e-mail. O DNS possui um tipo de consulta separado para mail exchangers.

malha de rede (mesh network)

Um tipo de rede na qual um conjunto de nós estabelece ligações ponto a

ponto entre pares de nós e se compromete a encaminhar pacotes em nome de outros nós. Uma malha é normalmente usada para uma rede sem fio.

Management Information Base

Veja *MIB*.

máscara

Veja *máscara de endereço*.

máscara de endereço (address mask)

Um valor de 32 bits com os bits em 1 indicando o prefixo de rede e os bits em 0 indicando o sufixo do host. No IPv4, uma máscara de endereço é de 32 bits; em IPv6, ela é de 128 bits de tamanho.

máscara de sub-rede

Veja *máscara de endereço*.

MBONE (Multicast BackBONE)

Um acordo de cooperação entre sites para encaminhar datagramas multicast através da Internet pelo uso do tunelamento IP.

Mbps (Milhões de bits por segundo)

Uma medida da taxa de transmissão de dados equivalente a 10^6 bits por segundo. Veja também *Gbps* e *Kbps*.

mDNS (multicast DNS)

Um mecanismo que usa multicast IP para permitir que um host encontre o endereço IP de outro host ou os endereços de hosts que oferecem um serviço.

mesh-under

Uma abordagem para a construção de redes de malha que utiliza protocolos de camada 2 para transmitir pacotes. Veja *route-over*.

MIB (Management Information Base)

O conjunto de variáveis (banco de dados) que um sistema executando um agente SNMP mantém.

migração para o IPv6

Uma referência para a mudança a partir de uma Internet IPv4 para uma Internet IPv6. Várias estratégias têm sido propostas, incluindo o uso de pilhas duplas (cada computador funciona tanto com IPv4 quanto com IPv6) e tradução (um dispositivo na fronteira entre uma de rede IPv4 e uma rede IPv6 envia a carga útil – payload – de um tipo de datagra-ma em um datagrama de outro tipo).

MIME (Multipurpose Internet Mail Extensions)

Um padrão usado para codificar dados, tais como imagens, como texto ASCII imprimível para transmissão por e-mail.

MLS (Multi-Layer Switching)

Uma tecnologia de comutação IP oferecida pela Cisco Systems.

modelagem de tráfego

Garantia de que o tráfego saindo está em conformidade com uma velocidade especificada.

modelo de referência

Uma descrição de como os protocolos em camadas se encaixam. O TCP/IP usa um modelo de referência de cinco camadas; anteriormente, os protocolos usavam o modelo de referência de sete camadas da ISO.

modelo de referência de cinco camadas

Um protocolo de camadas modelo usado por TCP/IP. Veja o *modelo de referência de sete camadas.*

modelo de referência de sete camadas

Um modelo mais antigo de protocolo em camadas criado pela ISO. Veja *modelo de referência de cinco camadas.*

modo promíscuo (promiscuous mode)
Um recurso do hardware de interface de rede que permite a um computador receber todos os pacotes na rede.

MOSPF (Multicast Open Shortest Path First)
Extensões de multicast ao protocolo de roteamento OSPF.

MPLS (Multi-Protocol Label Switching)
Uma tecnologia que usa hardware de comutação de alta velocidade para transportar datagramas IP. O MPLS é derivado da comutação IP.

mrouted (multicast route daemon)
Um programa usado com uma pilha de protocolo que aceita multicast IP para estabelecer encaminhamento multicast.

MSL (Maximum Segment Lifetime)
O tempo mais longo que um datagrama pode sobreviver na Internet. Os protocolos usam o MSL para garantir um limite de tempo em que os pacotes duplicados possam sobreviver.

MSS (Maximum Segment Size)
Um valor negociado ao usar TCP, que especifica a maior quantidade de dados que podem ser transmitidos em um segmento.

MTU (Maximum Transmission Unit)
A maior quantidade de dados que podem ser transferidos através de uma determinada rede física em um único pacote.

MTU do caminho (path MTU)
A MTU mínima ao longo de um caminho da origem ao destino (ou seja, o tamanho do maior datagrama que pode ser enviado pelo caminho sem fragmentação).

multicast
Uma técnica que permite que cópias de um único pacote sejam transmitidas a um subconjunto selecionado de todos os destinos possíveis. Alguns hardwares (como Ethernet) aceitam multicast. Veja

broadcast e *unicast*.

multicast IP

Um esquema de endereçamento e encaminhamento que permite a transmissão de datagramas IP para um subconjunto de hosts. A Internet atualmente não possui recursos extensivos para manipular multicast IP.

multiplexar

Combinar dados de várias origens em um único fluxo, de maneira que eles possam ser separados novamente mais tarde. Veja *demultiplexar*.

NACK (Negative ACKnowledgment)

A mensagem voltou a negar um pedido.

NAP (Network Access Point)

Um local físico no qual os ISPs interconectam redes. Veja também *IXP*.

NAPT (Network Address and Port Translation)

A forma mais popular da NAT, em que números de porta de protocolo e endereços IP são usados na tradução.

NAT (Network Address Translation)

Uma tecnologia que permite que hosts usando endereços privativos se comuniquem com um destino arbitrário na Internet.

NFS (Network File System)

Um protocolo que usa UDP para permitir que um conjunto de computadores cooperados acesse os sistemas de arquivo uns dos outros.

NIC (Network Interface Card)

Um dispositivo de hardware que é encaixado no barramento em um computador e conecta o computador a uma rede.

NOC (Network Operations Center)

Originalmente, a organização na BBN que monitorava e controlava várias redes que formavam parte da Internet. Hoje, usado para qualquer

organização que gerencia uma rede.

nome de domínio (domain name)
Um nome consistindo em uma sequência de rótulos separados por pontos. Um nome de domínio único é atribuído a cada computador na Internet.

notação decimal com ponto (dotted decimal notation)
Uma forma sintática usada para representar inteiros binários de 32 bits. Quatro valores de 8 bits são escritos na base 10 com pontos separando-os.

notação hexadecimal com ponto (dotted hex notation)
Uma forma sintática usada para representar valores binários que consistem em valores hexadecimais para cada quantidade de 8 bits com pontos separando-os.

notação quádrupla com ponto (dotted quad notation)
Uma forma sintática usada para representar valores binários que consistem em valores hexadecimais para cada quantidade de 16 bits com pontos separando-os.

NVT (Network Virtual Terminal)
O protocolo orientado a caractere usado pelo TELNET.

octeto
Uma unidade de 8 bits de dados. Embora os engenheiros frequentemente usem o termo byte como sinônimo de octeto, um byte pode ser menor ou maior do que 8 bits.

off link
Dois nós IPv6 que partilham um prefixo, mas não estão diretamente conectados. Na maioria das vezes, o termo refere-se a uma rede de malha. Veja *on link*.

on link
Dois nós IPv6 que partilham um prefixo e estão diretamente conectados.

Veja *off link*.

OpenFlow

Um protocolo usado para implementar Software Defined Networking. Veja *SDN*.

ordem de byte de rede (network byte order)

O padrão TCP/IP para transmissão de inteiros que especifica que o byte mais significativo aparece primeiro (big endian).

ordem de byte padrão (standard byte order)

Veja *ordem de byte de rede*.

Organização Internacional para Padronização (International Organization for Standardization)

Veja *ISO*.

OSPF (Open Shortest Path First)

Um protocolo de roteamento de estado de enlace projetado pelo IETF.

OUI (Organizationally Unique Identifier)

Parte de um endereço atribuído a uma organização que fabrica hardware de rede; a organização atribui um endereço único a cada dispositivo usando seu OUI mais um número de sufixo.

P2P (Peer to Peer)

As aplicações de modelo usado para propagar informações de forma rápida, em que cada cliente se compromete a agir como um servidor e disponibilizar informações para outros clientes.

pacote

Uma pequena unidade de dados enviada através de uma rede de comutação de pacotes.

padrões da série OC

Uma série de padrões para a transmissão de dados por fibra ótica. Por

exemplo, o OC-48 possui uma velocidade de bits de aproximadamente 2,5 gigabits por segundo.

PAR (Positive Acknowledgement with Retransmission)

Os protocolos de modelo usados para realizar o transporte de dados confiável.

PCM (Pulse Code Modulation)

Um padrão para codificação de voz usado na telefonia digital, que produz 8 mil amostras de 8 bits por segundo.

PDU (Protocol Data Unit, também Packet Data Unit)

Um termo ISO para pacote que foi adotado para uso com mensagens SNMP.

PIM-DM (Protocol Independent Multicast Dense Mode)

Um protocolo de roteamento multicast semelhante ao DVMRP.

PIM-SM (Protocol Independent Multicast Sparse Mode)

Um protocolo de roteamento multicast que usa um método semelhante ao CBT.

PING (Packet InterNet Groper)

O nome de um programa usado em Redes TCP/IP para testar a acessibilidade de destinos, enviando-lhes uma requisição de eco ICMP e esperando uma resposta. O termo agora é usado como um verbo (por exemplo, "por favor, pingue o host *A* para ver se ele está ativo").

PoE (Power over Ethernet)

Uma tecnologia usada com telefones IP para fornecer energia através do mesmo cabo usado para uma conexão Ethernet.

policiamento de tráfego

Verificação para garantir que o tráfego chegando adere às restrições preestabelecidas.

ponte (bridge)

Um dispositivo que conecta duas ou mais redes e encaminha quadros entre elas. As pontes diferem dos roteadores porque elas usam endereços físicos, enquanto os roteadores usam endereços IP.

ponto de reprodução (playback point)

A quantidade de dados mínima necessária em um buffer de jitter antes que a reprodução comece.

POP (Post Office Protocol)

Um protocolo usado para acessar e extrair e-mail de uma caixa de correio. A versão atual é POP3, e também existe uma versão segura chamada POP3S.

porta

Veja *porta de protocolo*.

porta bem conhecida

Um número de porta de protocolo usado pelo TCP ou UDP, que é pré-atribuído a um serviço específico (por exemplo, e-mail).

porta de protocolo

A abstração usada pelos protocolos de transporte TCP/IP para distinguir entre múltiplos destinos dentro de um determinado host. TCP e UDP usam inteiros pequenos como números de porta.

POS (Packet Over SONET)

O método utilizado para enviar pacotes através de uma rede SONET.

POTS (Plain Old Telephone Service)

Uma referência ao sistema de telefonia de voz convencional.

PPP (Point to Point Protocol)

Um protocolo para tratar quadros ao enviar datagramas IP através de uma linha serial.

PPPoE

Uma versão do Point-to-Point Protocol (protocolo ponto a ponto) usada

através de Ethernet.

prefixo do provedor
Um esquema de endereçamento em que um ISP possui um prefixo de rede e atribui a cada consumidor endereços que começam com o prefixo.

problema de salto extra
Uma condição de erro em que a comunicação é possível, mas cada datagrama leva uma viagem extra desnecessária através de uma rede.

protocolo
Uma descrição formal dos formatos de mensagem e as regras que duas ou mais máquinas precisam seguir para trocar essas mensagens.

proxy
Qualquer dispositivo ou sistema que age no lugar de outro (por exemplo, um servidor web proxy age no lugar de outro servidor web).

proxy ARP
A técnica em que uma máquina, normalmente um roteador, responde a requisições ARP destinadas a outro, fornecendo seu próprio endereço físico. O proxy ARP permite que várias redes compartilhem um único prefixo IP.

pseudocabeçalho (pseudo-header)
Informações adicionais usadas em um cálculo de checksum, mas não presentes no datagrama UDP ou no segmento TCP. Um pseudocabeçalho inclui endereços IP de origem e destino.

PSTN (Public Switched Telephone Network)
O sistema de telefonia e voz convencional.

push
A operação que uma aplicação realiza em uma conexão TCP para forçar os dados a serem enviados imediatamente. Um bit no cabeçalho de segmento marca os dados enviados nessa operação.

PVC (Permanent Virtual Circuit)

O tipo de circuito virtual ATM estabelecido por um administrador em vez de um software em um computador. Veja *SVC*.

QoS (Quality of Service)

Limites sobre a perda, retardo, jitter e vazão mínima que uma rede garante distribuir.

RARP (Reverse ARP)

Um protocolo antigo usado para obter um endereço IPv4; agora está sendo usado em centros de dados para informar sobre um interruptor de migração de máquina virtual.

RDP (Remote Desktop Protocol)

Um protocolo definido pela Microsoft, que fornece uma capacidade de desktop remoto a seus sistemas operacionais.

reconhecimento (acknowledgement)

Uma resposta enviada por um receptor para indicar uma recepção bem-sucedida. Normalmente, um reconhecimento é enviado quando um pacote chega.

reconhecimento adiado (delayed acknowledgement)

Uma heurística empregada por um TCP receptor para evitar a síndrome da janela tola.

reconhecimento cumulativo

O esquema de reconhecimento usado pelo TCP em que um sack informa o ponto em um fluxo no qual os dados foram recebidos com sucesso. Veja *SACK*.

reconhecimento positivo (positive acknowledgement)

Sinônimo de reconhecimento.

reconhecimento seletivo

Veja *SACK*.

reconstituição (reassembly)

O processo de coletar todos os fragmentos de um datagrama IP e usá-los para criar uma cópia do datagrama original. O destino final realiza a reconstituição.

RED (Random Early Detection)

Uma técnica que os roteadores usam no lugar do tail-drop para gerenciar buffers quando a memória começa a encher. Veja *tail-drop*.

rede anônima (anonymous network)

Um sinônimo para rede não numerada.

rede CL

Veja *serviço sem conexão*.

rede de longa distância

Termo mais antigo para WAN (rede remota).

rede não numerada

Uma técnica para conservar prefixos de rede IP, em que um prefixo não é atribuído a uma conexão não hierárquica entre dois roteadores.

rede ponto a ponto

Qualquer tecnologia de rede que conecta exatamente duas máquinas (por exemplo, um circuito alugado ou conexão discada).

redirect

Uma mensagem ICMP enviada de um roteador a um host em uma rede local para instruir o host a mudar de rota.

redução multiplicativa (multiplicative decrease)

Uma técnica usada pelo TCP para reduzir a transmissão quando ocorre congestionamento. O tamanho da janela efetiva é reduzido pela metade toda vez que um segmento é perdido.

registro

Uma organização autorizada a atribuir blocos de endereços IP. Os registros incluem: AFRINIC, APNIC, ARIN, LATNIC e RIPE.

registro MX (registro Mail eXchanger)
Um tipo de registro de DNS usado para armazenar o endereço de um servidor de e-mail para um determinado domínio.

reprodução (replay)
Uma situação de erro em que os pacotes de uma sessão anterior são erroneamente aceitos como parte de uma sessão posterior.

requisição e resposta de eco (echo request and reply)
Um tipo de mensagem que é usada para testar conectividade de rede. O programa ping usa requisição de eco e mensagens de resposta ICMP.

requisitos de host
Dois documentos RFC que especificam revisões e atualizações de muitos protocolos TCP/IP. Veja *requisitos de roteador*.

requisitos de roteador (router requirements)
Um documento que contém atualizações aos protocolos TCP/IP usados nos roteadores. Veja *requisitos de host*.

reset (TCP)
Um segmento enviado pelo TCP em resposta a uma condição de erro, como um segmento malformado que entra ou um segmento que entra para o qual o receptor não possui um registro ou uma conexão.

resolução de endereço (address resolution)
Conversão de um endereço de protocolo em um endereço físico correspondente (por exemplo, conversão de um endereço IP em um endereço Ethernet). Veja *ARP* e *IPv6-ND*.

resolução de nome
O processo de mapear um nome para um endereço correspondente. O sistema de nome de domínio fornece um mecanismo para nomear

computadores em que os programas usam servidores de nome remotos para resolver (converter) um nome de máquina em um endereço IP.

resolução

Veja *resolução de endereço*.

retransmissão adaptável (adaptive retransmission)

O esquema usado pelo TCP para fazer o timer de retransmissão controlar mudanças no tempo de viagem de ida e volta.

reverso envenenado (poison reverse)

Uma heurística usada por protocolos de vetor de distância, como RIP, para evitar loops de encaminhamento.

RFC (Request For Comments)

O nome de uma série de documentos que contêm pesquisas, medidas, ideias, técnicas e observações, bem como padrões de protocolo TCP/IP propostos e aceitos. As RFCs estão disponíveis *on-line*.

RIP (Routing Information Protocol)

Um protocolo usado para propagar informações de roteamento dentro de um sistema autônomo.

RIPE

Veja *registro*.

rota

O caminho que o tráfego de rede toma da origem ao destino.

rota de origem (source route)

Uma rota que é determinada pela origem e especificada como uma lista dos roteadores intermediários específicos que o pacote deve visitar. Veja *LSR* e *SSR*.

rota específica do host (host-specific route)

Uma entrada em uma tabela de roteamento que se refere a um único host, em oposição a rotas que se referem a uma rede, uma sub-rede IP ou a um padrão.

Rota-padrão (default route)

Uma única entrada em uma lista de roteadores, que cobre todos os destinos que não estão incluídos explicitamente. As tabelas de roteamento na maioria dos roteadores e hosts contêm uma entrada para uma rota-padrão.

roteador

Veja *roteador IP*.

roteador IP

Um sistema de rede de finalidade especial que conecta duas ou mais redes (possivelmente heterogêneas) e passa tráfego IP entre elas.

roteador one-armed

Um roteador IP que entende dois domínios de endereçamento, mas possui apenas uma conexão de rede física. Também chamado de firewall one-armed.

roteamento de sistema interautônomo (inter-autonomous system routing)

Um sinônimo para roteamento exterior. Veja *BGP*.

roteamento entre domínios (inter-domain routing)

Um sinônimo de roteamento exterior. Veja *BGP* e *CIDR*.

routed (route daemon)

Um programa UNIX que implementa o protocolo RIP.

route-over

Uma abordagem para a construção de redes de malha que utiliza protocolos da camada 3 para transmitir pacotes. Veja *submalha*.

RPC (Remote Procedure Call – chamada de procedimento remoto)

Uma tecnologia em que um programa chama serviços através de uma rede fazendo chamadas de procedimento modificado.

RPF (Reverse Path Forwarding)

Uma tecnologia usada para propagar pacotes de broadcast, que garante que não há qualquer loop de encaminhamento. O IP usa o RPF para propagar datagramas de broadcast e multicast de sub-rede.

RPM (Reverse Path Multicast)

Um método geral para multicasting que usa o algoritmo TRPB.

RST (ReSeT)

Uma abreviação para um segmento reset TCP.

RSVP (Resource ReserVation Protocol)

O protocolo que permite que uma extremidade requisite um fluxo com QoS específica; os roteadores ao longo do caminho até o destino precisam concordar antes de aprovarem a requisição.

RTCP (RTP Control Protocol)

O protocolo associado ao RTP usado para controlar uma sessão.

RTO (Round-trip Time-Out)

O tempo que um protocolo espera antes de transmitir um pacote. Veja *RTT*.

RTP (Real-time Transport Protocol)

O principal protocolo usado para transferir dados em tempo real, como voz e vídeo, através da Internet.

RTT (Round Trip Time)

O tempo total percorrido para um único pacote ou datagrama deixar um computador, alcançar outro e voltar.

SACK (Selective ACKnowledgement)

Uma adição opcional ao mecanismo de reconhecimento TCP, que permite que um receptor especifique quais segmentos foram recebidos.

Compare com *reconhecimento cumulativo*.

salto a salto (hop-by-hop)

A referência ao tratamento que é realizado em cada roteador (em oposição ao fim a fim). IPv6 classifica alguns cabeçalhos de extensão como salto a salto.

scp (secure copy program)

Um programa que funciona como o comando copy do UNIX, *cp*, exceto que o *scp* pode transferir arquivos de um computador para outro e usa criptografia para manter a transferência segura.

SDN (Software Defined Networking)

A separação do plano de controle do plano de dados de um dispositivo de rede que permite a um controlador externo que especifique o encaminhamento.

SDR (Software Defined Radio)

Uma abordagem análoga à SDN que permite ao software que defina características de rádio, tais como frequência e modulação.

segmento

A unidade de transferência usada pelo TCP.

segurança do perímetro

Um mecanismo de segurança de rede que coloca um firewall em cada conexão entre uma posição e redes externas.

sequências de tamanho fixo em sub-redes (fixed-length subnetting)

Um esquema de atribuição de endereços de sub-rede em que todas as redes físicas de uma organização usam a mesma máscara. A alternativa é a sub-rede de comprimento variável.

serviço orientado à conexão (connection-oriented service)

Característica do serviço oferecido por qualquer tecnologia de rede em que as partes comunicando-se primeiramente estabelecem comunicação

e, depois, enviam dados. O TCP fornece serviço orientado à conexão assim como o hardware de ATM.

serviço provido

Um serviço que é configurado manualmente.

serviço sem conexão (connectionless service)

Característica do serviço oferecido por qualquer tecnologia de rede em que cada mensagem ou pacote identifica o destino e é manipulada independentemente. Tanto o IP quanto o UDP fornecem serviço sem conexão.

servidor

Um aplicativo que fornece serviço a clientes através de uma rede.

servidor de arquivos (file server)

Um aplicativo que fornece acesso a arquivos aos clientes remotos. O termo também é usado informalmente para o computador que executa a aplicação servidora de arquivos.

sinalização (signaling)

Um termo da telefonia que se refere aos protocolos que estabelecem um circuito.

síndrome da janela tola (silly window syndrome)

Uma condição que pode surgir no TCP, em que o receptor anuncia repetidamente uma pequena janela e o emissor envia repetidamente um pequeno segmento para preenchê-la.

SIP (Session Initiation Protocol)

Um protocolo desenvolvido pelo IETF para sinalização na telefonia IP.

sistema autônomo (autonomous system)

Veja *AS*.

sistema de nomes de domínios (Domain Name System)

Veja *DNS*.

SMTP (Simple Mail Transfer Protocol)

O protocolo padrão TCP/IP para transferir mensagens de e-mail de uma máquina para outra.

SNAP (SubNetwork Attachment Point)

Um padrão do IEEE para um pequeno cabeçalho colocado na área de dados de um quadro para especificar o tipo de dados.

SNMP (Simple Network Management Protocol)

Um protocolo usado para gerenciar dispositivos como hosts, roteadores e impressoras. Veja também *MIB*.

SOA (Start Of Authority)

Uma palavra-chave usada com o DNS para indicar o início dos registros sobre os quais um determinado servidor é a autoridade.

soft state

Uma técnica em que um receptor expira o tempo das informações em vez de depender do emissor para mantê-las.

Software Defined Networking

Veja *SDN*.

Software Defined Radio

Veja *SDR*.

SPF (Shortest Path First)

Uma classe de protocolos de atualização de roteamento que usa o algoritmo de Dijkstra para calcular os caminhos mais curtos. Veja *estado de link*.

SPI (Security Parameters Index)

O identificador que o IPSec usa para especificar a associação de

segurança que deve ser usada a fim de processar um datagrama.

SS7 (Signaling System 7)
O padrão do sistema de telefonia convencional usado para sinalização.

ssh (shell seguro)
Uma alternativa ao TELNET que usa criptografia para manter as sessões confidenciais.

SSL (Secure Sockets Layer)
Um padrão efetivo para comunicação segura, originalmente criado pela Netscape, Inc.

SSR (Strict Source Route)
Uma opção do IP que contém uma lista dos endereços de roteador que o datagrama precisa visitar em ordem. Veja *LSR*.

STD (STanDard)
A designação usada para classificar uma determinada RFC como descrevendo um protocolo padrão.

SubNetwork Attachment Point
Veja *SNAP*.

sub-rede de tamanho variável
Um esquema de atribuição de endereço de sub-rede em que cada rede física em uma organização pode ter uma máscara diferente. Veja *sub-rede de tamanho fixo*.

sub-rede lógica
Veja *LIS*.

SVC (Switched Virtual Circuit)
O tipo de circuito virtual ATM estabelecido quando necessário pelo software, em vez de por um administrador.

SWS

Veja *síndrome da janela tola*.

SYN (SYNchronizing segment – segmento de sincronização)

Primeiro segmento enviado pelo protocolo TCP, o SYN é usado para sincronizar os dois lados de uma conexão em preparação para abrir uma conexão.

tail drop (queda da cauda)

Uma política que os roteadores usam para gerenciar buffers descartando todos os pacotes que chegam após a memória estar cheia. Veja *RED*.

TCP (Transmission Control Protocol)

O protocolo de camada de transporte padrão que fornece o serviço de fluxo full duplex seguro de que muitos protocolos de aplicação dependem.

TCP/IP (Conjunto de Protocolos da Internet – Internet Protocol Suite)

O nome oficial dos protocolos que compõem TCP/IP.

TDM (Time Division Multiplexing)

Uma técnica usada para multiplexar vários sinais em um único canal de transmissão de hardware, permitindo que cada sinal use o canal por um curto período de tempo antes de ir para próximo. Veja *FDM*.

telefonia IP

Um sistema telefônico que usa IP para transportar voz digitalizada. Veja *VoIP*.

TELNET

O protocolo padrão do TCP/IP para acesso de terminal remoto.

tempo de espera (hold down)

Um período de tempo fixo e curto, após uma alteração para uma tabela de roteamento durante o qual nenhuma outra alteração é aceita. Hold down ajuda a evitar loops de encaminhamento.

tempo de viagem ida e volta (round trip time)

Veja *RTT*.

TFTP (Trivial File Transfer Protocol)

Um protocolo padrão TCP/IP para transferência de arquivos com capacidade e over-head mínimos.

tipo de roteamento de serviço

Um esquema em que a escolha do caminho depende das características da rede necessárias pelo fluxo, bem como o caminho mais curto até o destino.

TOS (Type Of Service)

Uma referência à interpretação original do campo em um cabeçalho IPv4, que permite que o emissor especifique o tipo de serviço desejado. Agora substituído pelo DiffServ.

traceroute

Um programa que usa valores de TTL cada vez maiores e mensagens de erro ICMP para determinar saltos intermediários ao longo do caminho até um destino.

transferência segura (reliable transfer)

Característica de um mecanismo que garante distribuir dados sem perda, sem dano, sem duplicação e na mesma ordem em que foram enviados, ou informar o emissor que a entrega é impossível.

TRPB (Truncated Reverse Path Broadcast)

Uma técnica usada no multicasting baseada em dados para encaminhar datagramas de multicast.

TRPF (Truncated Reverse Path Forwarding)

Um sinônimo para *TRPB*.

TTL (Time To Live)

Um valor no cabeçalho IP usado para garantir que um datagrama não permaneça na Internet para sempre, mesmo se as rotas forem

incorretamente configuradas.

tunelamento (tunneling)
Uma técnica em que um pacote é encapsulado em um protocolo de alto nível para transmissão. O tunelamento é usado para VPNs. Veja *IP-em-IP*.

UDP (User Datagram Protocol)
Um protocolo de transporte que permite a um aplicativo em uma máquina enviar um datagrama para um aplicativo em outra máquina.

União Internacional de Telecomunicações (International Telecommunications Union)
Veja *ITU*.

unicast
Um endereço que especifica encaminhamento de um pacote para um único destino. Veja *broadcast* e *multicast*.

unidade de transferência máxima
Veja *MTU*.

URL (Uniform Resource Locator)
Uma forma sintática que especifica um item e um protocolo usado para acessar o item.

VC (Virtual Circuit)
Um caminho, através de uma rede, de uma aplicação até outra que é usada para enviar dados.

vetor de distância (distance-vector)
Uma classe de protocolos de atualização de roteamento em que cada roteador participante envia a seus vizinhos uma lista das redes que ele pode alcançar e a distância até cada rede. Veja *estado de link*.

vetor-distância
Agora chamado de distância-vetor.

vínculo de endereço (address binding)
Sinônimo de resolução de endereço.

VLAN (Virtual Local Area Network)
Um conjunto de portas de um switch Ethernet que é configurado pelo gestor da rede para fazer parte de um único domínio de broadcast.

VLSM (Variable Length Subnet Mask)
A máscara usada com sub-redes de tamanho variável.

VoIP (Voice over IP)
Tecnologia que permite que ligações telefônicas de voz sejam transmitidas através da Internet. Veja *telefonia IP*.

VPN (Virtual Private Network)
Uma tecnologia que conecta, através da Internet, dois ou mais locais separados, mas permite-lhes funcionar como se fossem uma única rede privativa.

WAN (Wide Area Network)
Qualquer tecnologia de rede física que abrange grandes distâncias geográficas. Veja LAN e *rede de longa distância*.

Wi-Fi
Um consórcio que verifica a interoperação de produtos de rede sem fio 802.11, e um termo de marketing que os fornecedores usam para vender seus produtos.

Wi-MAX
Termo que os vendedores marketing usam para vender IEEE 802.16, produtos de rede sem fio.

Windows Sockets Interface

Uma variante da API de soquete, desenvolvida pela Microsoft, que também é chamada WINSOCK.

World Wide Web

O grande serviço de hipermídia disponível na Internet, que permite a um usuário pesquisar informações.

WRR (Weighted Round-Robin)

Um mecanismo de programação de pacote usado para aceitar pacotes de múltiplas filas e enviá-los através de um único canal de transmissão. O WRR permite que cada fila receba uma porcentagem garantida da largura de banda do canal.

WWW

Veja *World Wide Web*.

XDR (eXternal Data Representation)

O padrão para uma representação de dados independente da máquina. Para usar a XDR, um emissor traduz da representação da máquina local para a representação externa padrão e um receptor traduz da representação externa para a representação da máquina local.

X-Window System

Um sistema de software desenvolvido no MIT para apresentar e gerenciar saída em vídeos de bitmap.

ZigBee

Uma aliança de fornecedores que está padronizando os protocolos para o uso com uma rede de malha sem fio de sensores.

zona de autoridade (authority zone)

Uma parte da hierarquia de nome de domínio em que um único servidor de nome é a autoridade.

zona livre de default (default free zone)

O núcleo da Internet em que é proibido o roteamento padrão.