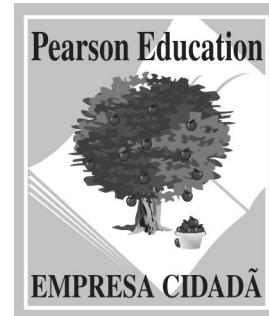


REDES DE COMPUTADORES E A INTERNET

5^a edição

Uma Abordagem Top-Down







KUROSE • ROSS

REDES DE COMPUTADORES E A INTERNET

5^a edição

Uma Abordagem Top-Down

Tradução

Opportunity Translations

Revisão Técnica

Wagner Luiz Zucchi

Professor Doutor do Departamento de Sistemas Eletrônicos
da Escola Politécnica da Universidade de São Paulo



São Paulo

Brasil Argentina Colômbia Costa Rica Chile Espanha Guatelamala México Peru Porto Rico Venezuela





© 2010 Pearson Education do Brasil

Título original: *Computer networking: a top-down approach featuring the Internet, fifth edition*

© 2010 Pearson Education, Inc.

Tradução autorizada a partir da edição original em inglês, publicada pela Pearson Education, Inc., sob o selo Addison Wesley.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Pearson Education do Brasil.

Diretor editorial: Roger Trimer

Gerente de editorial: Sabrina Cairo

Supervisor de produção editorial: Marcelo Françozo

Editoras: Gabriela Trevisan e Thelma Babaoka

Revisão: Maria Aiko Nishijima

Capa: Casa de Idéias, sobre projeto original de Joyce Cosentino Wells

Imagem de capa: © Reuters/Jean-Phillipe Arbs/Londres

Editoração eletrônica: Casa de Idéias

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Kurose, James F.

Redes de computadores e a Internet : uma abordagem top-down / James F. Kurose e Keith W. Ross ; tradução Opportunity translations ; revisão técnica Wagner Zucchi. -- 5. ed. -- São Paulo : Addison Wesley, 2010.

Título original: Computer networking fifth edition a top-down approach featuring the Internet.

Bibliografia.

ISBN 978-85-88639-97-3

1. Internet 2. Redes de computadores I. Ross, Keith W. II. Título.

09-10003

CDD-004.67

Índices para catálogo sistemático:

1. Internet : Redes de computadores : Processamento de dados 004.67

2009

Direitos exclusivos para a língua portuguesa cedidos à Pearson Education do Brasil, uma empresa do grupo Pearson Education

Av. Ermano Marchetti, 1435

CEP: 05038-001, São Paulo — SP

Fone: (11) 2178-8686 — Fax: (11) 2178-8688

e-mail: vendas@pearsoned.com



Para Julie e nossas três preciosidades:

Chris, Charlie e Nina

JFK

Para minha maravilhosa esposa Véronique

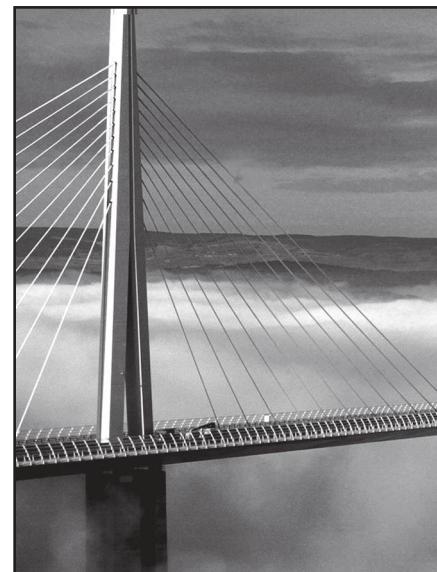
e nossas três filhas: Cécile, Claire e Katie

KWR





Sumário



Capítulo 1 Redes de computadores e a Internet	1
1.1 O que é a Internet?.....	2
1.1.1 Uma descrição dos componentes da rede.....	2
1.1.2 Uma descrição do serviço	4
1.1.3 O que é um protocolo?	5
1.2 A periferia da Internet	7
1.2.1 Programas clientes e servidores	7
1.2.2 Redes de acesso	8
1.2.3 Meios físicos.....	16
1.3 O núcleo da rede	18
1.3.1 Comutação de circuitos e comutação de pacotes	18
1.3.2 Como os pacotes percorrem as redes de comutadores de pacotes?	24
1.3.3 ISPs e backbones da Internet	25
1.4 Atraso, perda e vazão em redes de comutação de pacotes.....	26
1.4.1 Uma visão geral de atraso em redes de comutação de pacotes.....	27
1.4.2 Atraso de fila e perda de pacote.....	29
1.4.3 Atraso fim a fim	31
1.4.4 Vazão nas redes de computadores	33
1.5 Camadas de protocolo e seus modelos de serviço	35
1.5.1 Arquitetura de camadas	35
1.5.2 Mensagens, segmentos, datagramas e quadros	40
1.6 Redes sob ameaça	41
1.7 História das redes de computadores e da Internet.....	45
1.7.1 Desenvolvimento da comutação de pacotes: 1961-1972	45
1.7.2 Redes proprietárias e trabalho em rede: 1972-1980	46
1.7.3 Proliferação de redes: 1980-1990.....	47

1.7.4	A explosão da Internet: a década de 1990	48
1.7.5	Desenvolvimentos recentes	49
1.8	Resumo	50
Capítulo 2 Camada de aplicação.....		61
2.1	Princípios de aplicações de rede	61
2.1.1	Arquiteturas de aplicação de rede	62
2.1.2	Comunicação entre processos	64
2.1.3	Serviços de transporte disponíveis para aplicações	66
2.1.4	Serviços de transporte providos pela Internet	68
2.1.5	Protocolos de camada de aplicação	71
2.1.6	Aplicações de rede abordadas neste livro	72
2.2	A Web e o HTTP	72
2.2.1	Descrição geral do HTTP	72
2.2.2	Conexões persistentes e não persistentes	74
2.2.3	Formato da mensagem HTTP	76
2.2.4	Interação usuário-servidor: cookies	80
2.2.5	Caches Web	81
2.2.6	GET condicional	84
2.3	Transferência de arquivo: FTP	85
2.3.1	Comandos e respostas FTP	87
2.4	Correio eletrônico na Internet	87
2.4.1	SMTP	89
2.4.2	Comparação com o HTTP	91
2.4.3	Formatos de mensagem de correio e MIME	92
2.4.4	Protocolos de acesso ao correio	92
2.5	DNS: o serviço de diretório da Internet	96
2.5.1	Serviços fornecidos pelo DNS	96
2.5.2	Visão geral do modo de funcionamento do DNS	98
2.5.3	Registros e mensagens DNS	102
2.6	Aplicações P2P	106
2.6.1	Distribuição de arquivos P2P	107
2.6.2	Distributed Hash Tables (DHTs)	111
2.6.3	Estudo de caso: telefonia por Internet P2P com Skype	115
2.7	Programação e desenvolvimento de aplicações com TCP	116
2.7.1	Programação de aplicações com TCP	117
2.7.2	Um exemplo de aplicação cliente–servidor em Java	118
2.8	Programação de sockets com UDP	123
2.9	Resumo	129
Capítulo 3 Camada de transporte		140
3.1	Introdução e serviços de camada de transporte	140
3.1.1	Relação entre as camadas de transporte e de rede	142
3.1.2	Visão geral da camada de transporte na Internet	143

3.2	Multiplexação e demultiplexação.....	144
3.3	Transporte não orientado para conexão: UDP.....	150
3.3.1	Estrutura do segmento UDP	152
3.3.2	Soma de verificação UDP	153
3.4	Princípios da transferência confiável de dados.....	154
3.4.1	Construindo um protocolo de transferência confiável de dados.....	155
3.4.2	Protocolos de transferência confiável de dados com paralelismo	163
3.4.3	Go-Back-N.....	166
3.4.4	Repetição seletiva (SR).....	170
3.5	Transporte orientado para conexão: TCP	174
3.5.1	A conexão TCP	174
3.5.2	Estrutura do segmento TCP.....	176
3.5.3	Estimativa do tempo de viagem de ida e volta e de esgotamento de temporização	180
3.5.4	Transferência confiável de dados.....	183
3.5.5	Controle de fluxo	189
3.5.6	Gerenciamento da conexão TCP.....	190
3.6	Princípios de controle de congestionamento	196
3.6.1	As causas e os custos do congestionamento	196
3.6.2	Mecanismos de controle de congestionamento	201
3.6.3	Exemplo de controle de congestionamento assistido pela rede: controle de congestionamento ATM ABR.....	202
3.7	Controle de congestionamento no TCP	203
3.7.1	Equidade.....	210
3.8	Resumo	213
Capítulo 4 A camada de rede		228
4.1	Introdução	229
4.1.1	Repasso e roteamento.....	229
4.1.2	Modelos de serviço de rede.....	232
4.2	Redes de circuitos virtuais e de datagramas.....	234
4.2.1	Redes de circuitos virtuais	234
4.2.2	Redes de datagramas	236
4.2.3	Origens das redes de circuitos virtuais e de datagramas	238
4.3	O que há dentro de um roteador?.....	239
4.3.1	Portas de entrada.....	240
4.3.2	Elemento de comutação	242
4.3.3	Portas de saída.....	244
4.3.4	Onde ocorre formação de fila?.....	244
4.4	O Protocolo da Internet (IP): repasse e endereçamento na Internet.....	247
4.4.1	Formato do datagrama	247
4.4.2	Endereçamento IPv4	252
4.4.3	Protocolo de Mensagens de Controle da Internet (ICMP)	262

4.4.4	IPv6	265
4.4.5	Uma breve investida em segurança IP	269
4.5	Algoritmos de roteamento.....	270
4.5.1	O algoritmo de roteamento de estado de enlace (LS)	273
4.5.2	O algoritmo de roteamento de vetor de distâncias (DV).....	276
4.5.3	Roteamento hierárquico.....	282
4.6	Roteamento na Internet.....	285
4.6.1	Roteamento intra-AS na Internet: RIP	285
4.6.2	Roteamento intra-AS na Internet: OSPF	288
4.6.3	Roteamento externo a sistemas autônomos: BGP.....	290
4.7	Roteamento broadcast e multicast	295
4.7.1	Algoritmos de roteamento broadcast	296
4.7.2	Multicast.....	300
4.8	Resumo	305
Capítulo 5 Camada de enlace e redes locais		318
5.1	Camada de enlace: introdução e serviços	318
5.1.1	Os serviços fornecidos pela camada de enlace.....	319
5.1.2	Onde a camada de enlace é implementada?	321
5.2	Técnicas de detecção e correção de erros	323
5.2.1	Verificações de paridade	324
5.2.2	Métodos de soma de verificação	326
5.2.3	Verificação de redundância cíclica (CRC)	326
5.3	Protocolos de acesso múltiplo.....	328
5.3.1	Protocolos de divisão de canal.....	330
5.3.2	Protocolos de acesso aleatório	331
5.3.3	Protocolos de revezamento	336
5.3.4	Redes locais (LANs)	337
5.4	Endereçamento na camada de enlace.....	338
5.4.1	Endereços MAC	338
5.4.2	ARP (protocolo de resolução de endereços).....	339
5.5	Ethernet.....	343
5.5.1	Estrutura do quadro Ethernet.....	344
5.5.2	CSMA/CD: o protocolo de acesso múltiplo da Ethernet	346
5.5.3	Tecnologias Ethernet	349
5.6	Comutadores de camada de enlace	351
5.6.1	Repasse e filtragem.....	351
5.6.3	Propriedades de comutação da camada de enlace	353
5.6.4	Comutadores versus roteadores	353
5.6.5	Rede local virtual (VLANs [Virtual Local Area Network])	355
5.7	PPP: o protocolo ponto a ponto.....	358
5.7.1	Enquadramento de dados PPP.....	359
5.8	Virtualização de enlace: uma rede como camada de enlace	361

5.9 Um dia na vida de uma solicitação de página Web	363
5.10 Resumo	367
Capítulo 6 Redes sem fio e redes móveis.....	377
6.1 Introdução	378
6.2 Características de enlaces e redes sem fio.....	380
6.2.1 CDMA	383
6.3 Wi-Fi: LANs sem fio 802.11	385
6.3.1 A arquitetura 802.11	387
6.3.2 O protocolo MAC 802.11	389
6.3.3 O quadro IEEE 802.11.....	394
6.3.4 Mobilidade na mesma sub-rede IP	396
6.3.5 Recursos Avançados em 802.11.....	397
6.3.6 Mais além de 802.11: Bluetooth e WiMAX.....	398
6.4 Acesso celular à Internet	401
6.4.1 Uma visão geral da arquitetura celular	402
6.5 Gerenciamento da mobilidade: princípios	405
6.5.1 Endereçamento.....	407
6.5.2 Roteamento para um nó móvel.....	409
6.6 IP móvel.....	412
6.7 Gerenciamento de mobilidade em redes celulares.....	415
6.7.1 Roteando chamadas para um usuário móvel	417
6.7.2 Transferências em GSM (handoffs)	418
6.8 Sem fio e mobilidade: impacto sobre protocolos de camadas superiores.....	420
6.9 Resumo	422
Capítulo 7 Redes multimídia.....	428
7.1 Aplicações de rede multimídia	429
7.1.1 Exemplos de aplicações multimídia.....	429
7.1.2 Obstáculos para a multimídia na Internet de hoje	431
7.1.3 Como a Internet deveria evoluir para dar melhor suporte à multimídia?	432
7.1.4 Compressão de áudio e vídeo	433
7.2 Áudio e vídeo de fluxo contínuo armazenados.....	436
7.2.1 Acesso a áudio e vídeo por meio de um servidor Web	436
7.2.2 Envio de multimídia de um servidor de fluxo contínuo a uma aplicação auxiliar	437
7.2.3 RTSP (protocolo de fluxo contínuo em tempo real)	439
7.3 Fazendo o melhor possível com o serviço de melhor esforço	442
7.3.1 As limitações de um serviço de melhor esforço	442
7.3.2 Eliminação da variação de atraso no receptor para áudio	444
7.3.3 Recuperação de perda de pacotes	447
7.3.4 Distribuição de multimídia: redes de distribuição de conteúdo.....	449
7.3.5 Dimensionando Redes de melhor desempenho para fornecer um serviço de qualidade	452

7.4	Protocolos para aplicações interativas em tempo real	453
7.4.1	Protocolo de tempo real (RTP).....	453
7.4.2	Protocolo de controle RTP (RTCP)	456
7.4.3	SIP	458
7.4.4	H.323.....	463
7.5	Fornecendo classes de serviço múltiplo	464
7.5.1	Cenários motivadores	465
7.5.2	Mecanismos de escalonamento e regulação.....	468
7.5.3	Diffserv	474
7.6	Fornecendo garantias de qualidade de serviços.....	478
7.6.1	Um exemplo motivador	478
7.6.2	Reserva de recurso, admissão de chamada e configuração de chamada	479
7.6.3	QoS garantida na Internet: Intserv e RSVP.....	482
7.7	Resumo	483

Capítulo 8 Segurança em redes de computadores..... 492

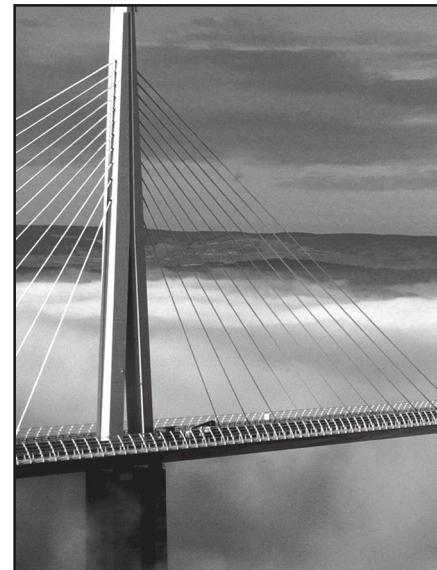
8.1	O que é segurança na rede?	493
8.2	Princípios de criptografia	494
8.2.1	Criptografia de chaves simétricas	495
8.2.2	Criptografia de chave pública	500
8.3	Integridade de mensagem e autenticação do ponto final.....	504
8.3.1	Funções de hash criptográficas.....	505
8.3.2	Código de autenticação da mensagem.....	506
8.3.3	Assinaturas digitais	507
8.3.4	Autenticação do ponto final.....	512
8.4	Protegendo o e-mail	516
8.4.1	E-mail seguro	517
8.4.2	PGP.....	520
8.5	Protegendo conexões TCP: SSL	521
8.5.1	Uma visão abrangente	522
8.5.2	Uma visão mais completa	524
8.6	Segurança na camada de rede: IPsec e redes virtuais privadas	526
8.6.1	IPsec e redes virtuais privadas (VPNs)	526
8.6.2	Os protocolos AH e ESP.....	527
8.6.3	Associações de segurança.....	527
8.6.4	O datagrama IPsec.....	528
8.6.5	Gerenciamento de chave no IPsec	531
8.7	Segurança em LANs sem fio.....	531
8.7.1	Privacidade Equivalente Cabeada (WEP)	532
8.7.2	IEEE 802.11i	533
8.8	Segurança operacional: firewalls e sistemas de detecção de invasão	535
8.8.1	Firewalls	535
8.8.2	Sistemas de detecção de intrusos	540
8.9	Resumo	544

Capítulo 9 Gerenciamento de rede	553
9.1 O que é gerenciamento de rede?	553
9.2 A infraestrutura do gerenciamento de rede	556
9.3 A estrutura de gerenciamento padrão da Internet	560
9.3.1 SMI (Estrutura de Informações de Gerenciamento).....	561
9.3.2 Base de informações de gerenciamento: MIB.....	564
9.3.3 Operações do protocolo SNMP e mapeamentos de transporte	565
9.3.4 Segurança e administração	568
9.4 ASN.1.....	570
9.5 Conclusão	574
Referências	578
Índice.....	603
Sobre os autores	615





Prefácio



Bem-vindo à quinta edição de *Redes de computadores e a Internet: uma abordagem top-down*. Desde a publicação da primeira edição, há nove anos, nosso livro foi adotado em centenas de universidades e instituições de ensino superior, traduzido para mais de dez idiomas e utilizado por mais de cem mil estudantes e profissionais no mundo inteiro. Muitos desses leitores entraram em contato conosco e ficamos extremamente satisfeitos com sua reação positiva.

Acreditamos que uma importante razão para esse sucesso é que o livro oferece uma abordagem moderna do ensino de redes de computadores. Você poderia perguntar: por que é necessária uma abordagem moderna? Nos últimos anos, testemunhamos duas mudanças revolucionárias no campo das redes — mudanças que não estão refletidas nos livros sobre o assunto publicados entre as décadas de 1980 e 1990.

Em primeiro lugar, a Internet dominou o universo das redes de computadores. Atualmente, em qualquer discussão séria sobre redes, é obrigatório ter a Internet em mente. Em segundo lugar, o maior crescimento nessa área ocorreu no âmbito dos serviços e das aplicações, o que pode ser confirmado pelo desenvolvimento da Web, pela utilização dos serviços de e-mail por milhares de usuários ao mesmo tempo, pela recepção de áudio e vídeo, pelo telefone por Internet, pelos serviços de mensagem instantânea, pelas aplicações P2P e pelo comércio eletrônico.

Quais são as novidades desta quinta edição?

Fizemos mudanças nesta quinta edição, mas mantivemos inalterados os aspectos que acreditamos ser (e os professores e estudantes que utilizaram nosso livro confirmaram) os mais importantes deste livro: a abordagem top-down, o foco na Internet, a atenção que demos aos princípios, à prática e sua abordagem e aos estilos acessíveis do ensino de redes de computadores.

Contudo, fizemos muitas mudanças significativas nesta edição. Iniciando pelo Capítulo 1, atualizamos a introdução referente a redes e atualizamos e expandimos nossa abordagem sobre redes de acesso (particularmente, o uso de redes a cabo, DSL e fiber-to-the-home como redes de acesso à Internet pública). No Capítulo 2, retiramos o material sobre a pesquisa ponto a ponto que se tornou fora de uso para dar espaço a uma nova seção sobre distributed hash tables. A apresentação do controle de congestionamento TCP no Capítulo 3 é agora baseada em uma representação gráfica (máquina de estado finito) de TCP, estruturando e esclarecendo nossa abordagem. O Capítulo 5 foi estendido significativamente, com seções novas sobre as redes locais virtuais (VLANS) e sobre “um dia na vida de uma solicitação de página Web”. Esta última seção relaciona todos os protocolos e atividades da rede envolvidos em satisfazer a solicitação aparentemente simples para buscar e exibir uma página Web de um servidor remoto, ajudando a ilustrar e sintetizar muito do material abordado nos cinco primeiros

capítulos. No Capítulo 6, removemos um pouco da “sopa de letrinhas” de padrões e protocolos da telefonia celular e acrescentamos uma nova seção sobre a arquitetura de redes de celular e como a rede de celular e a Internet operam em conjunto para prover serviços da Internet a aparelhos móveis como o Blackberry e o iPhone. Nossa abordagem sobre a segurança da rede, no Capítulo 8, sofreu uma revisão significativa. O conteúdo sobre a autenticação do ponto de chegada, encadeamento de blocos de cifras e criptografia da chave pública foi revisado, e o material sobre o IPsec foi reescrito e estendido para incluir as redes virtuais privadas (VPNs, do inglês virtual private networks). Ao longo de todo o livro, incluímos exemplos modernos e referências atualizadas.

Público-alvo

Este livro destina-se a um estudo inicial de redes de computadores. Pode ser usado em cursos de ciência da computação e de engenharia elétrica. Em termos de linguagem de programação, requer que os estudantes conheçam as linguagens C, C++ ou Java (mas apenas em alguns lugares). Embora este livro seja mais minucioso e analítico do que muitos outros de introdução às redes de computadores, raramente utiliza conceitos matemáticos que não sejam ensinados no ensino médio. Fizemos um esforço deliberado para evitar o uso de quaisquer conceitos avançados de cálculo, probabilidade ou processos estocásticos (embora tenhamos incluído alguns problemas para alunos com tal conhecimento avançado). Consequentemente, o livro é apropriado para cursos de graduação e para o primeiro ano dos cursos de pós-graduação. É também muito útil para os profissionais do setor de telecomunicações.

O que há de singular neste livro?

O assunto rede de computadores é bastante vasto e complexo e envolve muitos conceitos, protocolos e tecnologias que se entrelaçam inextricavelmente. Para dar conta desse escopo e complexidade, muitos livros sobre redes são, em geral, organizados de acordo com as “camadas” de uma arquitetura de rede. Com a organização em camadas, os estudantes podem vislumbrar a complexidade das redes de computadores — eles aprendem os conceitos e os protocolos distintos de uma parte da arquitetura e, ao mesmo tempo, visualizam o grande quadro da interconexão entre as camadas. Do ponto de vista pedagógico, nossa experiência pessoal confirma que essa abordagem em camadas é, de fato, muito boa. Entretanto, achamos que a abordagem tradicional, a bottom-up — da camada física para a camada de aplicação —, não é a melhor abordagem para um curso moderno de redes de computadores.

Uma abordagem top-down

Na primeira edição, propusemos uma inovação adotando uma visão top-down — isto é, começando na camada de aplicação e descendo até a camada física.

A abordagem top-down oferece diversos benefícios importantes. Em primeiro lugar, o livro dá ênfase à camada de aplicação, que tem sido a área de “grande crescimento” das redes de computadores. De fato, muitas das recentes revoluções nesse ramo — incluindo a Web, o compartilhamento de arquivos P2P e o fluxo contínuo de mídia — tiveram lugar nessa camada. Essa abordagem de ênfase inicial à camada de aplicação é diferente das seguidas por muitos outros livros, que têm apenas pouco material sobre aplicações de redes, seus requisitos, paradigmas da camada de aplicação (por exemplo, cliente-servidor e ponto a ponto) e interfaces de programação de aplicação. Em segundo lugar, nossa experiência como professores (e a de muitos outros que utilizaram este livro) confirma que ensinar aplicações de rede logo no início do curso é uma poderosa ferramenta motivadora. Os estudantes ficam mais entusiasmados ao aprender como funcionam as aplicações de rede — aplicações como o e-mail e a Web, que a maioria deles usa diariamente. Entendendo as aplicações, o estudante pode entender os serviços de rede necessários ao suporte de tais aplicações. Pode também, por sua vez, examinar as várias maneiras como esses serviços são fornecidos e implementados nas camadas mais baixas. Assim, a discussão das aplicações logo no início fornece a motivação necessária para os demais assuntos do livro. Em terceiro lugar, essa abordagem top-down habilita o professor a apresentar o desenvolvimento de aplicações de rede no estágio inicial.

Os estudantes não apenas veem como funcionam aplicações e protocolos populares, como também aprendem que é fácil criar suas próprias aplicações e protocolos de aplicação de rede. Com a abordagem top-down, eles entram imediatamente em contato com as noções de interfaces de programação de aplicações (application programming interfaces — API) e com os modelos de serviços e protocolos — conceitos importantes que reaparecem em todas as camadas subsequentes. Ao apresentar exemplos de programação de sockets em Java, destacamos as ideias centrais sem confundir os estudantes com códigos complexos. Estudantes de engenharia elétrica e ciência da computação provavelmente não terão dificuldades para entender o código Java.

Um foco na Internet

Continuamos a utilizar a arquitetura e os protocolos da Internet como veículo primordial para estudar conceitos fundamentais de redes de computadores. É claro que também incluímos conceitos e protocolos de outras arquiteturas de rede. Mas os holofotes estão claramente dirigidos à Internet, fato refletido na organização do livro, que gira em torno da arquitetura de cinco camadas da Internet: aplicação, transporte, rede, enlace e física.

Outro benefício de colocar a Internet sob os holofotes é que a maioria dos estudantes de ciência da computação e de engenharia elétrica está ávida por conhecer a Internet e seus protocolos. Eles sabem que a Internet é uma tecnologia revolucionária e inovadora e podem constatar que ela está provocando uma profunda transformação em nosso mundo. Dada a enorme relevância da Internet, os estudantes estão naturalmente curiosos em saber o que há por trás dela. Assim, fica fácil para um professor manter seus alunos interessados nos princípios básicos, usando a Internet como guia.

Ensinando princípios de rede

Duas das características exclusivas deste livro — sua abordagem top-down e seu foco na Internet — aparecem no título e subtítulo. Se pudéssemos, teríamos acrescentado uma terceira palavra ao subtítulo — princípios. O campo das redes agora está suficientemente maduro para que uma quantidade de assuntos de importância fundamental possa ser identificada. Por exemplo, na camada de transporte, entre os assuntos importantes estão a comunicação confiável por uma camada de rede não confiável, o estabelecimento/encerramento de conexões e mútua apresentação, o controle de congestionamento e de fluxo e a multiplexação. Na camada de rede, dois assuntos muito importantes são: como determinar “bons” caminhos entre dois roteadores e como interconectar um grande número de redes heterogêneas. Na camada de enlace, um problema fundamental é como compartilhar um canal de acesso múltiplo. Na segurança de rede, técnicas para prover sigilo, autenticação e integridade de mensagens são baseadas em fundamentos da criptografia. Este livro identifica as questões fundamentais de redes e apresenta abordagens para enfrentar essas questões. Aprendendo esses princípios, o estudante adquire conhecimento de longa validade — muito tempo após os padrões e protocolos de rede de hoje tornarem-se obsoletos, os princípios que ele incorpora continuarão importantes e relevantes. Acreditamos que o uso da Internet para apresentar o assunto aos estudantes e a ênfase dada à abordagem das questões e das soluções permitirão que os alunos entendam rapidamente qualquer tecnologia de rede.



Material de apoio

Companion
Website

No site de apoio do livro (www.aw.com/kurose_br), professores e estudantes podem acessar materiais adicionais em qualquer dia, durante 24 horas.

Para professores:

Fornecemos um pacote de suplementos para auxiliar o ensino do conteúdo deste livro.

Manual de soluções (em inglês). O site do livro oferece um manual de soluções, em inglês, para os exercícios apresentados ao final de cada capítulo.

Apresentações em PowerPoint. O site do livro apresenta slides em PowerPoint para todos os nove capítulos. Os slides dão detalhes completos sobre cada capítulo e neles usamos gráficos e animações — e não apenas marcadores — para tornar sua aula visualmente interessante e atraente. Fornecemos os slides originais aos professores para que eles possam personalizá-los do modo que melhor atenda às suas necessidades de ensino. Alguns desses slides foram fornecidos por professores que utilizaram nosso livro em seus cursos.

Material de aprendizagem interativo. O site Web contém diversos applets Java interativos, que animam muitos dos principais conceitos de redes. O site também possui quizzes interativos que permitem que o aluno verifique sua compreensão básica sobre o assunto. Os professores podem completar essas características interativas com aulas ou usá-las como minilabs.

Material técnico adicional. Removemos a abordagem de alguns assuntos existentes para deixar o livro com um volume razoável. Por exemplo, para abrir espaço para o novo conteúdo desta edição, retiramos o material sobre as redes ATM e pesquisas ponto a ponto. Esse material pode ser encontrado no site Web do livro.

Tarefas de programação. O site Web também provê diversas tarefas de programação detalhadas, que incluem construir um servidor Web multitarefa, construir um cliente de e-mail com uma interface GUI, programar os lados remetentes e destinatários de um protocolo de transporte de dados confiável, programar um algoritmo de roteamento distribuído e muito mais.

Wireshark labs. A compreensão sobre protocolos de rede pode ser extremamente aprofundada ao observá-los em ação. O site Web provê diversas tarefas Wireshark que permitem que os estudantes observem a sequência de mensagens trocadas entre duas entidades de protocolos. O site Web inclui Wireshark labs distintos em HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, Wi-Fi, SSL e no rastreamento de todos os protocolos envolvidos em satisfazer a solicitação para buscar uma página Web. Novos labs serão adicionados com o tempo.

Esse material é de uso exclusivo para professores e está protegido por senha. Para ter acesso a eles, os professores que adotam o livro devem entrar em contato com seu representante Pearson ou enviar e-mail para universitarios@pearsoned.com.

Para estudantes:

Exercícios on-line autocorrigíveis. Teste seus conhecimentos com exercícios de múltipla escolha e, se quiser, envie o resultado para seu professor.

Características pedagógicas

Há quase 20 anos damos aulas de redes de computadores. Adicionamos a este livro uma experiência agregada de mais de 45 anos de ensino para milhares de estudantes. Durante esse período, também participamos ativamente na área de pesquisas sobre redes de computadores. (De fato, Jim e Keith se conheceram quando faziam mestrado, frequentando um curso sobre redes de computadores ministrado por Mischa Schwartz, em 1979, na Universidade de Colúmbia.) Achamos que isso nos dá uma boa perspectiva do que foi a rede e de qual será, provavelmente, seu futuro. Não obstante, resistimos às tentações de dar ao material deste livro um viés que favorecesse nossos projetos de pesquisa prediletos. Se você estiver interessado em nossas pesquisas, consulte nosso site Web pessoal. Este livro é sobre redes de computadores modernas — é sobre protocolos e tecnologias contemporâneas, bem como sobre os princípios subjacentes a esses protocolos e tecnologias. Também achamos que aprender (e ensinar!) redes pode ser divertido. Esperamos que um certo senso de humor e a utilização de analogias e exemplos do mundo real que aparecem neste livro tornem este material ainda mais divertido.

Adendos, princípios na prática e segurança em foco

O campo das redes de computadores tem uma história rica e fascinante. Fizemos, neste livro, um esforço especial para contar a história das redes de computadores, o qual se materializou em uma seção histórica especial no Capítulo 1 e em 12 adendos históricos espalhados pelos demais capítulos. Nesses artigos históricos, apresentamos a invenção da comutação de pacotes, a evolução da Internet, o nascimento de importantes empresas gigantes de redes, como a Cisco e a 3Com, e muitos outros eventos relevantes. Os estudantes certamente se sentirão estimulados por esses acontecimentos históricos. Em cada capítulo, incluímos um adendo especial que acentua um princípio importante de rede de computadores. Esses adendos auxiliarão o estudante a compreender alguns dos conceitos fundamentais que estão sendo aplicados às redes modernas. Um pouco de nossa abordagem mais abrangente sobre segurança da rede aparece nos textos complementares “Segurança em foco” em cada um dos capítulos principais deste livro.

Entrevistas

Inserimos uma outra característica original que inspiraram e motivaram os leitores — entrevistas com inovadores famosos no campo de redes. Apresentamos entrevistas com Leonard Kleinrock, Bram Cohen, Sally Floyd, Vinton G. Cerf, Simon S. Lam, Charlie Perkins, Henning Schulzrinne, Steven M. Bellovin e Jeff Case.

O primeiro capítulo deste livro apresenta um apanhado geral sobre redes de computadores. Com a introdução de muitos conceitos e terminologias fundamentais, ele monta o cenário para o restante do livro. Todos os outros capítulos dependem diretamente desse primeiro. Recomendamos que os professores, após o terem completado, percorram em sequência os capítulos 2 ao 5, seguindo nossa filosofia top-down. Cada um dos cinco primeiros capítulos utiliza material dos capítulos precedentes. Após ter completado os cinco primeiros capítulos, o professor terá bastante flexibilidade. Não há interdependência entre os quatro últimos capítulos, de modo que eles podem ser ensinados em qualquer ordem. Conhecemos professores que, após ensinar o capítulo introdutório, ensinam o Capítulo 5 e depois trabalham de baixo para cima, ou até mesmo professores que começam a ensinar pelo meio (Capítulo 4) e depois não seguem nenhuma ordem. Contudo, cada um dos quatro depende de material dos cinco primeiros. Muitos professores ensinam os primeiros cinco capítulos e então um dos quatro últimos para arrematar.

Uma nota final: gostaríamos que vocês entrassem em contato conosco

Incentivamos professores e estudantes a nos enviarem e-mails (em inglês) com qualquer comentário que possam ter sobre o livro. Incentivamos os professores, também, a nos enviar novos problemas (e soluções) que complementem os problemas existentes. Publicaremos tudo isso no espaço para professores no site Web. Recomendamos aos professores e alunos que criem novas aplicações Java que ilustrem os conceitos e protocolos apresentados neste livro. Se você tiver uma aplicação que julgue apropriada para este livro, apresente-a, por gentileza, aos autores (em inglês). Se a aplicação (incluindo notação e terminologia) for adequada, teremos o prazer de incluí-la no site Web do livro em inglês (www.aw.com/kurose-ross), com a devida referência a seus autores.

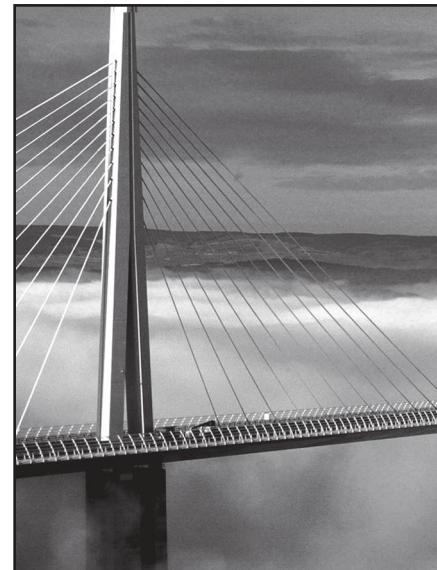
Não deixem de nos enviar URLs interessantes ou de discordar de qualquer uma de nossas afirmações e nos informar o que funciona e o que não funciona. Digam-nos, ainda, o que poderia (ou o que não deveria) ser incluído na próxima edição. Enviem seus e-mails para kurose@cs.umass.edu e ross@poly.edu.

Até a data de publicação deste livro, todos os URLs mencionados no texto estavam funcionando. Devido ao caráter dinâmico da Internet, porém, alguns deles podem mudar ou deixar de existir. Para tentar encontrar os sites e informações mencionados, use uma ferramenta de busca como www.google.com ou www.yahoo.com.





Agradecimentos



Desde o início deste projeto, em 1996, muitas pessoas nos deram inestimável auxílio e influenciaram nossas ideias sobre como melhor organizar e ministrar um curso sobre redes. Nossa MUITO OBRIGADO a todos os que nos ajudaram, sobretudo às centenas de estudantes, acadêmicos e profissionais que nos enviaram sugestões e comentários sobre as edições anteriores e sugestões para futuras edições deste livro. Nossos agradecimentos especiais para:

Al Aho (Columbia University)
Hisham Al-Mubaid (University of Houston-Clear Lake)
Pratima Akkunoor (Arizona State University)
Paul Amer (University of Delaware)
Shamiul Azom (Arizona State University)
Lichun Bao (University of California at Irvine)
Paul Barford (University of Wisconsin)
Bobby Bhattacharjee (University of Maryland)
Steven Bellovin (Columbia University)
Pravin Bhagwat (Wibhu)
Supratik Bhattacharyya (previously at Sprint)
Ernst Biersack (Eurécom Institute)
Shahid Bokhari (University of Engineering & Technology, Lahore)
Jean Bolot (Sprint)
Daniel Brushteyn (ex-aluno da University of Pennsylvania)
Ken Calvert (University of Kentucky)
Evandro Cantu (Universidade Federal de Santa Catarina)
Jeff Case (SNMP Research International)
Jeff Chaltas (Sprint)

Vinton Cerf (Google)
Byung Kyu Choi (Michigan Technological University)
Bram Cohen (BitTorrent, Inc.)
Constantine Coutras (Pace University)
John Daigle (University of Mississippi)
Edmundo A. de Souza e Silva (Universidade Federal do Rio de Janeiro)
Philippe Decuetos (Eurécom Institute)
Christophe Diot (Thomson Research)
Prithula Dhunghel (Polytechnic Institute of NYU)
Michalis Faloutsos (University of California at Riverside)
Wu-chi Feng (Oregon Graduate Institute)
Sally Floyd (ICIR, University of California at Berkeley)
Paul Francis (Max Planck Institute)
Lixin Gao (University of Massachusetts)
JJ Garcia-Luna-Aceves (University of California at Santa Cruz)
Mario Gerla (University of California at Los Angeles)
David Goodman (Polytechnic University)
Tim Griffin (Cambridge University)
Max Hailperin (Gustavus Adolphus College)

- Bruce Harvey (Florida A&M University, Florida State University)
Carl Hauser (Washington State University)
Rachelle Heller (George Washington University)
Phillipp Hoschka (INRIA/W3C)
Wen Hsin (Park University)
Albert Huang (ex-aluno da University of Pennsylvania)
Esther A. Hughes (Virginia Commonwealth University)
Jobin James (University of California at Riverside)
Sugih Jamin (University of Michigan)
Shivkumar Kalyanaraman (Rensselaer Polytechnic Institute)
Jussi Kangasharju (University of Darmstadt)
Sneha Kasera (University of Utah)
Hyojin Kim (ex-aluno da University of Pennsylvania)
Leonard Kleinrock (University of California at Los Angeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Arizona State University)
Rakesh Kumar (Bloomberg)
Miguel A. Labrador (University of South Florida)
Simon Lam (University of Texas)
Steve Lai (Ohio State University)
Tom LaPorta (Penn State University)
Tim-Berners Lee (World Wide Web Consortium)
Lee Leitner (Drexel University)
Brian Levine (University of Massachusetts)
William Liang (former University of Pennsylvania student)
Willis Marti (Texas A&M University)
Nick McKeown (Stanford University)
Josh McKinzie (Park University)
Deep Medhi (University of Missouri, Kansas City)
Bob Metcalfe (International Data Group)
Sue Moon (KAIST)
Erich Nahum (IBM Research)
Christos Papadopoulos (Colorado State University)
Craig Partridge (BBN Technologies)
Radia Perlman (Sun Microsystems)
Jitendra Padhye (Microsoft Research)
Vern Paxson (University of California at Berkeley)
Kevin Phillips (Sprint)
George Polyzos (Athens University of Economics and Business)
Sriram Rajagopalan (Arizona State University)
Ramachandran Ramjee (Microsoft Research)
Ken Reek (Rochester Institute of Technology)
Martin Reisslein (Arizona State University)
Jennifer Rexford (Princeton University)
Leon Reznik (Rochester Institute of Technology)
Sumit Roy (University of Washington)
Avi Rubin (Johns Hopkins University)
Dan Rubenstein (Columbia University)
Douglas Salane (John Jay College)
Despina Sarapilla (Cisco Systems)
Henning Schulzrinne (Columbia University)
Mischa Schwartz (Columbia University)
Harish Sethu (Drexel University)
K. Sam Shanmugan (University of Kansas) Prashant Shenoy (University of Massachusetts)
Clay Shields (Georgetown University)
Subin Shrestha (University of Pennsylvania)
Mihail L. Sichitiu (NC State University)
Peter Steenkiste (Carnegie Mellon University)
Tatsuya Suda (University of California at Irvine)
Kin Sun Tam (State University of New York at Albany)
Don Towsley (University of Massachusetts)
David Turner (California State University, San Bernardino)
Nitin Vaidya (University of Illinois)
Michele Weigle (Clemson University)
David Wetherall (University of Washington)
Ira Winston (University of Pennsylvania)
Di Wu (Polytechnic Institute of NYU)
Raj Yavatkar (Intel)
Yechiam Yemini (Columbia University)
Ming Yu (State University of New York at Binghamton)
Ellen Zegura (Georgia Institute of Technology)
Honggang Zhang (Suffolk University)
Hui Zhang (Carnegie Mellon University)
Lixia Zhang (University of California at Los Angeles)
Shuchun Zhang (ex-aluno da University of Pennsylvania)
Xiaodong Zhang (Ohio State University)
ZhiLi Zhang (University of Minnesota)
Phil Zimmermann (consultor independente)
Cliff C. Zou (University of Central Florida)



Gostaríamos de agradecer a Honggang Zhang, da Suffolk University, por revisar e aperfeiçoar alguns problemas presentes nesta edição. Queremos agradecer, também, a toda a equipe da Addison-Wesley — em particular, a Michael Hirsch, Marilyn Lloyd e Stephanie Sellinger — que fizeram um trabalho realmente notável nesta quinta edição (e que teve de suportar dois autores muito complicados e quase sempre atrasados).

Agradecemos aos artistas gráficos Janet Theurer e Patrice Rossi Calkin, pelo trabalho que executaram nas figuras deste livro, e a Nesbitt Graphics, Harry Druding e Rose Kernan, pelo maravilhoso trabalho de produção desta edição.

Finalmente, um agradecimento muito especial a Susan Hartman, nossa antiga editora na Addison-Wesley, e, mais uma vez, a Michael Hirsch, nosso editor na Addison-Wesley. Este livro não seria o que é (e talvez nem tivesse existido) sem a administração cordial de ambos, constante incentivo, paciência quase infinita, bom humor e perseverança.

Os editores da edição brasileira dedicam este livro ao prof. Nery Machado Filho (in memoriam).

Nota do Revisor Técnico

Mais uma vez foi com imensa satisfação que pude apreciar e rever a tradução do material apresentado em *Redes de computadores e a Internet: uma abordagem top-down*. De minha parte, sou grato pelos comentários recebidos sobre a primeira edição e conto com a colaboração de alunos, professores e profissionais de redes para a melhoria do trabalho.

WLZ

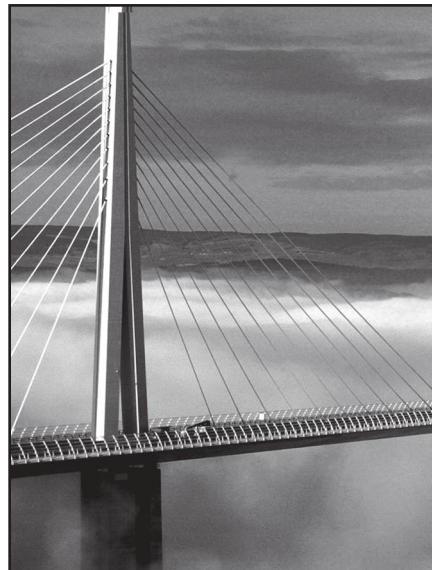






Capítulo 1

Redes de computadores e a Internet



A Internet de hoje é provavelmente o maior sistema de engenharia já criado pela humanidade, com centenas de computadores conectados, links de comunicação e comutadores; centenas de milhares de usuários que se conectam esporadicamente por meio de telefones celulares e PDAs; e dispositivos como sensores, webcams, console para jogos, quadros de imagens, e até mesmo máquinas de lavar sendo conectadas à Internet. Dado que a Internet é tão ampla e possui inúmeros componentes e utilidades, há a possibilidade de compreender como ela (e, em um amplo sentido, as redes de computadores) funciona? Existem princípios de orientação e estrutura que forneçam um fundamento para a compreensão de um sistema surpreendentemente complexo e abrangente? Se a resposta for sim, é possível que, nos dias de hoje, seja interessante e divertido aprender sobre rede de computadores? Felizmente, as respostas para todas essas perguntas é um retumbante SIM! Na verdade, nosso objetivo neste livro é fornecer uma introdução moderna ao campo dinâmico das redes de computadores, apresentando os princípios e o entendimento prático necessários para utilizar não apenas as redes de hoje, como também as de amanhã.

O primeiro capítulo apresenta um panorama de redes de computadores e da Internet. Nossa objetivo é pintar um quadro amplo que nos permita ver a floresta por entre as árvores. Cobriremos um terreno bastante extenso neste capítulo de introdução e discutiremos várias peças de uma rede de computadores, sem perder de vista o quadro geral. Este capítulo lança as fundações para o restante do livro.

O panorama geral de redes de computadores que apresentaremos neste capítulo será estruturado como segue. Após apresentarmos brevemente a terminologia e os conceitos fundamentais, examinaremos primeiramente os componentes básicos de hardware e software que compõem uma rede. Partiremos da periferia da rede e examinaremos os sistemas finais e aplicações de rede executados nela. Consideraremos os serviços de transporte fornecidos a essas aplicações. Em seguida exploraremos o cerne de uma rede de computadores examinando os enlaces e comutadores que transportam dados, bem como as redes de acesso e meios físicos que conectam sistemas finais ao núcleo da rede. Aprenderemos que a Internet é uma rede de redes e observaremos como essas redes se conectam umas com as outras.

Após concluirmos essa revisão sobre a periferia e o núcleo de uma rede de computadores, adotaremos uma visão mais ampla e mais abstrata na segunda metade deste capítulo. Examinaremos as causas do atraso de transferência de dados e das perdas em uma rede de computadores e forneceremos modelos quantitativos simples para o atraso fim a fim, modelos que levam em conta atrasos de transmissão, propagação e fila. Depois apresentaremos

alguns princípios fundamentais de arquitetura em redes de computadores, a saber: protocolos em camadas e modelos de serviço. Aprenderemos, também, que as redes de computadores são vulneráveis a diferentes tipos de ameaças; analisaremos algumas dessas ameaças e como a rede de computadores pode se tornar mais segura. Finalmente, encerraremos este capítulo com um breve histórico da computação em rede.

1.1 O que é a Internet?

Neste livro, usamos a Internet pública, uma rede de computadores específica, como o veículo principal para discutir as redes de computadores e seus protocolos. Mas o que é a Internet? Há diversas maneiras de responder a essa questão. Primeiro, podemos descrever detalhadamente os aspectos principais da Internet, ou seja, os componentes de software e hardware básicos que a formam. Segundo, podemos descrever a Internet em termos de uma infraestrutura de redes que fornece serviços para aplicações distribuídas. Iniciaremos com a descrição dos componentes, utilizando a Figura 1.1 como ilustração para a nossa discussão.

1.1.1 Uma descrição dos componentes da rede

A Internet é uma rede de computadores que interconecta milhares de dispositivos computacionais ao redor do mundo. Há pouco tempo, esses dispositivos eram basicamente computadores de mesa, estações de trabalho Linux, e os assim chamados servidores que armazenam e transmitem informações, como páginas da Web e mensagens de e-mail. No entanto, cada vez mais sistemas finais modernos da Internet, como TVs, laptops, consoles para jogos, telefones celulares, webcams, automóveis, dispositivos de sensoriamento ambiental, quadros de imagens, e sistemas internos elétricos e de segurança, estão sendo conectados à rede. Realmente, o termo *rede de computadores* está começando a soar um tanto desatualizado, dados os muitos equipamentos não tradicionais que estão sendo ligados à Internet. No jargão da Internet, todos esses equipamentos são denominados hospedeiros ou sistemas finais. Em julho de 2008, havia aproximadamente 600 milhões de sistemas finais ligados à Internet [ISC, 2009], sem contar os telefones celulares, laptops e outros dispositivos que são conectados à rede de maneira intermitente.

Sistemas finais são conectados entre si por enlaces (links) de comunicação e comutadores de pacotes. Na Seção 1.2, veremos que há muitos tipos de enlaces de comunicação, que são constituídos de diferentes tipos de meios físicos, entre eles cabos coaxiais, fios de cobre, fibras ópticas e ondas de rádio. Enlaces diferentes podem transmitir dados em taxas diferentes, sendo a taxa de transmissão de um enlace medida em bits por segundo.

Quando um sistema final possui dados para enviar a outro sistema final, o sistema emissor segmenta esses dados e adiciona bytes de cabeçalho a cada segmento. Os pacotes de informações resultantes, conhecidos como pacotes no jargão de rede de computadores, são enviados através da rede ao sistema final de destino, onde são reagregados aos dados originais.

Um comutador de pacotes encaminha o pacote que está chegando em um de seus enlaces de comunicação de entrada para um de seus enlaces de comunicação de saída. Há comutadores de pacotes de todos os tipos e formas, mas os dois mais proeminentes na Internet de hoje são roteadores e comutadores de camada de enlace (*switches*). Esses dois tipos de comutadores encaminham pacotes a seus destinos finais. Os comutadores de camada de enlace são tipicamente utilizados em redes de acesso, enquanto os roteadores são utilizados principalmente no núcleo da rede. A sequência de enlaces de comunicação e comutadores de pacotes que um pacote percorre desde o sistema final remetente até o sistema final receptor é conhecida como rota ou caminho através da rede. É difícil de estimar a exata quantidade de tráfego na Internet [Odylsko, 2003]. A PriMetrica [PriMetrica, 2009] estima que 10 terabits por segundo de capacidade internacional foram utilizados por provedores da Internet pública em 2008, sendo que essa capacidade duplica, aproximadamente, a cada dez anos.

As redes comutadas por pacotes (que transportam pacotes) são, de muitas maneiras, semelhantes às redes de transporte de rodovias, estradas e cruzamentos (que transportam veículos). Considere, por exemplo, uma fábrica que precise transportar uma quantidade de carga muito grande a algum depósito localizado a milhares de

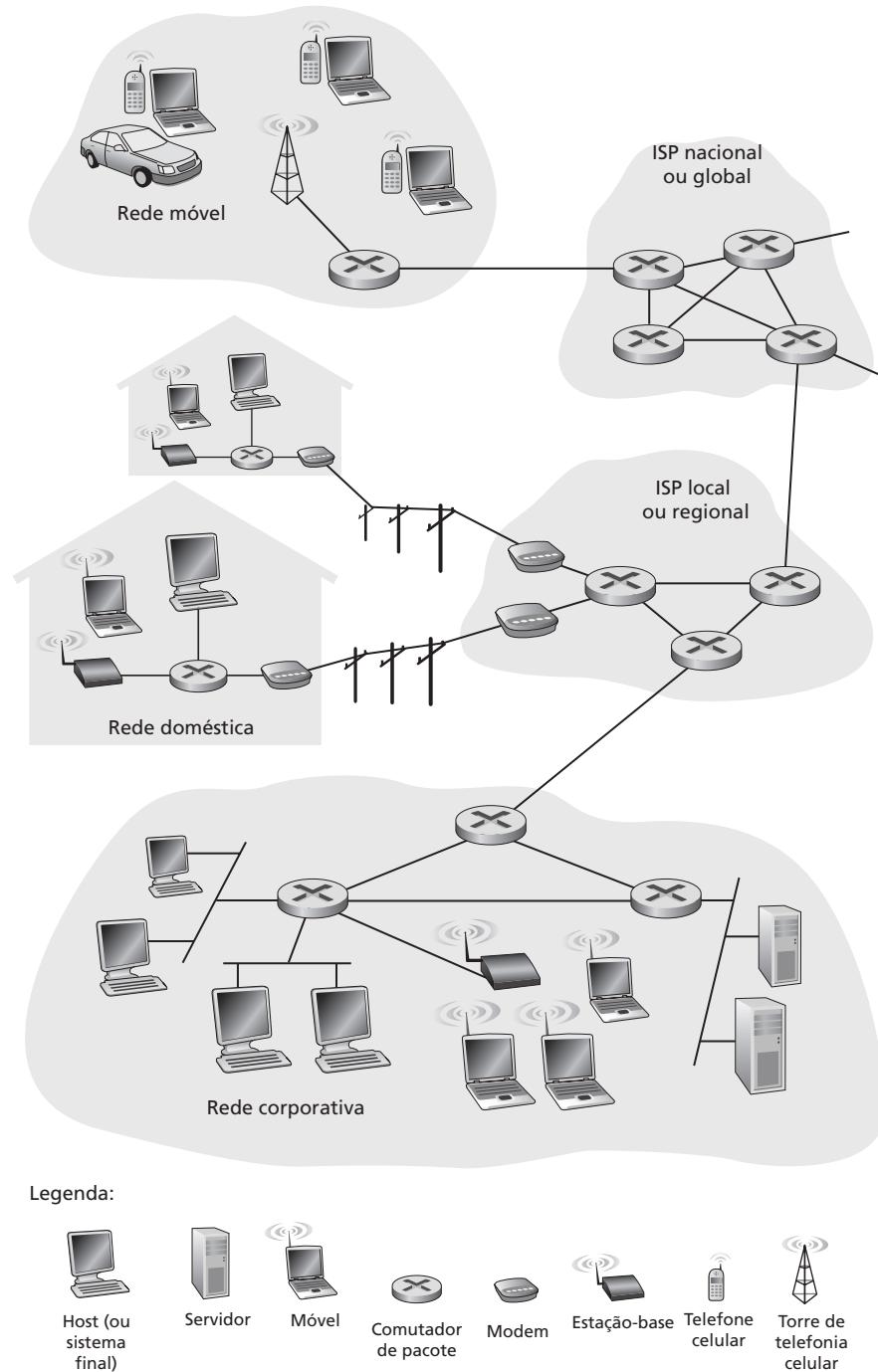


Figura 1.1 Alguns componentes da Internet

quilômetros. Na fábrica, a carga é dividida e carregada em uma frota de caminhões. Cada caminhão viaja, independentemente, pela rede de rodovias, estradas e cruzamentos ao depósito de destino. No depósito, a carga é descarregada e agrupada com o resto da carga pertencente à mesma remessa. Deste modo, os pacotes se assemelham aos caminhões, os enlaces de comunicação se assemelham às rodovias e estradas, os comutadores de pacote se assemelham aos cruzamentos e cada sistema final se assemelha aos prédios. Assim como o caminhão faz o percurso pela rede de transporte, o pacote utiliza uma rede de computadores.

Sistemas finais acessam a Internet por meio de Provedores de Serviços de Internet (Internet Service Providers — ISPs), entre eles ISPs residenciais como empresas de TV a cabo ou empresas de telefonia; ISPs corporativos, ISPs de universidades e ISPs que fornecem acesso sem fio em aeroportos, hotéis, cafés e outros locais públicos. Cada ISP é uma rede de comutadores de pacotes e enlaces de comunicação. ISPs proveem aos sistemas finais uma variedade de tipos de acesso à rede, incluindo acesso por modem discado de 56 kbps, acesso residencial de banda larga como modem de cabo coaxial ou DSL (linha digital de assinante), acesso por LAN de alta velocidade e acesso sem fio. ISPs também fornecem acesso a provedores de conteúdo, conectando sites Web diretamente à Internet. Esta se interessa pela conexão entre os sistemas finais, portanto os ISPs que fornecem acesso a esses sistemas também devem se interconectar. Esses ISPs de nível mais baixo são interconectados por meio de ISPs de nível mais alto, nacionais e internacionais, como AT&T e Sprint. Um ISP de nível mais alto consiste em roteadores de alta velocidade interconectados com enlaces de fibra ótica de alta velocidade. Cada rede ISP, seja de nível mais alto ou mais baixo, é gerenciada de forma independente, executa o protocolo IP (ver adiante) e obedece a certas convenções de nomeação e endereço. Examinaremos ISPs e sua interconexão mais detalhadamente na Seção 1.3.

Os sistemas finais, os comutadores de pacotes e outras peças da Internet executam protocolos que controlam o envio e o recebimento de informações. O TCP (Transmission Control Protocol — Protocolo de Controle de Transmissão) e o IP (Internet Protocol — Protocolo da Internet) são dois dos protocolos mais importantes da Internet. O protocolo IP especifica o formato dos pacotes que são enviados e recebidos entre roteadores e sistemas finais. Os principais protocolos da Internet são conhecidos coletivamente como TCP/IP. Começaremos a examinar protocolos neste capítulo de introdução. Mas isso é só um começo — grande parte deste livro trata de protocolos de redes de computadores!

Dada a importância de protocolos para a Internet, é adequado que todos concordem sobre o que cada um dos protocolos faz. É aqui que os padrões entram em ação. Padrões da Internet são desenvolvidos pela IETF (Internet Engineering Task Force — Força de Trabalho de Engenharia da Internet) [IETF, 2004]. Os documentos padronizados da IETF são denominados RFCs (*request for comments* — pedido de comentários). Os RFCs começaram como solicitações gerais de comentários (por isso o nome) para resolver problemas de arquitetura que a precursora da Internet enfrentava. Os RFCs tendem a ser bastante técnicos e detalhados. Definem protocolos como TCP, IP, HTTP (para a Web) e SMTP (Protocolo Simples de Transferência de Correio — Simple Mail Transfer Protocol) (para e-mail). Atualmente, existem mais de 5.000 RFCs. Outros organismos também especificam padrões para componentes de rede, mais notavelmente para enlaces da rede. O IEEE 802 LAN/MAN Standards Committee [IEEE, 802, 2009], por exemplo, especifica os padrões Ethernet e Wi-Fi sem fio.

1.1.2 Uma descrição do serviço

A discussão anterior identificou muitos dos componentes que compõem a Internet. Mas também podemos descrevê-la partindo de um ângulo completamente diferente — ou seja, como *uma infraestrutura que provê serviços a aplicações*. Tais aplicações incluem correio eletrônico, navegação na Web, mensagem instantânea, Voz sobre IP (VoIP), Internet via rádio, vídeo em tempo real, jogos distribuídos, compartilhamento de arquivos peer-to-peer (P2P), televisão pela Internet, login remoto e muito mais. Essas aplicações são conhecidas como aplicações distribuídas, uma vez que envolvem diversos sistemas finais que trocam informações mutuamente. De forma significativa, as aplicações da Internet são executadas em sistemas finais — e não em comutadores de pacote no núcleo da rede. Embora os comutadores de pacote facilitem a troca de dados entre os sistemas, eles não estão relacionados com a aplicação, que é a fonte ou destino de dados.

Vamos explorar um pouco mais o significado de uma infraestrutura que fornece serviços a aplicações. Nesse sentido, suponha que você tenha uma grande ideia para uma aplicação distribuída para a Internet, uma que possa beneficiar a humanidade ou que simplesmente o enriqueça e o torne famoso. Como transformar essa ideia em uma aplicação real da Internet? Como as aplicações são executadas em sistemas finais, você precisará criar componentes de software que sejam executados em sistemas finais. Você pode, por exemplo, criar seus componentes em Java, C, ou Python. Agora, já que você está desenvolvendo uma aplicação distribuída para a Internet, os componentes do software executados em diferentes sistemas finais precisarão enviar dados uns aos outros. E, aqui,

chegamos ao assunto principal — o que leva ao modo alternativo de descrever a Internet como uma plataforma para aplicações. De que modo um componente da aplicação, executado em um sistema final, orienta a Internet a enviar dados a outro componente de software executado em outro sistema final?

Os sistemas finais ligados à Internet proveem uma Interface de Programação de Aplicação (API) que especifica como o componente do software que é executado no sistema final solicita à infraestrutura da Internet que envie dados a um componente de software de destino específico, executado em outro sistema final. A API da Internet é um conjunto de regras que o software emissor deve cumprir para que a Internet seja capaz de enviar os dados ao componente de destino. Discutiremos a API da Internet mais detalhadamente no Capítulo 2. Agora, vamos traçar uma simples comparação, que será utilizada com frequência neste livro. Suponha que Alice queria enviar uma carta para Bob utilizando o serviço postal. Alice, é claro, não pode simplesmente escrever a carta (os dados) e atirá-la pela janela. Em vez disso, o serviço postal necessita que Alice coloque a carta em um envelope; escreva o nome completo de Bob, endereço e CEP no centro do envelope; feche; coloque um selo no canto superior direito do envelope; e, finalmente, coloque o envelope em uma caixa de correio oficial. Dessa maneira, o serviço postal possui sua própria “API de serviço postal”, ou conjunto de regras, que Alice deve cumprir para que sua carta seja entregue a Bob. De um modo semelhante, a Internet possui uma API que o software emissor de dados deve seguir para que a Internet envie os dados para o software receptor.

O serviço postal, naturalmente, fornece mais de um serviço a seus clientes: entrega expressa, confirmação de envio, carta simples e muito mais. Igualmente, a Internet provê diversos serviços a suas aplicações. Ao desenvolver uma aplicação para a Internet, você também deve escolher um dos serviços que a rede oferece. Uma descrição dos serviços da Internet será apresentada no Capítulo 2.

Esta segunda descrição da Internet — uma infraestrutura de fornecimento de serviços a aplicações distribuídas — é muito importante. Cada vez mais, os avanços na tecnologia dos componentes da Internet estão sendo guiados pelas necessidades de novas aplicações. Portanto, é importante ter sempre em mente que a Internet é uma infraestrutura na qual novas aplicações estão constantemente sendo inventadas e disponibilizadas.

Acabamos de apresentar duas descrições da Internet: uma delas diz respeito a seus componentes de hardware e software, e a outra, aos serviços que ela oferece a aplicações distribuídas. Mas talvez você ainda esteja confuso sobre o que é a Internet. O que é comutação de pacotes, TCP/IP e API? O que são roteadores? Que tipos de enlaces de comunicação estão presentes na Internet? O que é uma aplicação distribuída? Como uma torradeira ou um sensor de variações meteorológicas pode ser ligado à Internet? Se você está um pouco assustado com tudo isso agora, não se preocupe — a finalidade deste livro é lhe apresentar os mecanismos da Internet e também os princípios que determinam como e por que ela funciona. Explicaremos esses termos e questões importantes nas seções e nos capítulos subsequentes.

1.1.3 O que é um protocolo?

Agora que já entendemos um pouco o que é a Internet, vamos considerar uma outra palavra fundamental usada em redes de computadores: *protocolo*. O que é um protocolo? O que um protocolo faz?

Uma analogia humana

Provavelmente é mais fácil entender a ideia de um protocolo de rede de computadores considerando primeiramente algumas analogias humanas, já que executamos protocolos o tempo todo. Considere o que você faz quando quer perguntar as horas a alguém. Um diálogo comum é ilustrado na Figura 1.2.

O protocolo humano (ou as boas maneiras, ao menos) ordena que, ao iniciarmos uma comunicação com outra pessoa, primeiramente a cumprimos (o primeiro “oi” da Figura 1.2). A resposta comum para um “oi” é um outro “oi”. Implicitamente, tomamos a resposta cordial “oi” como uma indicação de que podemos prosseguir e perguntar as horas. Uma resposta diferente ao “oi” inicial (tal como “Não me perturbe!”, “I don’t speak Portuguese!” ou alguma resposta impública) poderia indicar falta de vontade ou incapacidade de comunicação. Nesse caso, o protocolo humano seria não perguntar que horas são. Às vezes, não recebemos nenhuma resposta para uma pergunta, caso em que normalmente desistimos de perguntar as horas à pessoa.

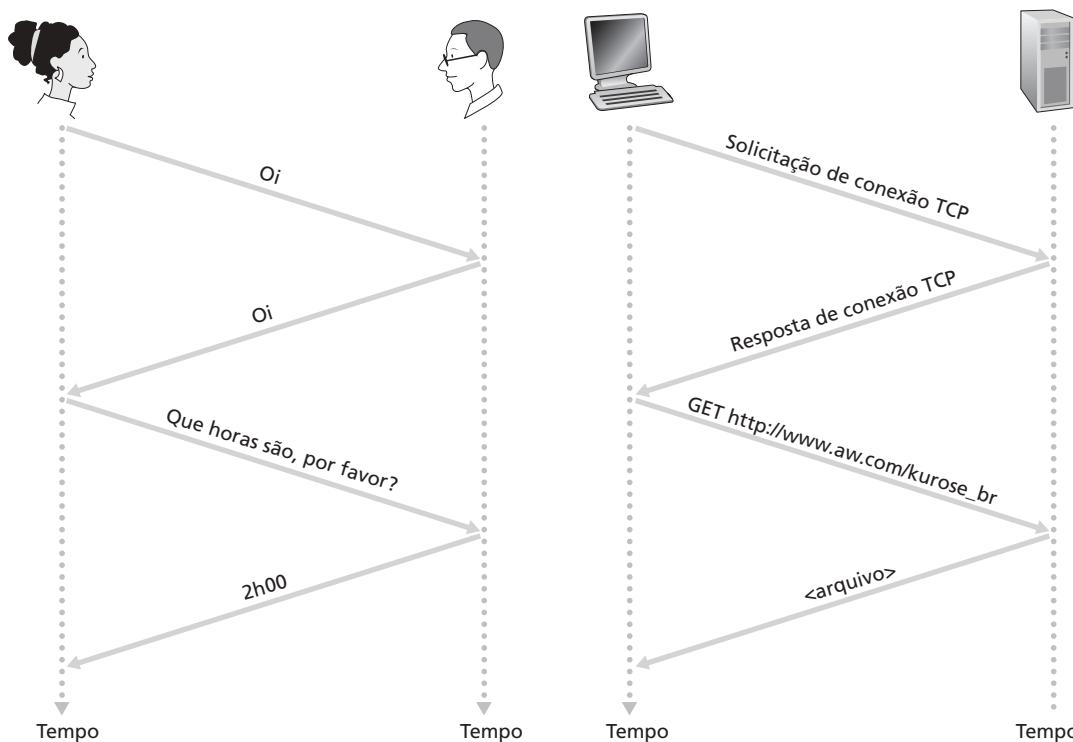


Figura 1.2 Um protocolo humano e um protocolo de rede de computadores

Note que, no nosso protocolo humano, *há mensagens específicas que enviamos e ações específicas que realizamos em reação às respostas recebidas ou a outros eventos (como nenhuma resposta após certo tempo)*. É claro que mensagens transmitidas e recebidas e ações realizadas quando essas mensagens são enviadas ou recebidas ou quando ocorrem outros eventos desempenham um papel central em um protocolo humano. Se as pessoas executarem protocolos diferentes (por exemplo, se uma pessoa tem boas maneiras, mas a outra não; se uma delas entende o conceito de horas, mas a outra não), os protocolos não interagem e nenhum trabalho útil pode ser realizado. O mesmo é válido para redes — é preciso que duas (ou mais) entidades comunicantes executem o mesmo protocolo para que uma tarefa seja realizada.

Vamos considerar uma segunda analogia humana. Suponha que você esteja assistindo a uma aula (sobre redes de computadores, por exemplo). O professor está falando monotonamente sobre protocolos e você está confuso. Ele para e pergunta: “Alguma dúvida?” (uma mensagem que é transmitida a todos os alunos e recebida por todos os que não estão dormindo). Você levanta a mão (transmitindo uma mensagem implícita ao professor). O professor percebe e, com um sorriso, diz “Sim...” (uma mensagem transmitida, incentivando-o a fazer sua pergunta — professores *adoram* perguntas) e você então faz sua pergunta (isto é, transmite sua mensagem ao professor). O professor ouve (recebe sua mensagem) e responde (transmite uma resposta a você). Mais uma vez, percebemos que a transmissão e a recepção de mensagens e um conjunto de ações convencionais, realizadas quando as mensagens são enviadas e recebidas, são o coração desse protocolo de pergunta e resposta.

Protocolos de rede

Um protocolo de rede é semelhante a um protocolo humano; a única diferença é que as entidades que trocam mensagens e realizam ações são componentes de hardware ou software de algum equipamento (por exemplo, computador, PDA, telefones celulares, roteador ou outro equipamento habilitado para rede). Todas as atividades na Internet que envolvem duas ou mais entidades remotas comunicantes são governadas por um protocolo. Por exemplo, protocolos implementados em hardware nas placas de interface de rede de dois computadores conectados fisicamente controlam o fluxo de bits no ‘cabo’ entre as duas placas de interface de rede; protocolos

de controle de congestionamento em sistemas finais controlam a taxa com que os pacotes são transmitidos entre a origem e o destino; protocolos em roteadores determinam o caminho de um pacote da origem ao destino. Protocolos estão em execução por toda a Internet e, consequentemente, grande parte deste livro trata de protocolos de rede de computadores.

Como exemplo de um protocolo de rede de computadores com o qual você provavelmente está familiarizado, considere o que acontece quando fazemos uma requisição a um servidor Web, isto é, quando digitamos o URL de uma página Web no browser. Isso é mostrado no lado direito da Figura 1.2. Primeiramente, o computador enviará uma mensagem de requisição de conexão ao servidor Web e aguardará uma resposta. O servidor receberá essa mensagem de requisição de conexão e retornará uma mensagem de resposta de conexão. Sabendo que agora está tudo certo para requisitar o documento da Web, o computador envia então o nome da página Web que quer buscar naquele servidor com uma mensagem GET. Por fim, o servidor retorna a página da Web (arquivo) para o computador.

Dados o exemplo humano e o exemplo de rede anteriores, as trocas de mensagens e as ações realizadas quando essas mensagens são enviadas e recebidas são os elementos fundamentais para a definição de um protocolo:

Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.

A Internet e as redes de computadores em geral fazem uso intenso de protocolos. Diferentes tipos de protocolos são usados para realizar diferentes tarefas de comunicação. À medida que for avançando na leitura deste livro, você perceberá que alguns protocolos são simples e diretos, enquanto outros são complexos e intelectualmente profundos. Dominar a área de redes de computadores equivale a entender o que são, por que existem e como funcionam os protocolos de rede.

1.2 A periferia da Internet

Nas seções anteriores, apresentamos uma descrição da Internet e de protocolos de rede fazendo analogias com nossos atos. Agora passaremos a tratar com um pouco mais de profundidade os componentes de uma rede de computadores (e da Internet, em particular). Nesta seção, começamos pela periferia de uma rede e examinamos os componentes com os quais estamos mais familiarizados — a saber, computadores, PDAs, telefones celulares e outros equipamentos que usamos diariamente. Na seção seguinte, passaremos da periferia para o núcleo da rede e estudaremos comutação e roteamento em redes de computadores.

Como descrito na seção anterior, no jargão de rede de computadores, os computadores e outros dispositivos conectados à Internet são frequentemente chamados de sistemas finais. Eles são assim chamados porque se encontram na periferia da Internet, como mostrado na Figura 1.3. Os sistemas finais da Internet incluem computadores de mesa (por exemplo, PCs de mesa, MACs e Linux Boxes), servidores (por exemplo, Web e servidores de e-mails), e computadores móveis (por exemplo, computadores portáteis, PDAs e telefones com conexão sem fio à Internet). Além disso, um crescente número de aparelhos alternativos estão sendo utilizados com a Internet como sistemas finais (veja página seguinte).

Sistemas finais também são denominados *hospedeiros (hosts)* porque hospedam (isto é, executam) programas de aplicação, tais como um programa browser da Web, um programa servidor da Web, um programa leitor de e-mail ou um servidor de e-mail. Neste livro, utilizaremos os termos hospedeiros e sistemas finais como sinônimos. Às vezes, sistemas finais são ainda subdivididos em duas categorias: clientes e servidores. Informalmente, clientes costumam ser PCs de mesa ou móveis, PDAs e assim por diante, ao passo que servidores tendem a ser máquinas mais poderosas que armazenam e distribuem páginas Web, vídeo em tempo real, retransmissão de e-mails e assim por diante.

1.2.1 Programas clientes e servidores

No contexto de software de rede há ainda uma outra definição de cliente e de servidor, à qual nos referiremos em todo o livro. Um programa cliente é um programa que funciona em um sistema final, que solicita e recebe um

História

Um conjunto impressionante de sistemas finais da Internet

Não faz muito tempo, os sistemas finais conectados à Internet eram primordialmente computadores tradicionais, como máquinas de mesa e servidores de grande capacidade. Desde o final da década de 1990 até hoje, um amplo leque de equipamentos e dispositivos interessantes, cada vez mais diversos, vem sendo conectado à Internet. A característica comum desses equipamentos é que eles precisam enviar e receber dados digitais de, e para, outros equipamentos. Tendo em vista a onipresença da Internet, seus protocolos bem definidos (padronizados) e a disponibilidade comercial de hardware capacitado para ela, é natural usar sua tecnologia para interconectar esses equipamentos.

Alguns deles parecem ter sido criados exclusivamente para diversão. Um computador de mesa utilizando IP com recurso de moldura de foto [Ceiva, 2009] solicita fotos digitais de um servidor remoto e as apresenta em um dispositivo que parece uma moldura tradicional de fotografia; uma torradeira da Internet baixa informações meteorológicas de um servidor e grava uma imagem da previsão do tempo do dia em questão (por exemplo, nublado, com sol) na sua torrada matinal [BBC, 2001]. Outros dispositivos fornecem informações úteis — câmeras Web apresentam as condições meteorológicas e de tráfego ou fazem monitoramento para fornecer uma localização de interesse; eletrodomésticos conectados à Internet (incluindo máquinas de lavar roupas, geladeiras e fogões) podem ser monitorados e controlados remotamente por meio de um browser. Telefones celulares que utilizam IP com recursos de GPS (como o novo iPhone da Apple) permitem fácil navegação pela Web e transmissão de e-mails e mensagens. Uma nova classe de sistemas de sensoriamento em rede promete revolucionar o modo como observamos e interagimos com nosso ambiente. Redes de sensores incorporadas ao ambiente físico permitem a monitoração de edifícios, pontes, atividade sísmica, *habitats* da fauna selvagem, estuários e das camadas inferiores da atmosfera [CENS, 2009, CASA, 2009]. Aparelhos biomédicos podem ser incorporados e conectados em rede, causando numerosos problemas de privacidade e segurança [Halperin, 2008]. Uma tag de RFID ou um sensor integrado ligado a qualquer objeto pode fornecer informações a respeito desse objeto na Internet, levando a uma “Internet de coisas” [ITU, 2005].

serviço de um programa servidor, que funciona em um outro sistema final. A Web, o e-mail, a transferência de arquivo, o login remoto, os grupos de discussão e muitas outras aplicações populares adotam o modelo cliente-servidor. Uma vez que um programa cliente normalmente roda em um computador e o programa servidor, em outro, aplicações cliente-servidor de Internet são, por definição, aplicações distribuídas. O programa cliente e o programa servidor interagem enviando mensagens um para o outro pela Internet. Nesse nível de abstração, os roteadores, enlaces e outros componentes da Internet funcionam como uma caixa-preta que transfere mensagens entre os componentes distribuídos, comunicantes, de um aplicação da Internet. Esse é o nível de abstração representado na Figura 1.3.

Nem todas as aplicações da Internet de hoje consistem em programas puramente clientes que interagem com programas puramente servidores. Muitas aplicações são, cada vez mais, peer-to-peer (P2P), nas quais os sistemas finais interagem e executam programas que apresentam funções de servidor e de cliente. Por exemplo, em aplicações P2P de compartilhamento de arquivos (como o BitTorrent e o eMule), o programa no sistema final do usuário funciona como um cliente quando solicita um arquivo de outro par; e o programa funciona como um servidor quando envia um arquivo para outro par. Na telefonia por Internet, as duas partes comunicantes interagem como pares; uma parte não requisita serviço da outra em nenhum sentido. Examinaremos detalhadamente as semelhanças e diferenças entre arquiteturas cliente-servidor e P2P no Capítulo 2.

1.2.2 Redes de acesso

Tendo considerado as aplicações e sistemas finais na “periferia da Internet”, vamos agora considerar as redes de acesso — os enlaces físicos que conectam um sistema final ao primeiro roteador (também conhecido como “roteador

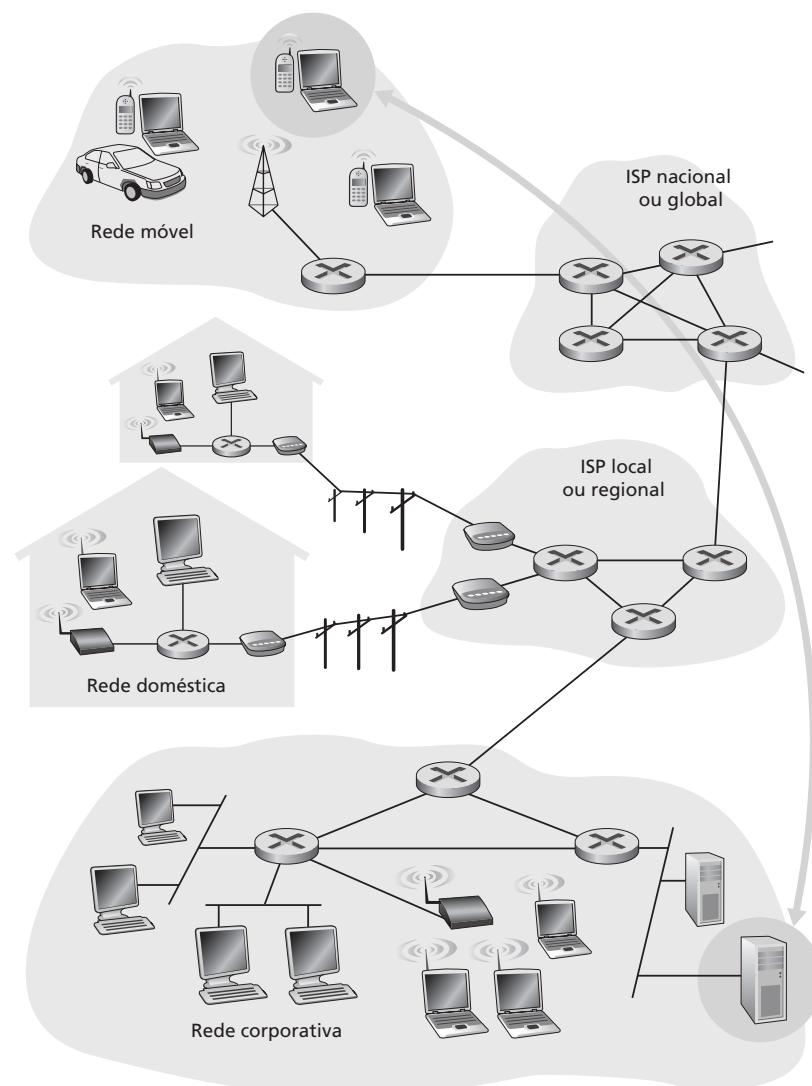


Figura 1.3 Intereração entre sistemas finais

de borda”) partindo de um sistema final até outro qualquer. A Figura 1.4 apresenta diversos tipos de enlaces de acesso do sistema final ao roteador de periferia; os enlaces de acesso estão destacados nas linhas cinzas mais espessas. Esta seção analisa muitas das tecnologias mais comuns em rede de acesso, em termos gerais, de baixa à alta velocidade.

Veremos mais adiante que muitas dessas tecnologias empregam, em níveis que variam, parcelas da tradicional infraestrutura telefônica com fio. Essa infraestrutura é fornecida por um servidor telefônico local, que simplesmente chamaremos de operadora. São exemplos de operadoras a Verizon, nos Estados Unidos, e a France Telecom, na França. Cada residência (casa e apartamento) possui um par direto de fios de cobre trançado para um comutador da operadora na região, localizado em um edifício chamado “central telefônica” (CT) no jargão da telefonia. (Discutiremos par de fios de cobre trançado mais adiante nesta seção.) Uma operadora local possuirá, normalmente, centenas de CTs e ligará cada cliente à sua CT mais próxima.

Dial-up (discado)

Nos anos 1990, quase todos os usuários residenciais acessavam a Internet por meio de linhas telefônicas analógicas utilizando um modem discado. Atualmente, muitos usuários de países não desenvolvidos e de áreas rurais em países desenvolvidos (onde o acesso à banda larga é indisponível) ainda têm acesso à Internet discada. Na verdade, estima-se que 10% dos usuários residenciais nos Estados Unidos usavam Internet discada em 2008 [Pew, 2008].

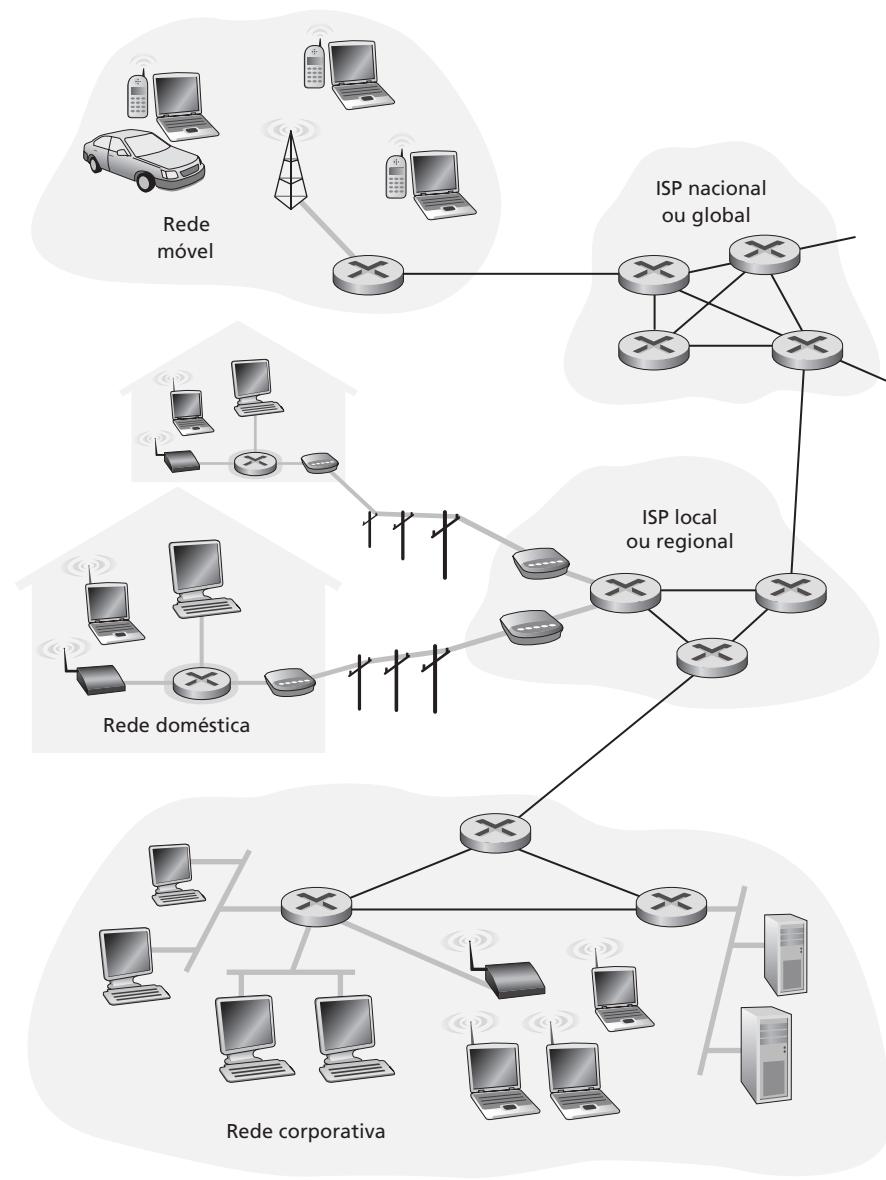


Figura 1.4 Acesso a redes

Emprega-se o termo “dial-up” porque o software do usuário, na verdade, disca (em inglês, *dial*) um número de telefone do ISP e realiza uma ligação telefônica tradicional com o ISP (por exemplo, com a AOL). Como mostrado na Figura 1.5, o computador é ligado a um modem discado, que, por sua vez, é conectado à linha telefônica analógica da residência. Essa linha telefônica é composta por cabos de cobre trançados e é a mesma linha telefônica utilizada para fazer ligações comuns. O modem da residência converte a saída digital do computador em um formato analógico apropriado para transmissão pela linha telefônica analógica. Na outra extremidade da conexão, um modem do ISP converte o sinal analógico em forma digital para inserir dados no roteador do ISP.

O acesso discado à Internet possui duas desvantagens principais. Em primeiro lugar, é extremamente lento, fornecendo uma taxa máxima de 56 kbps. A essa taxa, o download de uma música em formato MP3 de três minutos leva, aproximadamente, oito minutos, e um dia é o tempo que leva o download de um filme de 1 Gigabyte! Em segundo lugar, o acesso discado bloqueia a linha telefônica comum do usuário — enquanto uma pessoa usa o modem para navegar na Web, ninguém mais pode receber ou realizar ligações comuns com a mesma linha.

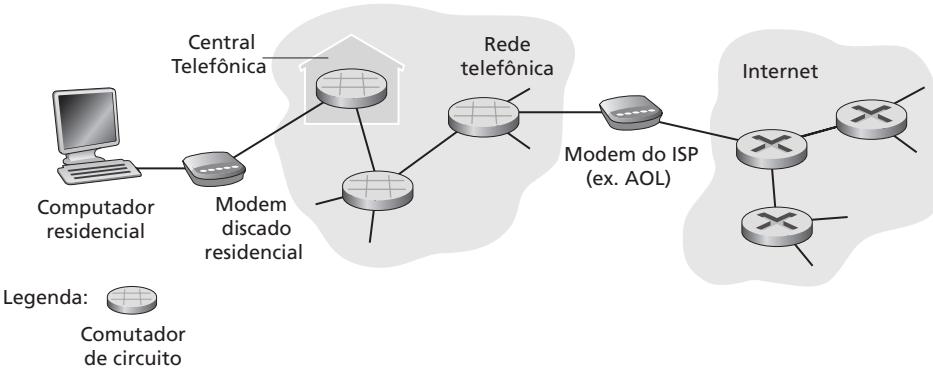


Figura 1.5 Acesso à Internet discada

DSL

Atualmente, os dois tipos de acesso residencial banda larga mais predominantes são a linha digital de assinante (DSL) ou a cabo. Em muitos países desenvolvidos hoje, mais de 50% das residências possuem acesso banda larga, sendo que a Coreia do Sul, Islândia, Holanda, Dinamarca e Suíça lideram o terreno com mais de 74% de penetração nas residências em 2008 [ITIF, 2008]. Nos Estados Unidos, a tecnologia DSL e a cabo têm a mesma participação no mercado para acesso banda larga [Pew, 2008]. Fora dos Estados Unidos e do Canadá, a DSL domina particularmente na Europa, onde em muitos países mais de 90% das conexões banda larga são DSL.

Normalmente uma residência obtém acesso DSL à Internet da mesma empresa que fornece acesso telefônico local com fio (por exemplo, a operadora local). Assim, quando a DSL é utilizada, uma operadora do cliente é também seu provedor de serviços de Internet (ISP). Como mostrado na Figura 1.6, cada modem DSL do cliente utiliza a linha telefônica existente (par de fios de cobre trançado) para trocar dados com um multiplexador digital de acesso à linha do assinante (DSLAM), normalmente localizado na CT da operadora. A linha telefônica conduz, simultaneamente, dados e sinais telefônicos, que são codificados em diferentes frequências:

- um canal downstream* de alta velocidade, com uma banda de 50 kHz a 1 MHz;
- um canal upstream* de velocidade média, com uma banda de 4 kHz a 50 kHz;
- um canal de telefone bidirecional comum, com uma banda de 0 a 4 kHz.

Essa abordagem faz com que a conexão DSL pareça três conexões distintas, de modo que um telefonema e a conexão com a Internet podem compartilhar a conexão DSL ao mesmo tempo. (Descreveremos essa técnica de multiplexação por divisão de frequência na Seção 1.3.1.) Do lado do consumidor, para os sinais que chegam até sua casa, um distribuidor separa os dados e os sinais telefônicos e conduz o sinal com os dados para o modem DSL. Na operadora, na CT, o DSLAM separa os dados e os sinais telefônicos e envia aqueles para a Internet. Centenas ou mesmo milhares de residências se conectam a um único DSLAM [Cha, 2009; Dischinger, 2007].

A tecnologia DSL possui duas principais vantagens sobre o acesso discado. Primeiro, ela pode transmitir e receber dados a taxas muito mais elevadas. Normalmente, um cliente DSL terá uma taxa de transmissão na faixa de 1 a 2 Mbps (da CT para a residência) e a taxa de recebimento de 128 kbps a 1 Mbps. Em razão de as taxas de transmissão e recebimento serem diferentes, o acesso é conhecido como assimétrico. A segunda vantagem principal é que os usuários podem, simultaneamente, falar ao telefone e acessar à Internet. Diferentemente do acesso dial-up, os usuários não discam um número de telefone do ISP para acessar à rede; pelo contrário, eles têm uma conexão permanente ao DSLAM do provedor (e, portanto, à Internet).

* N.R.T.: Os termos “downstream” e “upstream” descrevem a transmissão de dados *para* o cliente e *do* cliente, respectivamente.

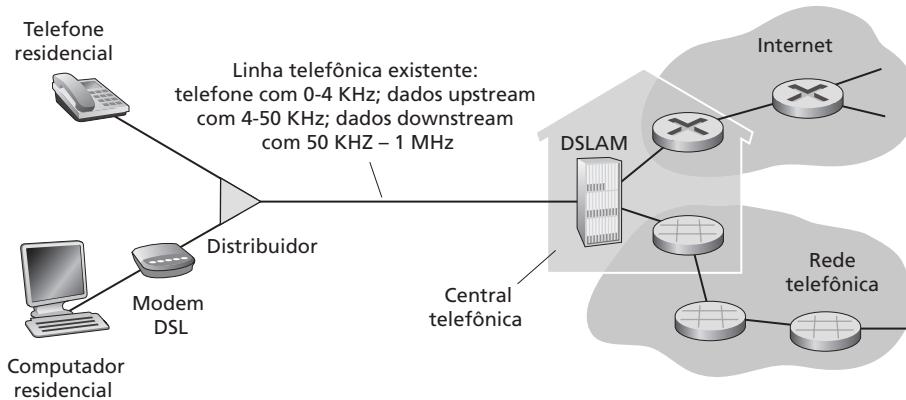


Figura 1.6 Acesso à Internet DSL

A taxa de transmissão de downstream e upstream real disponível para as residências é uma função da distância entre a casa e a CT, permitindo taxas de transmissão mais elevadas do que no acesso discado. Para aumentar a taxa de dados, a DSL conta com um processamento de sinal avançado e algoritmos de correção de erro, que podem levar a grandes atrasos de pacotes. No entanto, se a residência não estiver localizada dentro de 8 a 15 quilômetros da CT, a tecnologia de processamento de sinal DSL não será mais eficaz e a residência deve recorrer a uma forma alternativa de acesso à Internet.

Há também uma variedade de tecnologias DSL de alta velocidade em alguns países atualmente. Por exemplo, a tecnologia DSL de alta velocidade (VDSL), com a maior penetração, hoje, na Coreia e no Japão, apresenta taxas impressionantes de 12 a 55 Mbps para downstream e 1.6 a 20 Mbps para upstream [DSL, 2009].

Cabo

Muitas residências na América do Norte e em outros lugares recebem centenas de canais de televisão por redes de cabo coaxial. (Discutiremos cabo coaxial mais adiante nesta seção.) Em um sistema tradicional de televisão a cabo, um terminal de distribuição transmite canais de televisão através de uma rede de distribuição de cabo coaxial e amplificadores para as residências.

Enquanto as tecnologias DSL e dial-up utilizam-se da infraestrutura telefônica local existente da operadora, o acesso à Internet a cabo utiliza-se da infraestrutura de televisão a cabo existente da empresa de TV a cabo. Uma residência obtém acesso à Internet a cabo da mesma empresa que fornece a televisão a cabo. Como ilustrado na Figura 1.7, as fibras ópticas conectam o terminal de distribuição às junções da região, sendo o cabo coaxial tradicional utilizado para chegar às casas e apartamentos individualmente. Cada junção normalmente suporta de 500 a 5.000 casas. Em razão de a fibra e o cabo coaxial fazerem parte desse sistema, a rede é denominada híbrida fibra-coaxial (HFC).

O acesso à Internet a cabo necessita de modems especiais, denominados modems a cabo. Como o modem DSL, o modem a cabo é, normalmente, um aparelho externo que se conecta ao computador residencial através da porta Ethernet. (Discutiremos Ethernet em detalhes no Capítulo 5.) Os modems a cabo dividem a rede HFC em dois canais, um de transmissão (downstream) e um de recebimento (upstream). Como a tecnologia DSL, o acesso normalmente é assimétrico, com o canal downstream recebendo uma taxa de transmissão maior do que a do canal upstream.

Uma característica importante do acesso a cabo é o fato de ser um meio de transmissão compartilhado. Em especial, cada pacote enviado pelo terminal viaja pelos enlaces downstream até cada residência e cada pacote enviado por uma residência percorre o canal upstream até o terminal de transmissão. Por essa razão, se diversos usuários estiverem fazendo o download de um arquivo em vídeo simultaneamente no canal downstream, cada usuário receberá o arquivo a uma taxa significativamente menor do que a taxa de transmissão a cabo. Por outro lado, se há somente alguns usuários ativos que estão navegando na Web, então cada um poderá receber páginas da Web a uma taxa de downstream máxima, pois esses usuários raramente solicitarão uma página da Web ao mesmo tempo. Como o canal upstream também é compartilhado, é necessário um protocolo de acesso múltiplo distribuído para coordenar as transmissões e evitar colisões. (Discutiremos a questão de colisão no Capítulo 5.)

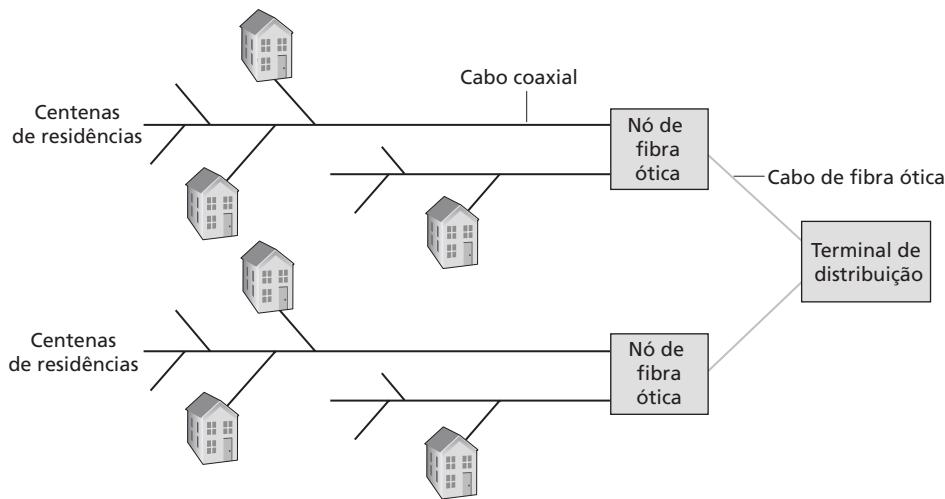


Figura 1.7 Uma rede de acesso híbrida fibra-coaxial

Os defensores da tecnologia DSL se apressam em frisar que ela é uma conexão ponto a ponto entre a residência e o ISP e que, portanto, a capacidade de transmissão total da conexão DSL entre a residência e o ISP é dedicada, e não compartilhada. Os defensores do cabo, no entanto, argumentam que uma rede HFC razoavelmente dimensionada provê taxas de transmissão mais altas do que a DSL. Não há dúvidas de que a guerra entre DSL e HFC pelo acesso residencial de alta velocidade já começou, principalmente na América do Norte. Nas áreas rurais, onde nem a DSL e a HFC estão disponíveis, pode ser utilizada uma conexão por satélite para conectar uma residência à Internet em velocidades superiores a 1Mbps; o StarBand e o HughesNet são dois provedores de acesso via satélite.

FTTH (Fiber-To-The-Home)

A fibra ótica (que será discutida na Seção 1.2.3) pode oferecer taxas de transmissão significativamente mais altas do que o par de fios de cobre trançado ou o cabo coaxial. Algumas operadoras locais (em muitos países diferentes), que recentemente passaram a utilizar fibra ótica nas residências, agora oferecem acesso à Internet de alta velocidade, bem como serviços telefônicos e televisivos através de fibras ópticas. Nos Estados Unidos, a Verizon saiu na frente com a tecnologia FTTH, lançando o serviço FIOS [Verizon FIOS, 2009].

Existem várias tecnologias concorrentes para a distribuição ótica das CTs às residências. A rede mais simples de distribuição ótica é chamada fibra direta, para a qual existe uma fibra saindo da CT para cada residência. Essa distribuição pode fornecer uma alta largura de banda, uma vez que cada cliente recebe sua própria fibra dedicada diretamente da central telefônica. Geralmente, cada fibra que sai da central telefônica é compartilhada por várias residências; a fibra é dividida em fibras individuais do cliente somente após ela se aproximar relativamente das residências. Existem duas arquiteturas concorrentes de rede de distribuição ótica que apresentam essa divisão: redes ópticas ativas (AONs) e redes ópticas passivas (PONs). A AON é basicamente a Ethernet comutada, assunto discutido no Capítulo 5. Aqui, falaremos brevemente sobre a PON, que é utilizada no serviço FIOS da Verizon. A Figura 1.8 mostra a FTTH utilizando a arquitetura de distribuição de PON. Cada residência possui um terminal de rede ótica (ONT), que é conectado por uma fibra ótica dedicada a um distribuidor da região. O distribuidor combina um número de residências (normalmente menos de 100) a uma única fibra ótica compartilhada, que se liga a um terminal de linha ótica (OLT) na CT da operadora. O OLT, que fornece conversão entre os sinais ópticos e elétricos, se conecta a Internet por meio de um roteador da operadora. Na residência, o usuário conecta um roteador residencial (geralmente um roteador sem fio) ao ONT e acessa à Internet através desse roteador. Na arquitetura de PON, todos os pacotes enviados do OLT ao distribuidor são nele replicados (semelhante ao terminal de distribuição a cabo).

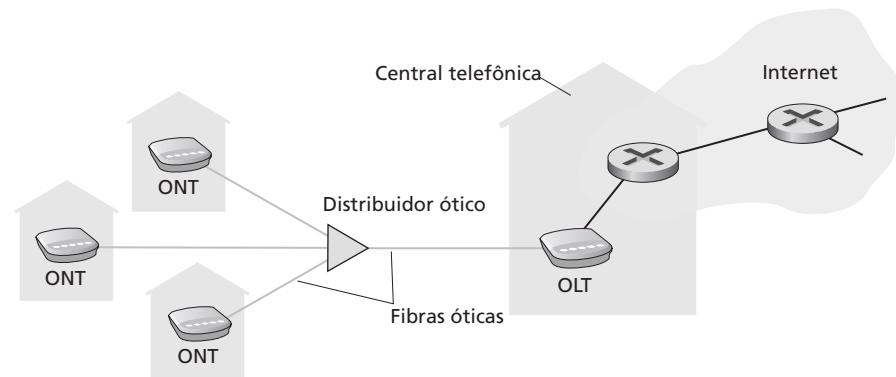


Figura 1.8 Acesso à Internet FTTH

A FTTH pode potencialmente prover taxas de acesso à Internet na faixa de gigabits por segundo. Porém, a maioria dos provedores de FTTH oferecem diferentes taxas, sendo que as mais altas custam muito mais. Atualmente, a maioria dos clientes FTTH preferem taxas de download na faixa de 10 a 20 Mbps e de upload na faixa de 2 a 10 Mbps. Além do acesso à Internet, as fibras ópticas possuem serviços telefônicos tradicionais e de transmissão televisiva.

Ethernet

Nos campi universitários e corporativos, uma rede local (LAN) geralmente é usada para conectar sistemas finais ao roteador da periferia. Embora existam muitos tipos de tecnologia LAN, a Ethernet é, de longe, a tecnologia de acesso mais predominante nas redes universitárias e corporativas. Como mostrado na Figura 1.9, os usuários da Ethernet utilizam par de fios de cobre trançado para se conectarem a um comutador Ethernet, uma tecnologia tratada com mais detalhes no Capítulo 5. O acesso à Ethernet normalmente possui 100 Mbps, enquanto os servidores possuem um acesso de 1 Gbps ou até mesmo 10 Gbps.

WiFi

Está cada vez mais comum as pessoas acessarem a Internet sem fio, seja por um laptop ou por um aparelho portátil, como um iPhone, Blackberry ou o Google phone (veja o quadro "Um conjunto impressionante de sistemas finais da Internet"). Hoje, há dois tipos comuns de acesso à Internet sem fio. Em uma LAN sem fio, os usuários transmitem/recebem pacotes para/de um ponto de acesso que, por sua vez, é conectado à Internet com

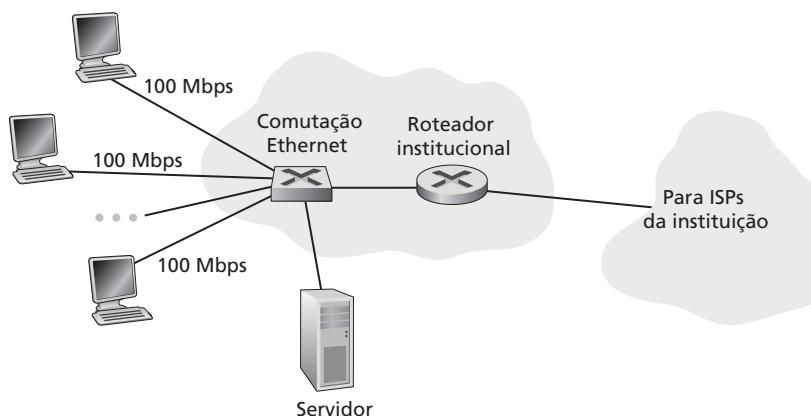


Figura 1.9 Acesso à rede Ethernet

fio. Um usuário LAN sem fio geralmente deve estar no espaço de alguns metros do ponto de acesso. Nas redes de acesso sem fio em amplas áreas, os pacotes são transmitidos para uma estação-base por meio da mesma infraestrutura sem fio utilizada para telefonia celular. Nesse caso, a estação-base é controlada pelo provedor da rede de celular, e o usuário normalmente deve estar dentro de alguns quilômetros da estação-base.

O acesso à LAN sem fio baseado na tecnologia IEEE 802.11, ou seja, WiFi, está presente em todo lugar — universidades, empresas, cafés, aeroportos, residências e, até mesmo, em aviões. A maioria das universidades instalou estações-base por todo o campus, permitindo que os alunos enviem e recebam e-mails ou naveguem na Web de qualquer lugar do campus. Em muitas cidades, é possível ficar na esquina de uma rua e estar dentro da faixa de dez ou vinte estações-base (para um mapa global de estações-base 802.11 que foram descobertas e acessadas por pessoas que apreciam coisas do tipo, veja [wigle.net, 2009]). Como discutido com detalhes no Capítulo 6, atualmente o 802.11 fornece uma taxa de transmissão compartilhada de até 54 Mbps.

Muitas residências unem o acesso residencial banda larga (ou seja, modems a cabo ou DSL) com a tecnologia LAN sem fio a um custo acessível para criar redes residenciais potentes. A Figura 1.10 mostra um esquema de uma típica rede doméstica. Essa rede consiste em um laptop móvel e um computador com fio; uma estação-base (o ponto de acesso sem fio), que se comunica com o computador sem fio; um modem a cabo, fornecendo acesso banda larga à Internet; e um roteador, que interconecta a estação-base e o computador fixo com o modem a cabo. Essa rede permite que os moradores da residência tenham acesso banda larga à Internet com um morador se movimentando da cozinha ao quintal e até os quartos.

Acesso sem fio em longa distância

Ao acessar à Internet através da tecnologia LAN sem fio, é preciso estar dentro de alguns metros do ponto de acesso. Isso é possível para acesso doméstico, em cafés, e, geralmente, dentro e ao redor de um edifício. Mas e se você estiver na praia, no ônibus ou no carro e precisar da Internet? Para o acesso em áreas amplas, os usuários de Internet móvel utilizam uma infraestrutura de telefone celular, acessando estações-base que estão a até 10 quilômetros de distância.

As empresas de telecomunicação têm investido enormemente na assim chamada terceira geração (3G) sem fio, que oferece acesso sem fio em amplas áreas por pacotes comutados a velocidades que ultrapassam 1 Mbps. Hoje, milhões de usuários estão utilizando essas redes para ler e enviar e-mails, navegar na Web e fazer download de música.

WiMAX

Como sempre, existe uma tecnologia potente esperando para destronar esses padrões. O WiMAX [Intel WiMAX, 2009; WiMAX Forum, 2009], também conhecido como IEEE 802.16, é um primo distante do protocolo

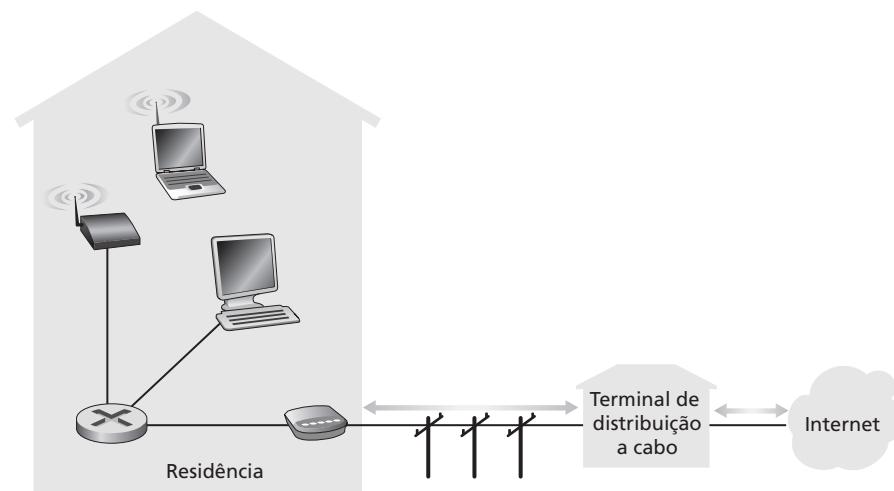


Figura 1.10 Esquema de uma típica rede doméstica

WiFi 802.11 discutido acima. O WiMAX funciona independentemente de uma rede de telefonia celular e promete velocidades de 5 a 10 Mbps ou maiores a distâncias superiores a dez quilômetros. A Sprint-Nextel investiu bilhões de dólares na implementação do WiMAX em 2007 e foi além. As tecnologias WiFi, WiMAX e 3G serão abrangidas em detalhes no Capítulo 6.

1.2.3 Meios físicos

Na subseção anterior, apresentamos uma visão geral de algumas das mais importantes tecnologias de acesso à Internet. Ao descrever essas tecnologias, indicamos também os meios físicos utilizados por elas. Por exemplo, dissemos que o HFC usa uma combinação de cabo de fibra ótica e cabo coaxial e que modems discados de 56 kbps e ADSL usam par de fios de cobre trançado. Dissemos também que redes de acesso móveis usam o espectro de rádio. Nesta subseção damos uma visão geral desses e de outros meios de transmissão comumente empregados na Internet.

Para definir o que significa meio físico, vamos pensar na curta vida de um bit. Considere um bit saindo de um sistema final, transitando por uma série de enlaces e roteadores e chegando a outro sistema final. Esse pobre bit é transmitido muitas e muitas vezes. Primeiramente, o sistema final originador transmite o bit e, logo em seguida, o primeiro roteador da série recebe-o; então, o primeiro roteador envia-o para o segundo roteador e assim por diante. Assim, nosso bit, ao viajar da origem ao destino, passa por uma série de pares transmissores-receptores, que o recebem por meio de ondas eletromagnéticas ou pulsos óticos que se propagam por um meio físico. Com muitos aspectos e formas possíveis, o meio físico não precisa ser obrigatoriamente do mesmo tipo para cada par transmissor-receptor ao longo do caminho. Alguns exemplos de meios físicos são: par de fios de cobre trançado, cabo coaxial, cabo de fibra ótica multimodo, espectro de rádio terrestre e espectro de rádio por satélite. Os meios físicos se enquadram em duas categorias: meios guiados e meios não guiados. Nos meios guiados, as ondas são dirigidas ao longo de um meio sólido, tal como um cabo de fibra ótica, um par de fios de cobre trançado ou um cabo coaxial. Nos meios não guiados, as ondas se propagam na atmosfera e no espaço, como é o caso de uma LAN sem fio ou de um canal digital de satélite.

Porém, antes de examinar as características dos vários tipos de meios, vamos discutir um pouco os seus custos. O custo real de um enlace físico (fio de cobre, cabo de fibra ótica e assim por diante) é em geral relativamente insignificante em comparação a outros custos da rede. Em especial, o custo da mão de obra de instalação do enlace físico pode ser várias vezes maior do que o do material. Por essa razão, muitos construtores instalam pares de fios trançados, fibra ótica e cabo coaxial em todas as salas de um edifício. Mesmo que apenas um dos meios seja usado inicialmente, há uma boa probabilidade de outro meio ser usado no futuro próximo — portanto, poupa-se dinheiro por não ser preciso instalar fiação adicional no futuro.

Par de fios de cobre trançado

O meio de transmissão guiado mais barato e mais comumente usado é o par de fios de cobre trançado, que vem sendo usado há mais de cem anos nas redes de telefonia. Realmente, mais de 99 por cento da fiação que conecta aparelhos telefônicos a centrais locais utilizam pares de fios de cobre trançados. Quase todos nós já vimos um par de fios trançado em casa ou no local de trabalho; esse par constituído de dois fios de cobre isolados, cada um com aproximadamente um milímetro de espessura, enrolados em espiral. Os fios são trançados para reduzir a interferência elétrica de pares semelhantes que estejam próximos. Normalmente, uma série de pares é conjugada dentro de um cabo, isolando-se os pares com blindagem de proteção. Um par de fios constitui um único enlace de comunicação. O par de fios trançado sem blindagem (*unshielded twisted pair* — UTP) é comumente usado em redes de computadores de edifícios, isto é, em LANs. Hoje, as taxas de transmissão de dados para as LANs de pares trançados estão na faixa de 10 Mbps a 1 Gbps. As taxas de transmissão de dados que podem ser alcançadas dependem da bitola do fio e da distância entre transmissor e receptor.

Quando a tecnologia da fibra ótica surgiu na década de 1980, muitos depreciaram o par trançado devido às suas taxas de transmissão de bits relativamente baixas. Alguns até acharam que a tecnologia da fibra ótica o substituiria completamente. Mas o par trançado não desistiu assim tão facilmente. A moderna tecnologia, tal como UTP

categoria 5, pode alcançar taxas de transmissão de dados de 100 Mbps para distâncias de até algumas centenas de metros. E, em distâncias mais curtas, podem ser atingidas taxas ainda maiores. No final, o par trançado firmou-se como a solução dominante para LANs de alta velocidade.

Como vimos na seção sobre redes de acesso, o par trançado também é comumente usado para acesso residencial à Internet. Vimos que a tecnologia do modem discado possibilita taxas de acesso de até 56 kbps com pares trançados. Vimos também que a tecnologia DSL (linha digital de assinante) habilitou usuários residenciais a acessar a Internet em velocidades maiores do que 6 Mbps com pares de fios trançados (quando as residências estão próximas a um modem de ISP).

Cabo coaxial

Como o par trançado, o cabo coaxial é constituído de dois condutores de cobre, porém concêntricos e não paralelos. Com essa configuração, isolamento e blindagem especiais, pode alcançar taxas altas de bits. Cabos coaxiais são muito comuns em sistemas de televisão a cabo. Como já comentamos, recentemente sistemas de televisão a cabo foram acoplados com modems a cabo para prover usuários residenciais de acesso à Internet a velocidades de 1 Mbps ou mais altas. Em televisão a cabo e acesso a cabo à Internet, o transmissor passa o sinal digital para uma banda de frequência específica e o sinal analógico resultante é enviado do transmissor para um ou mais receptores. O cabo coaxial pode ser utilizado como um meio compartilhado guiado. Especificamente, vários sistemas finais podem ser conectados diretamente ao cabo, e todos eles recebem qualquer sinal que seja enviado pelos outros sistemas finais.

Fibras óticas

A fibra ótica é um meio delgado e flexível que conduz pulsos de luz, sendo que cada um desses pulsos representa um bit. Uma única fibra ótica pode suportar taxas de transmissão elevadíssimas, de até dezenas ou mesmo centenas de gigabits por segundo. Fibras óticas são imunes à interferência eletromagnética, têm baixíssima atenuação de sinal até cem quilômetros e são muito difíceis de derivar. Essas características fizeram da fibra ótica o meio preferido para a transmissão guiada de grande alcance, em especial para cabos submarinos. Hoje, muitas redes telefônicas de longa distância dos Estados Unidos e de outros países usam exclusivamente fibras óticas, que também predominam no backbone da Internet. Contudo, o alto custo de equipamentos ópticos — como transmissores, receptores e comutadores — vem impedindo sua utilização para transporte a curta distância, como em LANs ou em redes de acesso residenciais. As velocidades de conexão do padrão Optical Carrier (OC) variam de 51,8 Mbps a 39,8 Gbps; essas especificações são frequentemente denominadas OC-n, em que a velocidade de conexão se iguala a $n \times 51,8$ Mbps. Os padrões usados atualmente incluem OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192 e OC-768. [IEC Optical, 2009; Goralski, 2001; Ramaswami, 1998; Mukherjee, 1997] apresentam uma abordagem de vários aspectos da rede ótica.

Canais de rádio terrestres

Canais de rádio carregam sinais dentro do espectro eletromagnético. São um meio atraente porque sua instalação não requer cabos físicos, podem atravessar paredes, dão conectividade ao usuário móvel e, potencialmente, podem transmitir um sinal a longas distâncias. As características de um canal de rádio dependem significativamente do ambiente de propagação e da distância pela qual o sinal deve ser transmitido. Condições ambientais determinam perda de sinal no caminho e atenuação por efeito de sombra (que reduz a intensidade do sinal quando ele transita por distâncias longas e ao redor/até através de objetos interferentes), atenuação por multivias (devido à reflexão do sinal quando atinge objetos interferentes) e interferência (devido a outros canais de rádio ou a sinais eletromagnéticos).

Canais de rádio terrestres podem ser classificados, de modo geral, em dois grupos: os de pequeno alcance, que funcionam em locais próximos, normalmente abrangendo de dez a algumas centenas de metros, e os de longo alcance, que abrangem dezenas de quilômetros.

As tecnologias LAN sem fio descritas na seção 1.2.2 utilizam canais de rádio local; as tecnologias de acesso em telefone celular utilizam canal de rádio de longo alcance. Abordaremos canais de rádio detalhadamente no Capítulo 6.

Canais de rádio por satélite

Um satélite de comunicação liga dois ou mais transmissores–receptores de micro-ondas baseados na Terra, denominados estações terrestres. Ele recebe transmissões em uma faixa de frequência, gera novamente o sinal usando um repetidor (sobre o qual falaremos a seguir) e o transmite em outra frequência. Satélites podem prover taxas de transmissão na faixa de gigabits por segundo. Dois tipos de satélites são usados para comunicações: satélites geoestacionários e satélites de órbita baixa (LEO).

Os satélites geoestacionários ficam permanentemente sobre o mesmo lugar da Terra. Essa presença estacionária é conseguida colocando-se o satélite em órbita a 36 mil quilômetros acima da superfície terrestre. Essa enorme distância da estação terrestre ao satélite e de seu caminho de volta à estação terrestre traz um substancial atraso de propagação de sinal de 280 milissegundos. Mesmo assim, enlaces por satélite, que podem funcionar a velocidades de centenas de Mbps, são frequentemente usados em redes de telefonia e no backbone da Internet.

Os satélites de baixa órbita são posicionados muito mais próximos da Terra e não ficam permanentemente sobre um único lugar. Eles giram ao redor da Terra (exatamente como a Lua) e podem se comunicar uns com os outros e com estações terrestres. Para prover cobertura contínua em determinada área, é preciso colocar muitos satélites em órbita. Hoje, existem muitos sistemas de comunicação de baixa altitude em desenvolvimento. A página da Web referente à constelação de satélites da Lloyd [Wood, 2009] fornece e coleta informações sobre esses sistemas para comunicações. A tecnologia de satélites de órbita baixa poderá ser utilizada para acesso à Internet no futuro.

1.3 O núcleo da rede

Após termos examinado a periferia da Internet, vamos agora nos aprofundar mais no núcleo da rede — a rede de comutadores de pacote e enlaces que interconectam os sistemas finais da Internet. Os núcleos da rede aparecem destacados em cinza na Figura 1.11.

1.3.1 Comutação de circuitos e comutação de pacotes

Há duas abordagens fundamentais para locomoção de dados através de uma rede de enlaces e comutadores: comutação de circuitos e comutação de pacotes. Em redes de comutação de circuitos, os recursos necessários ao longo de um caminho (buffers, taxa de transmissão de enlaces) para prover comunicação entre os sistemas finais são reservados pelo período da sessão de comunicação entre os sistemas finais. Em redes de comutação de pacotes, esses recursos *não* são reservados; as mensagens de uma sessão usam os recursos por demanda e, como consequência, poderão ter de esperar (isto é, entrar na fila) para conseguir acesso a um enlace de comunicação. Como simples analogia, considere dois restaurantes — um que exige e outro que não exige nem aceita reserva. Se quisermos ir ao restaurante que exige reserva, teremos de passar pelo aborrecimento de telefonar antes de sair de casa. Mas, quando chegarmos lá, poderemos, em princípio, ser imediatamente atendidos e servidos. No restaurante que não exige reserva, não precisaremos nos dar ao trabalho de reservar mesa, porém, quando lá chegarmos, talvez tenhamos de esperar.

As onipresentes redes de telefonia são exemplos de redes de comutação de circuitos. Considere o que acontece quando uma pessoa quer enviar a outra uma informação (por voz ou por fax) por meio de uma rede telefônica. Antes que o remetente possa enviar a informação, a rede precisa estabelecer uma conexão entre o remetente e o destinatário. Essa é uma conexão forte, na qual os comutadores existentes no caminho entre o remetente e o destinatário mantêm o estado dessa conexão. No jargão da telefonia, essa conexão é denominada circuito. Quando a rede estabelece o circuito, também reserva uma taxa de transmissão constante nos enlaces da rede durante o período da conexão. Visto que foi reservada largura de banda para essa conexão remetente-destinatário, o remetente pode transferir dados ao destinatário a uma taxa constante garantida.

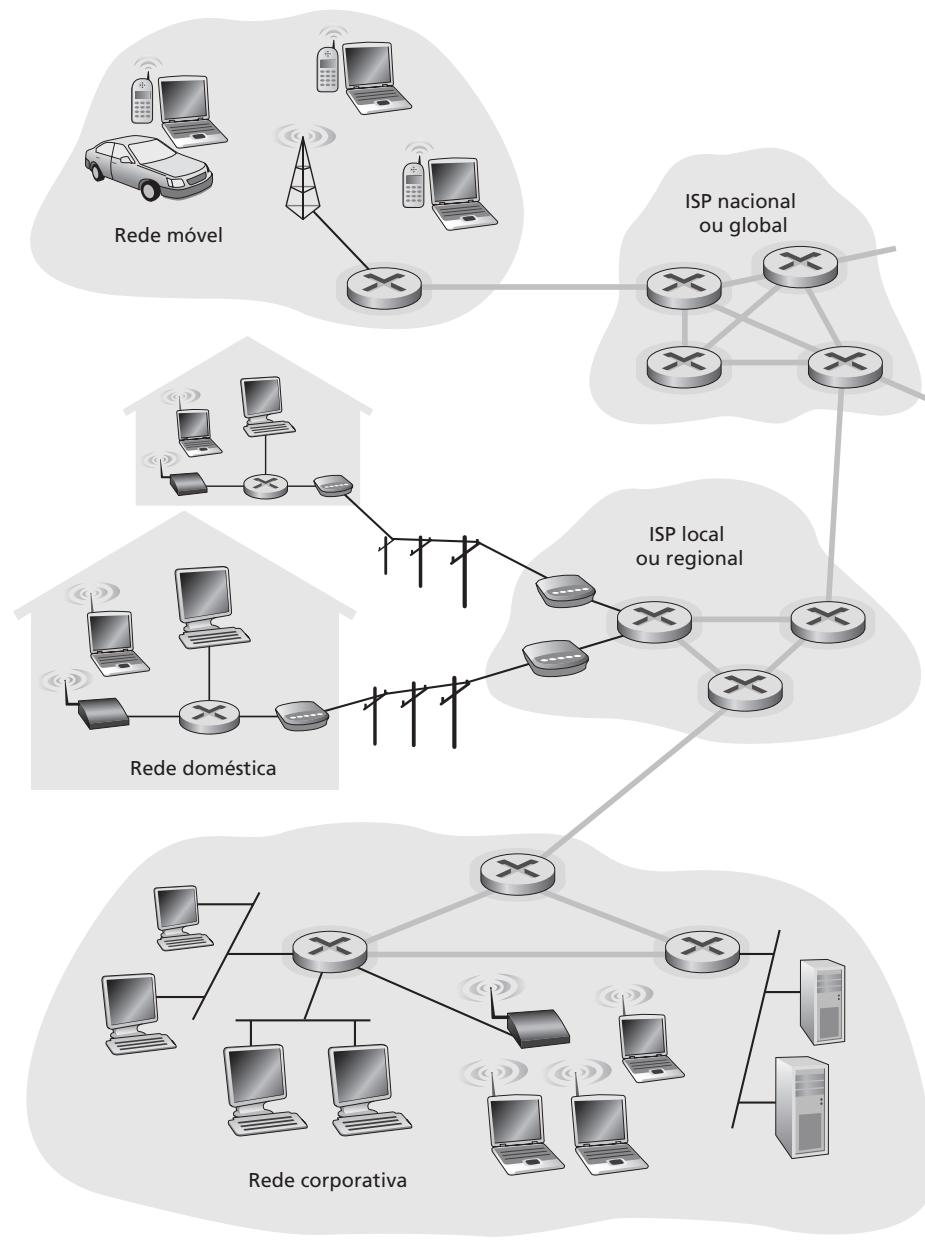


Figura 1.11 O núcleo da rede

Hoje, a Internet é a quintessência das redes de comutação de pacotes. Considere o que ocorre quando um sistema final quer enviar um pacote a outro sistema final pela Internet. Como acontece na comutação de circuitos, o pacote é transmitido por uma série de enlaces de comunicação. Mas, na comutação de pacotes, o pacote é enviado à rede sem reservar nenhuma largura de banda. Se um dos enlaces estiver congestionado porque outros pacotes precisam ser transmitidos pelo enlace ao mesmo tempo, então nosso pacote terá de esperar em um buffer na extremidade de origem do enlace de transmissão e sofrerá um atraso. A Internet faz o *melhor esforço* para entregar os dados prontamente, mas não dá nenhuma garantia.

Nem todas as redes de telecomunicação podem ser classificadas exatamente como redes de comutação de circuitos puras ou redes de comutação de pacotes puras. Não obstante, essa classificação fundamental em redes de comutação de pacotes e de comutação de circuitos é um excelente ponto de partida para a compreensão da tecnologia de redes de telecomunicação.

Comutação de circuitos

Este livro é sobre redes de computadores, Internet e comutação de pacotes, e não sobre redes telefônicas e comutação de circuitos. Mesmo assim, é importante entender por que a Internet e outras redes de computadores usam comutação de pacotes, e não a tecnologia mais tradicional de comutação de circuitos utilizada nas redes telefônicas. Por essa razão, examinaremos agora resumidamente a comutação de circuitos.

A Figura 1.12 ilustra uma rede de comutação de circuitos. Nessa rede, os quatro comutadores de circuitos estão interconectados por quatro enlaces. Cada um desses enlaces tem n circuitos, de modo que cada um pode suportar n conexões simultâneas. Cada um dos sistemas finais (por exemplo, PCs e estações de trabalho) está conectado diretamente a um dos circuitos. Quando dois sistemas finais querem se comunicar, a rede estabelece uma conexão fíme fíme dedicada entre os dois sistemas finais. (É claro que também são possíveis chamadas em conferência entre mais de dois equipamentos. Mas, para simplificar, por enquanto vamos supor que haja somente dois sistemas finais para cada conexão.) Assim, para que o sistema final A envie mensagens ao sistema final B, a rede deve primeiramente reservar um circuito em cada um dos dois enlaces. Como cada enlace tem n circuitos, para cada enlace usado pela conexão fíme fíme, esta fica com uma fração $1/n$ da largura de banda do enlace durante o período da conexão.

Multiplexação em redes de comutação de circuitos

Um circuito é implementado em um enlace por multiplexação por divisão de frequência (*frequency-division multiplexing* — FDM) ou por multiplexação por divisão de tempo (*time-division multiplexing* — TDM). Com FDM, o espectro de frequência de um enlace é compartilhado entre as conexões estabelecidas através desse enlace. Especificamente, o enlace reserva uma banda de frequência para cada conexão durante o período da ligação. Em redes telefônicas, a largura dessa banda de frequência normalmente é 4 kHz (isto é, 4 mil Hertz ou 4 mil ciclos por segundo). Estações de rádio FM também usam FDM para compartilhar o espectro de frequência entre 88 MHz e 108 MHz, sendo atribuída para cada estação uma banda de frequência específica.

Em um enlace TDM, o tempo é dividido em quadros de duração fixa, e cada quadro é dividido em um número fixo de compartimentos (*slots*). Quando estabelece uma conexão por meio de um enlace, a rede dedica à conexão um compartimento de tempo em cada quadro. Esses compartimentos são reservados para o uso exclusivo dessa conexão, e um dos compartimentos de tempo (em cada quadro) fica disponível para transmitir os dados dela.

A Figura 1.13 ilustra as técnicas FDM e TDM para um enlace de rede específico que suporta até quatro circuitos. Para FDM, o domínio de frequência é segmentado em quatro faixas, cada uma com largura de banda de 4 kHz. Para TDM, o domínio de tempo é segmentado em quadros, cada um com quatro compartimentos

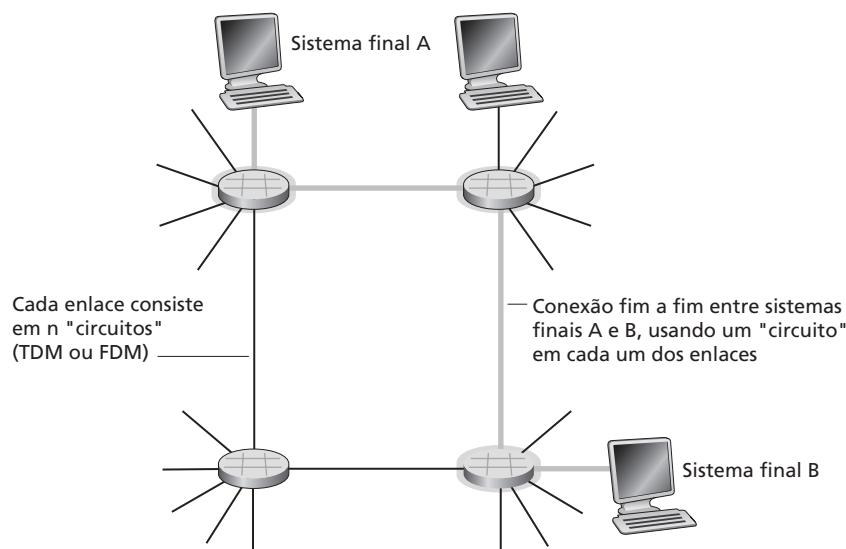


Figura 1.12 Uma rede simples de comutação de circuitos composta de quatro comutadores de circuito e quatro enlaces

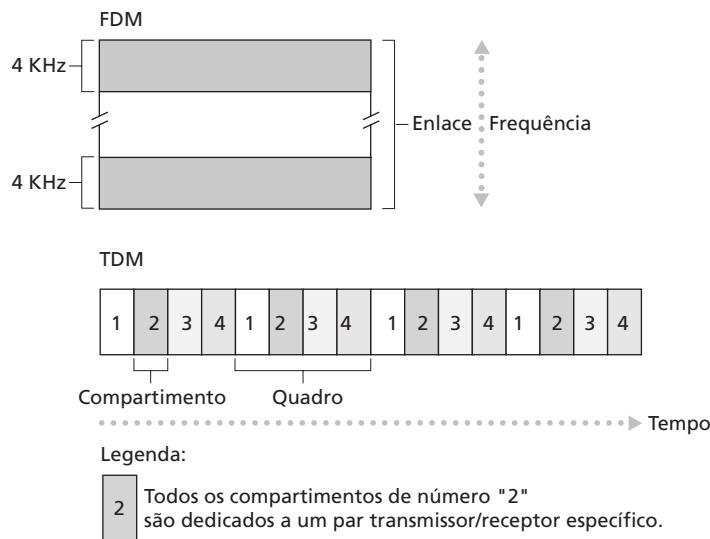


Figura 1.13 Com FDM, cada circuito dispõe continuamente de uma fração da largura de banda. Com TDM, cada circuito dispõe de toda a largura de banda periodicamente, durante breves intervalos de tempo (isto é, durante compartimentos de tempo)

de tempo; a cada circuito é designado o mesmo compartimento dedicado nos quadros sucessivos TDM. Para TDM, a taxa de transmissão de um circuito é igual à taxa do quadro multiplicada pelo número de bits em um compartimento. Por exemplo, se o enlace transmite 8 mil quadros por segundo e cada compartimento consiste em 8 bits, então a taxa de transmissão de um circuito é 64 kbps.

Os defensores da comutação de pacotes sempre argumentaram que comutação de circuitos é desperdício, porque os circuitos dedicados ficam ociosos durante períodos de silêncio. Por exemplo, quando um dos participantes de uma conversa telefônica para de falar, os recursos ociosos da rede (bandas de frequências ou compartimentos nos enlaces ao longo da rota da conexão) não podem ser usados por outras conexões em curso. Para um outro exemplo de como esses recursos podem ser subutilizados, considere um radiologista que usa uma rede de comutação de circuitos para acessar remotamente uma série de radiografias. Ele estabelece uma conexão, requisita uma imagem, examina-a e, em seguida, solicita uma nova imagem. Recursos de rede são atribuídos à conexão, mas não utilizados (isto é, são desperdiçados) durante o período em que o radiologista examina a imagem. Defensores da comutação de pacotes também gostam de destacar que estabelecer circuitos fim a fim e reservar larguras de banda fim a fim é complicado e exige softwares complexos de sinalização para coordenar a operação dos comutadores ao longo do caminho fim a fim.

Antes de encerrarmos esta discussão sobre comutação de circuitos, vamos examinar um exemplo numérico que deverá esclarecer melhor o assunto. Vamos considerar o tempo que levamos para enviar um arquivo de 640 kbytes do sistema final A ao sistema final B por uma rede de comutação de circuitos. Suponha que todos os enlaces da rede usem TDM de 24 compartimentos e tenham uma taxa de 1,536 Mbps. Suponha também que um circuito fim a fim leva 500 milissegundos para ser ativado antes que A possa começar a transmitir o arquivo. Em quanto tempo o arquivo será enviado? Cada circuito tem uma taxa de transmissão de $(1,536 \text{ Mbps})/24 = 64 \text{ kbps}$; portanto, demorará $(640 \text{ kbytes})/64 \text{ kbps} = 10 \text{ segundos}$ para transmitir o arquivo. A esses 10 segundos adicionamos o tempo de ativação do circuito, resultando 10,5 segundos para o envio. Observe que o tempo de transmissão é independente do número de enlaces: o tempo de transmissão seria 10 segundos se o circuito fim a fim passasse por um ou por uma centena de enlaces. (O atraso real fim a fim também inclui um atraso de propagação; ver Seção 1.4.)

Comutação de pacotes

Aplicações distribuídas trocam mensagens ao desempenhar suas tarefas. Mensagens podem conter qualquer característica que o projetista do protocolo queira. Podem desempenhar uma função de controle (por exemplo, as

mensagens ‘oi’ no nosso exemplo de comunicação entre pessoas) ou podem conter dados, tal como uma mensagem de e-mail, uma imagem JPEG ou um arquivo de áudio MP3. Em redes de computadores modernas, o originador fragmenta mensagens longas em porções de dados menores denominadas pacotes. Entre origem e destino, cada um desses pacotes percorre enlaces de comunicação e comutadores de pacotes (há dois tipos principais de comutadores de pacotes: roteadores e comutadores de camada de enlace). Pacotes são transmitidos por cada enlace de comunicação a uma taxa igual à de transmissão *total* do enlace.

A maioria dos comutadores de pacotes armazena e reenvia os pacotes nas entradas dos enlaces, numa técnica conhecida como armazena-e-reenvia e de acordo com a qual o comutador deve receber o pacote inteiro antes de poder começar a transmitir o primeiro bit do pacote para o enlace de saída. Assim, comutadores de pacotes apresentam um atraso de armazenagem e reenvio na entrada de cada enlace ao longo da rota do pacote. Vamos considerar agora quanto tempo demora para enviar um pacote de L bits de um sistema final para outro por uma rede de comutação de pacotes. Suponha que haja Q enlaces entre os dois sistemas finais, cada um com taxa de Rbps. Admita que atrasos de fila e de propagação fim a fim sejam desprezíveis e que não haja estabelecimento de conexão. Primeiramente o pacote deve ser transmitido para o primeiro enlace que se origina do sistema final A, o que leva L/R segundos. Em seguida, ele tem de ser transmitido por cada um dos $Q - 1$ enlaces remanescentes, isto é, deve ser armazenado e reenviado $Q - 1$ vezes. Portanto, o atraso total é QL/R .

A cada comutador de pacotes estão ligados vários enlaces. Para cada um destes, o comutador de pacotes tem um buffer de saída (também denominado fila de saída), que armazena pacotes prestes a serem enviados pelo roteador para aquele enlace. Os buffers de saída desempenham um papel fundamental na comutação de pacotes. Se um pacote que está chegando precisa ser transmitido por um enlace, mas o encontra ocupado com a transmissão de outro pacote, deve aguardar no buffer de saída. Desse modo, além dos atrasos de armazenagem e reenvio, os pacotes sofrem atrasos de fila no buffer de saída. Esses atrasos são variáveis e dependem do grau de congestionamento da rede. Como o espaço do buffer é finito, um pacote que está chegando pode encontrá-lo completamente lotado de outros pacotes que estão esperando transmissão. Nesse caso, ocorrerá uma perda de pacote — um pacote que está chegando ou um dos que já estão na fila é descartado. Voltando à analogia do restaurante apresentada anteriormente nesta seção, o atraso de fila é análogo ao tempo gasto no bar do restaurante esperando uma mesa, enquanto a perda de pacote é análoga a ouvir do garçom que devemos desistir porque já há muitas pessoas ali.

A Figura 1.14 ilustra uma rede simples de comutação de pacotes. Nessa figura e nas subsequentes, pacotes são representados por placas tridimensionais. A largura de uma placa representa o número de bits no pacote. Nessa figura todos os pacotes têm a mesma largura, portanto, o mesmo tamanho. Suponha que os sistemas finais A e B estejam enviando pacotes ao sistema final E. Os sistemas finais A e B primeiramente enviarão seus pacotes por

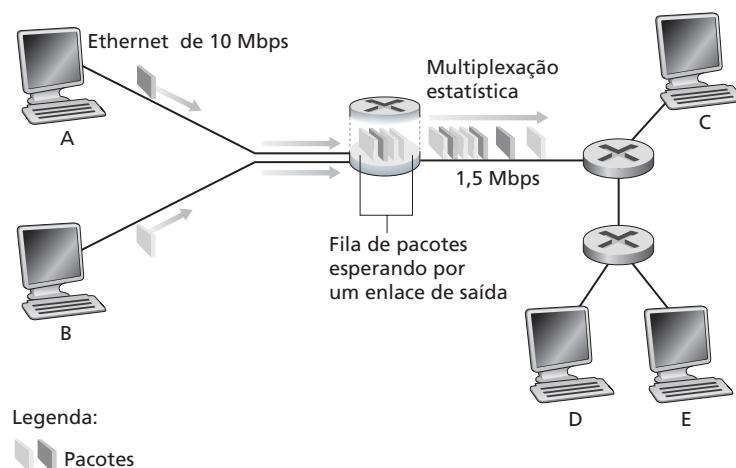


Figura 1.14 Comutação de pacotes

enlaces Ethernet de 10 Mbps até o primeiro comutador de pacotes, que vai direcioná-los para o enlace de 1,5 Mbps. Se a taxa de chegada de pacotes ao comutador for maior do que a taxa com que o comutador pode reenviar pacotes pelo enlace de saída de 1,5 Mbps, ocorrerá congestionamento, pois os pacotes formarão uma fila no buffer de saída do enlace antes de ser transmitidos para o enlace. Vamos analisar esse atraso mais detalhadamente na Seção 1.4.

Comutação de pacotes versus comutação de circuitos: multiplexação estatística

Agora que já descrevemos comutação de pacotes e comutação de circuitos, vamos comparar as duas. Opositores da comutação de pacotes frequentemente argumentam que ela não é adequada para serviços de tempo real (por exemplo, ligações telefônicas e videoconferência) por causa de seus atrasos fíns variáveis e imprevisíveis (que se devem principalmente a atrasos de fila variáveis e imprevisíveis). Defensores da comutação de pacotes argumentam que (1) ela oferece melhor compartilhamento de banda do que comutação de circuitos e (2) sua implementação é mais simples, mais eficiente e mais barata do que a implementação de comutação de circuitos. Uma discussão interessante sobre comutação de pacotes e comutação de circuitos pode ser encontrada em [Molinero-Fernandez, 2002]. De modo geral, quem não gosta de perder tempo fazendo reserva de mesa em restaurantes prefere comutação de pacotes à comutação de circuitos.

Por que a comutação de pacotes é mais eficiente? Vamos examinar um exemplo simples. Suponha que usuários compartilhem um enlace de 1 Mbps. Suponha também que cada usuário alterne períodos de atividade, quando gera dados a uma taxa constante de 100 kbps, e de inatividade, quando não gera dados. Suponha ainda que o usuário esteja ativo apenas 10 por cento do tempo (e fique ocioso, tomando cafezinho, durante os restantes 90 por cento). Com comutação de circuitos, devem ser reservados 100 kbps para *cada usuário* durante todo o tempo. Por exemplo, com TDM, se um quadro de um segundo for dividido em 10 compartimentos de tempo de 100 milissegundos cada, então seria alocado um compartimento de tempo por quadro a cada usuário.

Desse modo, o enlace pode suportar somente 10 (= 1 Mbps/100 kbps) usuários simultaneamente. Com comutação de pacotes, a probabilidade de haver um usuário específico ativo é 0,1 (isto é, 10 por cento). Se houver 35 usuários, a probabilidade de haver 11 ou mais usuários ativos simultaneamente é aproximadamente 0,0004. (O Problema 7 dos Exercícios de Fixação demonstra como essa probabilidade é calculada.) Quando houver dez ou menos usuários ativos simultaneamente (a probabilidade de isso acontecer é 0,9996), a taxa agregada de chegada de dados é menor ou igual a 1 Mbps, que é a taxa de saída do enlace. Assim, quando houver dez ou menos usuários ativos, pacotes de usuários fluirão pelo enlace essencialmente sem atraso, como é o caso na comutação de circuitos. Quando houver mais de dez usuários ativos simultaneamente, a taxa agregada de chegada de pacotes excederá a capacidade de saída do enlace, e a fila de saída começará a crescer. (E continuará a crescer até que a velocidade agregada de entrada caia novamente para menos de 1 Mbps, ponto em que o comprimento da fila começará a diminuir.) Como a probabilidade de haver mais de dez usuários ativos simultaneamente é ínfima nesse exemplo, a comutação de pacotes apresenta, essencialmente, o mesmo desempenho da comutação de circuitos, *mas o faz para mais de três vezes o número de usuários*.

Vamos considerar agora um segundo exemplo simples. Suponha que haja dez usuários e que um deles repentinamente gere mil pacotes de mil bits, enquanto os outros nove permanecem inativos e não geram pacotes. Com comutação de circuitos TDM de dez compartimentos de tempo por quadro, e cada quadro consistindo em mil bits, o usuário ativo poderá usar somente seu único compartimento por quadro para transmitir dados, enquanto os nove compartimentos restantes em cada quadro continuarão ociosos. Dez segundos se passarão antes que todo o milhão de bits de dados do usuário ativo seja transmitido. No caso da comutação de pacotes, o usuário ativo poderá enviá-los continuamente à taxa total de 1 Mbps, visto que não haverá outros usuários gerando pacotes que precisem ser multiplexados com os pacotes do usuário ativo. Nesse caso, todos os dados do usuário ativo serão transmitidos dentro de 1 segundo.

Os exemplos acima ilustram duas maneiras pelas quais o desempenho da comutação de pacotes pode ser superior à da comutação de circuitos. Também destacam a diferença crucial entre as duas formas de compartilhar a taxa de transmissão de um enlace entre várias correntes de bits. Comutação de circuitos aloca previamente a utilização do enlace de transmissão independentemente de demanda, com desperdício de tempo de enlace desnecessário alocado e não utilizado. Comutação de pacotes, por outro lado, aloca utilização de enlace *por*

demandada. A capacidade de transmissão do enlace será compartilhada pacote por pacote somente entre usuários que tenham pacotes que precisam ser transmitidos pelo enlace. Tal compartilhamento de recursos por demanda (e não por alocação prévia) às vezes é denominado multiplexação estatística de recursos.

Embora tanto a comutação de pacotes quanto a comutação de circuitos predominem nas redes de telecomunicação de hoje, a tendência é, sem dúvida, a comutação de pacotes. Até mesmo muitas das atuais redes de telefonia de comutação de circuitos estão migrando lentamente para a comutação de pacotes. Em especial, redes telefônicas frequentemente usam comutação de pacotes na parte cara de uma chamada telefônica para o exterior, isto é, na parte que não é processada em território nacional.

1.3.2 Como os pacotes percorrem as redes de comutadores de pacotes?

Anteriormente dissemos que um roteador conduz um pacote que chega em um de seus enlaces de comunicação para outro enlace. Mas como o roteador determina o enlace que deve conduzir o pacote? Na verdade, isso é feito de diferentes maneiras por diferentes tipos de rede de computadores. Neste capítulo introdutório, descreveremos uma abordagem popular, a saber, a abordagem empregada pela Internet.

Na Internet, cada pacote que atravessa a rede contém o seu endereço de destino em seu cabeçalho. Como os endereços postais, esse endereço possui uma estrutura hierárquica. Quando um pacote chega à um roteador na rede, o roteador examina uma parte do endereço de destino do pacote e conduz o pacote a um roteador adjacente. Especificamente falando, cada roteador possui uma base de encaminhamento que mapeia o endereço de destino (ou partes desse endereço) para enlaces de saída. Quando um pacote chega ao roteador, este examina o endereço e busca sua base utilizando esse endereço de destino para encontrar o enlace de saída apropriado. O roteador, então, direciona o pacote ao enlace de saída.

Vimos que um roteador usa um endereço de destino do pacote para indexar uma base de encaminhamento e determinar o enlace de saída apropriado. Mas essa afirmação traz ainda outra questão: como as bases de encaminhamento se configuram? Elas são configuradas manualmente em cada roteador ou a Internet utiliza um procedimento mais automático? Essa questão será estudada mais profundamente no Capítulo 4. Mas para aguçar seu apetite, observe que a Internet possui um número especial de protocolos de roteamento que são utilizados para configurar automaticamente as bases de encaminhamento. Um protocolo de roteamento pode, por exemplo, determinar o caminho mais curto de cada roteador a cada destino e utilizar os resultados desse caminho para configurar as bases de encaminhamento nos roteadores.

O processo de roteamento fim a fim é semelhante a um motorista que não quer fazer uso do mapa, preferindo pedir informações. Por exemplo, suponha que Joe vai dirigir da Filadélfia para 156 Lakeside Drive, em Orlando, Flórida. Primeiro, Joe vai ao posto de gasolina de seu bairro e pergunta como chegar a 156 Lakeside Drive, em Orlando, Flórida. O frentista do posto extrai a palavra Flórida do endereço e diz que Joe precisa pegar a interestadual I-95 South, cuja entrada fica ao lado do posto. Ele também diz a Joe para pedir outras informações assim que chegar a Flórida. Então, Joe pega a I-95 South até chegar a Jacksonville, na Flórida, onde pede mais informações a outro frentista. Este extrai a palavra Orlando do endereço e diz a Joe para continuar na I-95 até Daytona Beach, e lá se informar novamente. Em Daytona Beach, outro frentista também extrai a palavra Orlando do endereço e pede para que ele pegue a I-4 diretamente para Orlando. Joe segue suas orientações e chega a uma saída para Orlando. Ele vai até outro posto de gasolina, e desta vez o atendente extrai a palavra Lakeside Drive do endereço e diz a ele qual estrada seguir para Lakeside Drive. Assim que Joe chega a Lakeside Drive, ele pergunta a uma criança de bicicleta como chegar a seu destino. A criança extrai o número 156 do endereço e aponta para a casa. Joe finalmente chega a seu destino final.

Na analogia acima, os frentistas dos postos e a criança na bicicleta são semelhantes aos roteadores. As bases de encaminhamento, que estão no cérebro deles, foram configuradas por anos de experiência.

Você gostaria de ver a rota fim a fim que os pacotes realizam na Internet? Convidamos você a colocar a mão na massa e interagir com o programa Traceroute, visitando o site <http://www.traceroute.org>. (Para detalhes sobre o Traceoute, veja a Seção 1.4.)

1.3.3 ISPs e backbones da Internet

Vimos anteriormente que sistemas finais (PCs de usuários, PDAs, servidores Web, servidores de correio eletrônico e assim por diante) conectam-se à Internet por meio de um provedor local. O provedor pode fornecer uma conectividade tanto com fio como sem fio, utilizando um conjunto de tecnologias de acesso, incluindo DSL, modem de cabo, FTTH, Wi-Fi, telefone celular e WiMAX. Observe que o provedor local não precisa ser uma operadora de telefonia ou uma empresa de TV a cabo: pode ser, por exemplo, uma universidade (que oferece acesso à Internet para os alunos, a equipe e o corpo docente) ou uma empresa (que oferece acesso para seus funcionários). Mas conectar usuários finais e provedores de conteúdo a redes de acesso é apenas uma pequena peça do quebra-cabeça que é conectar as centenas de milhões de usuários e centenas de milhares de redes que compõem a Internet. A Internet é uma *rede de redes* — entender essa frase é a chave para resolver esse jogo.

Na Internet pública, redes de acessos situadas na borda da Internet são conectadas ao restante da rede segundo uma hierarquia de níveis de ISPs, como mostra a Figura 1.15. Os ISPs de acesso estão no nível mais baixo dessa hierarquia. No topo dela está um número relativamente pequeno de ISPs denominados ISPs de nível 1. Sob muitos aspectos, um ISP de nível 1 é igual a qualquer rede — tem enlaces e roteadores e está conectado a outras redes. Mas, considerando-se outros aspectos, ISPs de nível 1 são especiais. As velocidades de seus enlaces muitas vezes alcançam 622 Mbps ou mais, tendo os maiores deles enlaces na faixa de 2,5 a 10 Gbps. Consequentemente, seus roteadores são capazes de transmitir pacotes a taxas extremamente altas. ISPs de nível 1 também apresentam as seguintes características:

- conectam-se diretamente a *cada* um dos outros ISPs de nível 1;
- conectam-se a um grande número de ISPs de nível 2 e a outras redes clientes;
- cobertura internacional.

Esses ISPs também são conhecidos como redes de backbone da Internet. Citamos, como exemplos, Sprint, MCI (anteriormente UUNet/WorldCom), AT&T, Level3, Qwest e Cable & Wireless. Em meados de 2002, a WorldCom era, de longe, o maior ISP de nível 1 existente — mais de duas vezes maior do que seu rival mais próximo, segundo diversas medições de tamanho [Telegraphy, 2002]. O interessante é que nenhum grupo san-

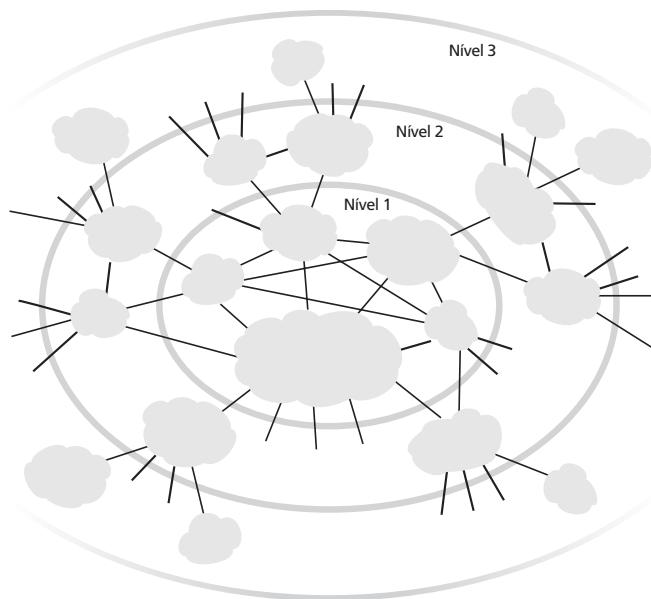


Figura 1.15 Interconexão de ISPs

ciona, oficialmente, seu status de nível 1. Como se costuma dizer, se você tiver de perguntar se é um membro de um grupo, provavelmente não é.

Um ISP de nível 2 normalmente tem alcance regional ou nacional e (o que é importante) conecta-se apenas a uns poucos ISPs de nível 1 (veja Figura 1.15).

Assim, para alcançar uma grande parcela da Internet global, um ISP de nível 2 tem de direcionar o tráfego por um dos ISPs de nível 1 com o qual está conectado. Um ISP de nível 2 é denominado um cliente do ISP de nível 1 com o qual está conectado, que, por sua vez, é denominado provedor de seu cliente. Muitas empresas de grande porte e instituições conectam suas redes corporativas diretamente a um provedor de nível 1 ou 2, tornando-se, desse modo, cliente daquele ISP. O provedor ISP cobra uma tarifa de seu cliente, que normalmente depende da taxa de transmissão do enlace que interliga ambos. Uma rede de nível 2 também pode preferir conectar-se diretamente a outras redes de mesmo nível, caso em que o tráfego pode fluir entre as duas sem ter de passar por uma rede de nível 1. Abaixo dos ISPs de nível 2 estão os de níveis mais baixos que se conectam à Internet por meio de um ou mais ISPs de nível 2 e, na parte mais baixa da hierarquia, estão os ISPs de acesso. Para complicar ainda mais as coisas, alguns provedores de nível 1 também são provedores de nível 2 (isto é, integrados verticalmente) e vendem acesso para Internet diretamente a usuários finais e provedores de conteúdo, bem como os ISPs de níveis mais baixos. Quando dois ISPs estão ligados diretamente um ao outro são denominados pares (peers) um do outro. Um estudo interessante [Subramanian, 2002] procura definir mais exatamente a estrutura em níveis da Internet estudando sua topologia em termos de relacionamentos cliente-provedor e entre parceiros (peer-peer). Para uma discussão mais clara sobre esses relacionamentos, veja Van der Berg, 2008.

Dentro da rede de um ISP, os pontos em que ele se conecta a outros ISPs (seja abaixo, acima ou no mesmo nível na hierarquia) são conhecidos como pontos de presença (*points of presence* — POPs). Um POP é simplesmente um grupo de um ou mais roteadores na rede do ISP com os quais roteadores em outros ISPs, ou em redes pertencentes a clientes do ISP, podem se conectar. Um provedor de nível 1 normalmente tem muitos POPs espalhados por diferentes localidades geográficas em sua rede e várias redes clientes e outros ISPs ligados a cada POP. Para se conectar ao POP de um provedor, uma rede cliente normalmente aluga um enlace de alta velocidade de um provedor de telecomunicações de terceiros e conecta um de seus roteadores diretamente a um roteador no POP do provedor. Dois ISPs de nível 1 também podem formar um par conectando um par de POPs, cada um proveniente de um dos dois ISPs. Além disso, dois ISPs podem ter vários pontos de emparelhamento conectando-se um ao outro em dois ou mais de pares de POPs.

Resumindo, a topologia da Internet é complexa, consistindo em dezenas de ISPs de níveis 1 e 2 e milhares de ISPs de níveis mais baixos. A cobertura dos ISPs é bastante diversificada; alguns abrangem vários continentes e oceanos e outros se limitam a pequenas regiões do mundo. Os ISPs de níveis mais baixos conectam-se a ISPs de níveis mais altos e estes (normalmente) se interconectam uns com os outros. Usuários e provedores de conteúdo são clientes de ISPs de níveis mais baixos e estes são clientes de ISPs de níveis mais altos.

1.4 Atraso, perda e vazão em redes de comutação de pacotes

Na Seção 1.1 dissemos que a Internet pode ser vista como uma infraestrutura que fornece serviços a aplicações distribuídas que são executadas nos sistemas finais. De modo ideal, gostaríamos que os serviços da Internet transferissem tantos dados quanto desejamos entre dois sistemas finais, instantaneamente, sem nenhuma perda. Infelizmente, esse é um objetivo elusivo, algo inalcançável. Ao contrário disso, as redes de computador, necessariamente, restringem a vazão (a quantidade de dados por segundo que podem ser transferidos) entre sistemas finais, apresentam atrasos entre sistemas finais e podem perder pacotes. Por um lado, infelizmente as leis físicas da realidade introduzem atraso e perda, bem como restringem a vazão. Por outro lado, como as redes de computadores possuem esses problemas, existem muitas questões fascinantes sobre como lidar com eles — mais do que questões suficientes para frequentar um curso de rede de computadores e motivar centenas de teses de doutorado! Nesta seção, começaremos a examinar e quantificar atraso, perda e vazão em redes de computadores.

1.4.1 Uma visão geral de atraso em redes de comutação de pacotes

Lembre-se de que um pacote começa em um sistema final (a origem), passa por uma série de roteadores e termina sua jornada em um outro sistema final (o destino). Quando um pacote viaja de um nó (sistema final ou roteador) ao nó subsequente (sistema final ou roteador), sofre, ao longo desse caminho, diversos tipos de atraso em *cada* nó existente no caminho. Os mais importantes deles são o atraso de processamento nodal, o atraso de fila, o atraso de transmissão e o atraso de propagação; juntos, eles se acumulam para formar o atraso nodal total. Para entender a fundo a comutação de pacotes e redes de computadores, é preciso entender a natureza e a importância desses atrasos.

Tipos de atraso

Vamos examinar esses atrasos no contexto da Figura 1.16. Como parte de sua rota fim a fim entre origem e destino, um pacote é enviado do nó anterior por meio do roteador A até o roteador B. Nossa meta é caracterizar o atraso nodal no roteador A. Note que este tem um enlace de saída que leva ao roteador B. Esse enlace é precedido de uma fila (também conhecida como buffer). Quando o pacote chega ao roteador A, vindo do nó anterior, o roteador examina o cabeçalho do pacote para determinar o enlace de saída apropriado e então o direciona ao enlace. Nesse exemplo, o enlace de saída para o pacote é o que leva ao roteador B. Um pacote pode ser transmitido por um enlace somente se não houver nenhum outro pacote sendo transmitido por ele e se não houver outros pacotes à sua frente na fila. Se o enlace estiver ocupado, ou com pacotes à espera, o pacote recém-chegado entrará na fila.

Atraso de processamento

O tempo requerido para examinar o cabeçalho do pacote e determinar para onde direcioná-lo é parte do atraso de processamento, que pode também incluir outros fatores, como o tempo necessário para verificar os erros em bits existentes no pacote que ocorreram durante a transmissão dos bits desde o nó anterior ao roteador A. Atrasos de processamento em roteadores de alta velocidade normalmente são da ordem de microssegundos, ou menos. Depois desse processamento nodal, o roteador direciona o pacote à fila que precede o enlace com o roteador B. (No Capítulo 4, estudaremos os detalhes da operação de um roteador.)

Atraso de fila

O pacote sofre um atraso de fila enquanto espera para ser transmitido no enlace. O tamanho desse atraso para um pacote específico dependerá da quantidade de outros pacotes que chegarem antes e que já estiverem na fila esperando pela transmissão. Se a fila estiver vazia, e nenhum outro pacote estiver sendo transmitido naquele momento, então o tempo de fila de nosso pacote será zero. Por outro lado, se o tráfego estiver pesado e houver muitos pacotes também esperando para ser transmitidos, o atraso de fila será longo. Em breve, veremos que o número de pacotes que um determinado pacote provavelmente encontrará ao chegar é uma função da intensi-

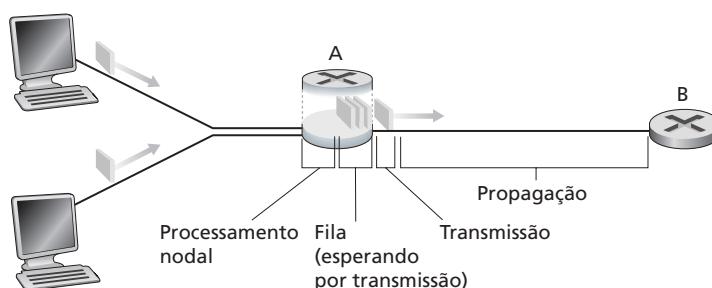


Figura 1.16 O atraso nodal no roteador A

dade e da natureza do tráfego que está chegando à fila. Na prática, atrasos de fila podem ser da ordem de micro a milissegundos.

Atraso de transmissão

Admitindo-se que pacotes são transmitidos segundo a estratégia de “o primeiro a chegar será o primeiro a ser processado”, como é comum em redes de comutação de pacotes, nosso pacote somente poderá ser transmitido depois que todos aqueles que chegaram antes tenham sido enviados. Denominemos o tamanho do pacote como L bits e a velocidade de transmissão do enlace do roteador A ao roteador B como R bits/s. A velocidade R é determinada pela velocidade de transmissão do enlace ao roteador B. Por exemplo, para um enlace Ethernet de 10 Mbps, a velocidade é $R \cdot 10$ Mbps; para um enlace Ethernet de 100 Mbps, a velocidade é $R \cdot 100$ Mbps. O atraso de transmissão (também denominado atraso de armazenamento e reenvio, como discutimos na Seção 1.3) é L/R . Esta é a quantidade de tempo requerida para empurrar (isto é, transmitir) todos os bits do pacote para o enlace. Na prática, atrasos de transmissão são comumente da ordem de micro a milissegundos.

Atraso de propagação

Assim que é lançado no enlace, um bit precisa se propagar até o roteador B. O tempo necessário para propagar o bit desde o início do enlace até o roteador B é o atraso de propagação. O bit se propaga à velocidade de propagação do enlace, a qual depende do meio físico do enlace (isto é, fibra ótica, par de fios de cobre trançado e assim por diante) e está na faixa de

$$2 \cdot 10^8 \text{ m/s} \text{ a } 3 \cdot 10^8 \text{ m/s}$$

que é igual à velocidade da luz, ou um pouco menor. O atraso de propagação é a distância entre dois roteadores dividida pela velocidade de propagação. Isto é, o atraso de propagação é d/s , onde d é a distância entre o roteador A e o roteador B, e s é a velocidade de propagação do enlace. Assim que o último bit do pacote se propagar até o nó B, ele e todos os outros bits precedentes do pacote serão armazenados no roteador B. Então, o processo inteiro continua, agora com o roteador B executando a retransmissão. Em redes WAN, os atrasos de propagação são da ordem de milissegundos.

Comparação entre atrasos de transmissão e de propagação

Os principiantes na área de redes de computadores às vezes têm dificuldade para entender a diferença entre atraso de transmissão e atraso de propagação. A diferença é sutil, mas importante. O atraso de transmissão é a quantidade de tempo requerida para o roteador empurrar o pacote para fora; é uma função do comprimento do pacote e da taxa de transmissão do enlace, mas nada tem a ver com a distância entre os dois roteadores. O atraso de propagação, por outro lado, é o tempo que leva para um bit se propagar de um roteador até o seguinte; é uma função da distância entre os dois roteadores, mas nada tem a ver com o comprimento do pacote ou com a taxa de transmissão do enlace.

Podemos esclarecer melhor as noções de atrasos de transmissão e de propagação com uma analogia. Considere uma rodovia que tenha um posto de pedágio a cada 100 quilômetros, como mostrado na Figura 1.17. Imagine que os trechos da rodovia entre os postos de pedágio sejam enlaces e que os postos de pedágio sejam roteadores. Suponha que os carros trafeguem (isto é, se propaguem) pela rodovia a uma velocidade de 100 km/h (isto é, quando o carro sai de um posto de pedágio, acelera instantaneamente até 100 km/h e mantém essa velocidade entre os dois postos de pedágio). Agora, suponha que dez carros viajem em comboio, um atrás do outro, em ordem fixa. Imagine que cada carro seja um bit e que o comboio seja um pacote. Suponha ainda que cada posto de pedágio libere (isto é, transmita) um carro a cada 12 segundos, que seja tarde da noite e que os carros do comboio sejam os únicos na estrada. Por fim, suponha que, ao chegar a um posto de pedágio, o primeiro carro do comboio aguarde na entrada até que os outros nove cheguem e formem uma fila atrás dele. (Assim, o comboio inteiro deve ser ‘armazenado’ no posto de pedágio antes de começar a ser ‘reenviado’.) O tempo necessário para que todo o comboio passe pelo posto de pedágio e volte à estrada é de $(10 \text{ carros})/(5 \text{ carros/minuto}) = 2 \text{ minutos}$. Esse tempo é análogo ao atraso de transmissão em um roteador. O tempo necessário para um carro trafegar da

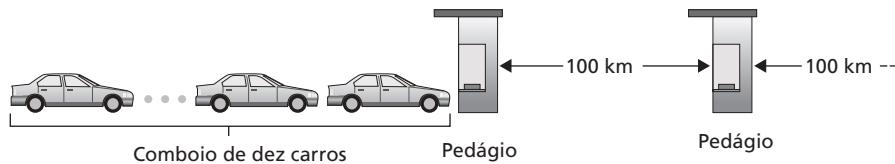


Figura 1.17 Analogia do comboio

saída de um posto de pedágio até o próximo posto de pedágio é de $(100 \text{ km})/(100 \text{ km/h}) = 1 \text{ hora}$. Esse tempo é análogo ao atraso de propagação. Portanto, o tempo decorrido entre o instante em que o comboio é ‘armazenado’ em frente a um posto de pedágio até o instante em que é ‘armazenado’ em frente ao seguinte é a soma do atraso de transmissão e do atraso de propagação — nesse exemplo, 62 minutos.

Vamos explorar um pouco mais essa analogia. O que aconteceria se o tempo de liberação do comboio no posto de pedágio fosse maior do que o tempo que um carro leva para trafegar entre dois postos? Por exemplo, suponha que os carros trafeguem a uma velocidade de 1.000 km/h e que o pedágio libere um carro por minuto. Então, o atraso de trânsito entre dois postos de pedágio é de 6 minutos e o tempo de liberação do comboio no posto de pedágio é de 10 minutos. Nesse caso, os primeiros carros do comboio chegarão ao segundo posto de pedágio antes que os últimos carros saiam do primeiro posto. Essa situação também acontece em redes de comutação de pacotes — os primeiros bits de um pacote podem chegar a um roteador enquanto muitos dos remanescentes ainda estão esperando para ser transmitidos pelo roteador precedente.

Se uma imagem vale mil palavras, então uma animação vale um milhão. O Companion Website apresenta um aplicativo interativo Java que ilustra e contrasta o atraso de transmissão e o atraso de propagação. Recomenda-se que o leitor visite esse aplicativo.

Se d_{proc} , d_{fila} , d_{trans} e d_{prop} forem, respectivamente, os atrasos de processamento, de fila, de transmissão e de propagação, então o atraso nodal total é dado por:

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{fila}} + d_{\text{trans}} + d_{\text{prop}}$$

A contribuição desses componentes do atraso pode variar significativamente. Por exemplo, d_{prop} pode ser desprezível (por exemplo, dois microssegundos) para um enlace que conecta dois roteadores no mesmo *campus* universitário; contudo, é de centenas de milissegundos para dois roteadores interconectados por um enlace de satélite geoestacionário e pode ser o termo dominante no d_{nodal} . De maneira semelhante, d_{trans} pode variar de desprezível a significativo. Sua contribuição normalmente é desprezível para velocidades de transmissão de 10 Mbps e mais altas (por exemplo, em LANs); contudo, pode ser de centenas de milissegundos para grandes pacotes de Internet enviados por enlaces de modems discados de baixa velocidade. O atraso de processamento, d_{proc} , é quase sempre desprezível; no entanto, tem forte influência sobre a produtividade máxima de um roteador, que é a velocidade máxima com que ele pode encaminhar pacotes.

1.4.2 Atraso de fila e perda de pacote

O mais complicado e interessante componente do atraso nodal é o atraso de fila, d_{fila} . Realmente, o atraso de fila é tão importante e interessante em redes de computadores que milhares de artigos e numerosos livros já foram escritos sobre ele [Bertsekas, 1991; Daigle, 1991; Kleinrock, 1975, 1976; Ross, 1995]. Neste livro, faremos apenas uma discussão intuitiva, de alto nível, sobre o atraso de fila; o leitor mais curioso pode consultar alguns dos livros citados (ou até mesmo escrever uma tese sobre o assunto!). Diferentemente dos três outros atrasos (a saber, d_{proc} , d_{trans} e d_{prop}), o atraso de fila pode variar de pacote a pacote. Por exemplo, se dez pacotes chegarem a uma fila vazia ao mesmo tempo, o primeiro pacote transmitido não sofrerá nenhum atraso, ao passo que o último pacote sofrerá um atraso relativamente grande (enquanto espera que os outros nove pacotes sejam transmitidos). Por conseguinte, para se caracterizar um atraso de fila, normalmente são utilizadas medições estatísticas, tais como atraso de fila médio, variância do atraso de fila e a probabilidade de ele exceder um valor especificado.

Quando o atraso de fila é grande e quando é insignificante? A resposta a essa pergunta depende da velocidade de transmissão do enlace, da taxa com que o tráfego chega à fila e de sua natureza, isto é, se chega intermitentemente, em rajadas. Para entendermos melhor, vamos adotar a para representar a taxa média com que os pacotes chegam à fila (a é medida em pacotes/segundo). Lembre-se de que R é a taxa de transmissão, isto é, a taxa (em bits/segundo) com que os bits são retirados da fila. Suponha também, para simplificar, que todos os pacotes tenham L bits. Então, a taxa média com que os bits chegam à fila é La bits/s. Por fim, imagine que a fila seja muito longa, de modo que, essencialmente, possa conter um número infinito de bits. A razão La/R , denominada intensidade de tráfego, frequentemente desempenha um papel importante na estimativa do tamanho do atraso de fila. Se $La/R > 1$, então a velocidade média com que os bits chegam à fila excederá a velocidade com que eles podem ser transmitidos para fora da fila. Nessa situação desastrosa, a fila tenderá a aumentar sem limite e o atraso de fila tenderá ao infinito! Por conseguinte, uma das regras de ouro da engenharia de tráfego é: *projete seu sistema de modo que a intensidade de tráfego não seja maior do que 1*.

Agora, considere o caso em que $La/R \leq 1$. Aqui, a natureza do tráfego influencia o atraso de fila. Por exemplo, se pacotes chegarem periodicamente — isto é, se chegar um pacote a cada L/R segundos — então todos os pacotes chegarão a uma fila vazia e não haverá atraso. Por outro lado, se pacotes chegarem em rajadas, mas periodicamente, poderá haver um significativo atraso de fila médio. Por exemplo, suponha que N pacotes cheguem ao mesmo tempo a cada $(L/R)N$ segundos. Então, o primeiro pacote transmitido não sofrerá atraso de fila; o segundo pacote transmitido terá um atraso de fila de L/R segundos e, de modo mais geral, o enésimo pacote transmitido terá um atraso de fila de $(n - 1)L/R$ segundos. Deixamos como exercício para o leitor o cálculo do atraso de fila médio para esse exemplo.

Os dois exemplos de chegadas periódicas que acabamos de descrever são um tanto acadêmicos. Na realidade, o processo de chegada a uma fila é *aleatório* — isto é, não segue um padrão e os intervalos de tempo entre os pacotes são ao acaso. Nessa hipótese mais realista, a quantidade La/R normalmente não é suficiente para caracterizar por completo a estatística do atraso. Não obstante, é útil para entender intuitivamente a extensão do atraso de fila. Em especial, se a intensidade de tráfego for próxima de zero, então as chegadas de pacotes serão poucas e bem espaçadas e é improvável que um pacote que esteja chegando encontre outro na fila. Consequentemente, o atraso de fila médio será próximo de zero. Por outro lado, quando a intensidade de tráfego for próxima de 1, haverá intervalos de tempo em que a velocidade de chegada excederá a capacidade de transmissão (devido às variações na taxa de chegada do pacote) e uma fila será formada durante esses períodos de tempo; quando a taxa de chegada for menor do que a capacidade de transmissão, a extensão da fila diminuirá. Todavia, à medida que a intensidade de tráfego se aproxima de 1, o comprimento médio da fila fica cada vez maior. A dependência qualitativa entre o atraso de fila médio e a intensidade de tráfego é mostrada na Figura 1.18.

Um aspecto importante a observar na Figura 1.18 é que, quando a intensidade de tráfego se aproxima de 1, o atraso de fila médio aumenta rapidamente. Uma pequena porcentagem de aumento na intensidade resulta em

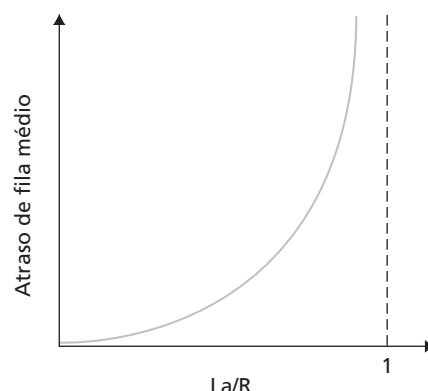


Figura 1.18 Dependência entre atraso de fila médio e intensidade de tráfego

um aumento muito maior no atraso, em termos de porcentagem. Talvez você já tenha percebido esse fenômeno na estrada. Se você dirige regularmente por uma estrada que normalmente está congestionada, o fato de ela estar sempre assim significa que a intensidade de tráfego é próxima de 1. Se algum evento causar um tráfego ligeiramente maior do que o usual, as demoras que você sofrerá poderão ser enormes.

Para compreender um pouco mais os atrasos de fila, visite o Companion Website, que apresenta um aplicativo Java interativo. Se você aumentar a taxa de chegada do pacote o suficiente de forma que a intensidade do tráfego exceda 1, você verá a fila aumentar ao longo do tempo.

Perda de pacote

Na discussão anterior, admitimos que a fila é capaz de conter um número infinito de pacotes. Na realidade, a capacidade da fila que precede um enlace é finita, embora a sua formação dependa bastante do projeto e do custo do comutador. Como a capacidade da fila é finita, na verdade os atrasos de pacote não se aproximam do infinito quando a intensidade de tráfego se aproxima de 1. O que realmente acontece é que um pacote pode chegar e encontrar uma fila cheia. Sem espaço disponível para armazená-lo, o roteador descartará esse pacote; isto é, ele será perdido. Esse excesso em uma fila pode ser observado novamente no aplicativo Java quando a intensidade do tráfego é maior do que 1. Do ponto de vista de um sistema final, uma perda de pacote é vista como um pacote que foi transmitido para o núcleo da rede, mas sem nunca ter emergido dele no destino. A fração de pacotes perdidos aumenta com o aumento da intensidade de tráfego. Por conseguinte, o desempenho em um nó é frequentemente medido não apenas em termos de atraso, mas também em termos da probabilidade de perda de pacotes. Como discutiremos nos capítulos subsequentes, um pacote perdido pode ser retransmitido fim a fim para garantir que todos os dados sejam finalmente transferidos da origem ao local de destino.

1.4.3 Atraso fim a fim

Até o momento, nossa discussão focalizou o atraso nodal, isto é, o atraso em um único roteador. Concluiremos essa discussão considerando brevemente o atraso da origem ao destino. Para entender esse conceito, suponha que haja $N - 1$ roteadores entre a máquina de origem e a máquina de destino. Imagine também que a rede não esteja congestionada (e, portanto, os atrasos de fila sejam desprezíveis), que o atraso de processamento em cada roteador e na máquina de origem seja d_{proc} , que a taxa de transmissão de saída de cada roteador e da máquina de origem seja R bits/s e que o atraso de propagação em cada enlace seja d_{prop} . Os atrasos nodais se acumulam e resultam em um atraso fim a fim

$$d_{\text{fim a fim}} = N (d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

onde, mais uma vez, $d_{\text{trans}} = L/R$ e L é o tamanho do pacote. Convidamos você a generalizar essa fórmula para o caso de atrasos heterogêneos nos nós e para o caso de um atraso de fila médio em cada nó.

Traceroute

Para perceber o que é realmente o atraso em uma rede de computadores, podemos utilizar o Traceroute, programa de diagnóstico que pode ser executado em qualquer máquina da Internet. Quando o usuário especifica um nome de hospedeiro de destino, o programa no hospedeiro de origem envia vários pacotes especiais em direção àquele destino. Ao seguir seu caminho até o destino, esses pacotes passam por uma série de roteadores. Um deles recebe um desses pacotes especiais e envia uma curta mensagem à origem. Essa mensagem contém o nome e o endereço do roteador.

Mais especificamente, suponha que haja $N - 1$ roteadores entre a origem e o destino. Então, a fonte enviará N pacotes especiais à rede e cada um deles estará endereçado ao destino final. Esses N pacotes especiais serão marcados de 1 a N , sendo a marca do primeiro pacote 1 e a do último, N . Assim que o enésimo roteador recebe o enésimo pacote com a marca n , não envia o pacote a seu destino, mas uma mensagem à origem. Quando o hospedeiro de destino recebe o pacote N , também envia uma mensagem à origem, que registra o tempo transcorrido entre o envio de um pacote e o recebimento da mensagem de retorno correspondente. A origem registra também o

nome e o endereço do roteador (ou do hospedeiro de destino) que retorna a mensagem. Dessa maneira, a origem pode reconstruir a rota tomada pelos pacotes que vão da origem ao destino e pode determinar os atrasos de ida e volta para todos os roteadores intervenientes. Na prática, o programa Traceroute repete o processo que acabamos de descrever três vezes, de modo que a fonte envia, na verdade, $3 \cdot N$ pacotes ao destino. O RFC 1393 descreve detalhadamente o Traceroute.

Eis um exemplo de resultado do programa Traceroute, no qual a rota traçada ia do hospedeiro de origem `gaia.cs.umass.edu` (na Universidade de Massachusetts) até `cis.poly.edu` (na Polytechnic University no Brooklyn). O resultado tem seis colunas: a primeira coluna é o valor n descrito acima, isto é, o número do roteador ao longo da rota; a segunda coluna é o nome do roteador; a terceira coluna é o endereço do roteador (na forma `xxx.xxx.xxx.xxx`); as últimas três colunas são os atrasos de ida e volta para três tentativas. Se a fonte receber menos do que três mensagens de qualquer roteador determinado (devido à perda de pacotes na rede), o Traceroute coloca um asterisco logo após o número do roteador e registra menos do que três tempos de duração de viagens de ida e volta para aquele roteador.

```

1 cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2 128.119.3.154 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3 border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 ms 0.484 ms 0.451 ms
4 acr1-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5 agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
6 acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7 pos10-2.core2.NewYork1.Level13.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8 gige9-1-52.hsipaccess1.NewYork1.Level13.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9 p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms

```

No exemplo anterior há nove roteadores entre a origem e o destino. Quase todos eles têm um nome e todos têm endereço. Por exemplo, o nome do Roteador 3 é `border4-rt-gi-1-3.gw.umass.edu` e seu endereço é `128.119.2.194`. Examinando os dados apresentados para esse roteador, verificamos que na primeira das três tentativas, o atraso de ida e volta entre a origem e o roteador foi de 1,03 milissegundos. Os atrasos para as duas tentativas subsequentes foram 0,48 e 0,45 milissegundos e incluem todos os atrasos que acabamos de discutir, ou seja, atrasos de transmissão, de propagação, de processamento do roteador e de fila. Como o atraso de fila varia com o tempo, o atraso de ida e volta do pacote n enviado a um roteador n pode, às vezes, ser maior do que o do pacote $n+1$ enviado ao roteador $n+1$. Realmente, observamos esse fenômeno no exemplo acima: os atrasos do roteador 6 são maiores que os atrasos do roteador 7!

Você quer experimentar o Traceroute por conta própria? Recomendamos muito que visite o site <http://www.traceroute.org>, que provê uma interface Web para uma extensa lista de fontes para traçar rotas. Escolha uma fonte, forneça o nome de hospedeiro para qualquer destino e o programa Traceroute fará todo o trabalho. Existem muitos programas de software gratuitos que apresentam uma interface gráfica para o Traceroute; um dos nossos favoritos é o PingPlotter [PingPlotter, 2009].

Sistema final, aplicativo e outros atrasos

Além dos atrasos de processamento, transmissão e de propagação, os sistemas finais podem adicionar outros atrasos significativos. Os modems discados apresentam um atraso de modulação/codificação, que pode ser da ordem de dezenas de microssegundos. (Os atrasos de modulação/codificação para outras tecnologias de acesso — incluindo a Ethernet, o modem a cabo e a DSL — são menos significativos e normalmente desprezíveis.) Um sistema final que quer transmitir um pacote para uma mídia compartilhada (por exemplo, como em um cenário WiFi ou Ethernet) pode, *intencionalmente*, atrasar sua transmissão como parte de seu protocolo por compartilhar a mídia com outros sistemas finais; vamos analisar tais protocolos no Capítulo 5. Um outro importante atraso é o atraso de empacotamento de mídia, o qual está presente nos aplicativos VoIP (voz sobre IP). No VoIP, o

remetente deve primeiro carregar um pacote com voz digitalizada e codificada antes de transmitir o pacote para a Internet. A etapa de carregar um pacote — chamada de atraso de empacotamento — pode ser significativa e ter impacto sobre a qualidade visível pelo usuário de uma chamada VoIP. Esse assunto será explorado mais adiante nos exercícios de fixação no final deste capítulo.

1.4.4 Vazão nas redes de computadores

Além do atraso e da perda de pacotes, outra medida de desempenho importante em redes de computadores é a vazão fim a fim. Para definir vazão, considere a transferência de um arquivo grande do Hospedeiro A para o Hospedeiro B através de uma rede de computadores. Essa transferência pode ser, por exemplo, um clipe extenso de um parceiro para outro por meio do sistema de compartilhamento P2P. A vazão instantânea a qualquer momento é a taxa (em bits/s) em que o Hospedeiro B está recebendo o arquivo. (Muitos aplicativos, incluindo muitos sistemas de compartilhamento P2P, exibem a vazão instantânea durante os downloads na interface do usuário — talvez você já tenha observado isso!) Se o arquivo consistir em F bits e a transferência levar T segundos para o Hospedeiro B receber todos os F bits, então a vazão média da transferência do arquivo é F/T bits/s. Para algumas aplicações, como a telefonia via Internet, é desejável ter um atraso baixo e uma vazão instantânea acima de algum limiar (por exemplo, superior a 24 kbps para telefonia via Internet, e superior a 256 kbps para alguns aplicativos de vídeo em tempo real). Para outras aplicações, incluindo as de transferência de arquivo, o atraso não é importante, mas é recomendado ter a vazão mais alta possível.

Para obter uma visão mais detalhada, vamos analisar alguns exemplos. A Figura 1.19 (a) mostra dois sistemas finais, um servidor e um cliente, conectados por dois enlaces de comunicação e um roteador. Considere a vazão para uma transferência de arquivo do servidor para o cliente. Suponha que R_s seja a taxa do enlace entre o roteador e o cliente; e R_c seja a taxa do enlace entre o roteador e o cliente. Imagine que os únicos bits enviados na rede inteira sejam os do servidor para o cliente. Agora vem a pergunta, neste cenário ideal, qual é a vazão servidor-para-cliente? Para responder a ela, pense nos bits como um líquido e nos enlaces de comunicação como canos. Evidentemente, o servidor não pode enviar os bits através de seu enlace a uma taxa mais rápida do que R_s bps, e o roteador não pode encaminhar os bits a uma taxa mais rápida do que R_c bps. Se $R_s < R_c$, então os bits enviados pelo servidor irão “correr” diretamente pelo roteador e chegar no cliente a uma taxa de R_c bps. Se, por outro lado, $R_c < R_s$, então o roteador não poderá encaminhar os bits tão rapidamente quanto ele os recebe. Neste caso, os bits somente deixarão o roteador a uma taxa R_c , dando uma vazão fim a fim de R_c . (Observe também que se os bits continuarem a chegar no roteador a uma taxa R_s , e continuarem a deixar o roteador a uma taxa R_c , o acúmulo de bits no roteador esperando para transmissão ao cliente só aumentará — uma situação, na maioria das vezes, indesejável!) Assim, para essa rede simples de dois enlaces, a vazão é mín { R_c, R_s }, ou seja, é a taxa de transmissão do enlace de gargalo. Após determinar a vazão, agora podemos

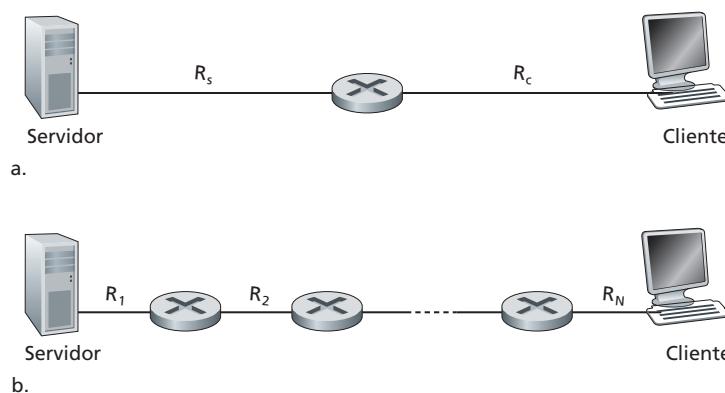


Figura 1.19 Vazão para uma transferência de arquivo do servidor ao cliente

aproximar o tempo que leva para transferir um arquivo grande de F bits do servidor ao cliente como $F/\min\{R_s, R_c\}$. Para um exemplo específico, suponha que você está fazendo o download de um arquivo MP3 de $F = 32$ milhões de bits, o servidor tem uma taxa de transmissão de $R_s = 2$ Mbps, e você tem um enlace de acesso de $R_c = 1$ Mbps. O tempo necessário para transferir o arquivo é, então, 32 segundos. Naturalmente essas expressões para tempo de vazão e de transferência são apenas aproximações, já que elas não contabilizam assuntos relacionados a protocolos e pacotes.

A Figura 1.19 (b) agora mostra uma rede com N enlaces entre o servidor e o cliente, com as taxas de transmissão de N enlaces sendo R_1, R_2, \dots, R_N . Aplicando a mesma análise da rede de dois enlaces, descobrimos que a vazão para uma transferência de arquivo do servidor ao cliente é $\min\{R_1, R_2, \dots, R_N\}$, a qual é novamente a taxa de transmissão do enlace de gargalo ao longo do caminho entre o servidor e o cliente.

Agora considere outro exemplo motivado pela Internet de hoje. A Figura 1.20 (a) mostra dois sistemas finais, um servidor e um cliente, conectados a uma rede de computadores. Considere a vazão para uma transferência de arquivo do servidor ao cliente. O servidor está conectado à rede com um enlace de acesso de taxa R_s e o cliente está conectado à rede com um enlace de acesso de R_c . Agora suponha que todos os enlaces no núcleo da rede de comunicação tenham taxas altas de transmissão, muito maiores do que R_s e R_c . Realmente, hoje, o núcleo da Internet está superabastecido com enlaces de alta velocidade que sofrem pouco congestionamento [Akella, 2003]. Suponha, também, que os únicos bits que estão sendo enviados em toda a rede sejam os do servidor para o cliente. Como o núcleo da rede de computadores é como um cano largo neste exemplo, a taxa a qual os bits correm da origem ao lugar de destino é novamente o mínimo de R_s e R_c , ou seja, $\text{vazão} = \min\{R_s, R_c\}$. Portanto, o fator coercitivo para vazão na Internet de hoje é, normalmente, o acesso à rede.

Para um exemplo final, considere a Figura 1.20 (b) na qual existem 10 servidores e 10 clientes conectados ao núcleo da rede de computadores. Neste exemplo, 10 downloads simultâneos estão sendo realizados, envolvendo 10 pares de clientes-servidores. Suponha que esses 10 downloads sejam o único tráfego na rede no momento presente. Como mostrado na figura, há um enlace no núcleo que é atravessado por todos os 10 downloads. Considere R a taxa de transmissão desse enlace. Suponha que todos os enlaces de acesso do servidor possuem a mesma taxa

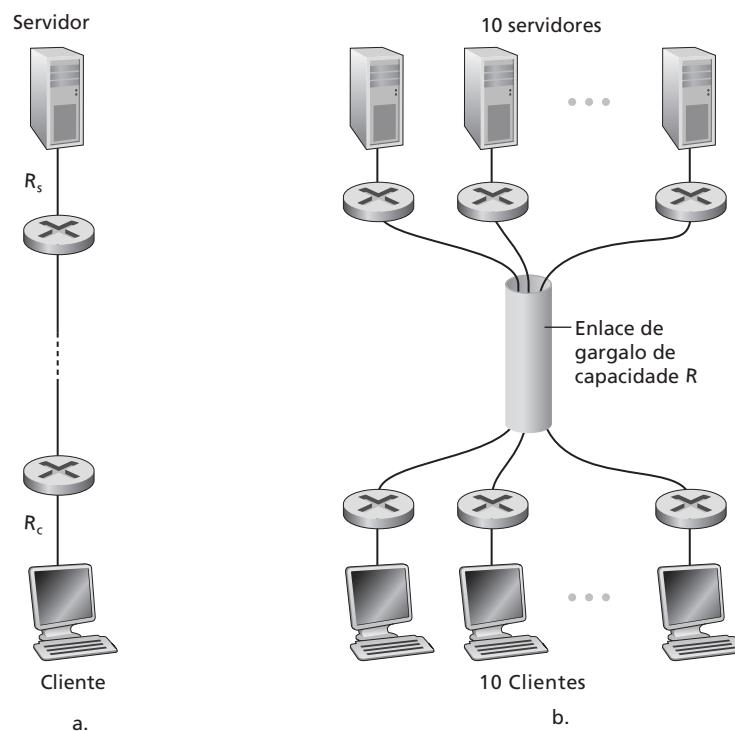


Figura 1.20 Vazão fim a fim: (a) O cliente baixa um arquivo do servidor; (b) 10 clientes fazem o download com 10 servidores

R_s , todos os enlaces de acesso do cliente possuem a mesma taxa R_c e a taxa de transmissão de todos os enlaces no núcleo — com exceção de um enlace comum de taxa R — sejam muito maiores do que R_s , R_c e R . Agora perguntamos, quais são as vazões dos downloads? Evidentemente, se a taxa do enlace comum, R , é grande — digamos que cem vezes maior do que R_s , R_c — então a vazão para cada download será novamente mín { R_s , R_c }. Mas e se a taxa do enlace comum for da mesma ordem que R_s e R_c ? Qual será a vazão neste caso? Vamos observar um exemplo específico. Suponha que $R_s = 2$ Mbps, $R_c = 1$ Mbps, $R = 5$ Mbps, e o enlace comum divide sua taxa de transmissão igualmente entre 10 downloads. Então, o gargalo para cada download não se encontra mais na rede de acesso, mas é o enlace compartilhado no núcleo, que somente fornece para cada download 500 kbps de vazão. Deste modo, a vazão fim a fim para cada download é agora reduzida a 500 kbps.

Os exemplos na Figura 1.19 e Figura 1.20 (a) mostram que a vazão depende das taxas de transmissão dos enlaces sobre as quais os dados correm. Vimos que quando não há tráfego interveniente, a vazão pode simplesmente ser aproximada como a taxa de transmissão mínima ao longo do caminho entre a origem e o local de destino. O exemplo na Figura 1.20 (b) mostra que, de maneira geral, a vazão depende não somente das taxas de transmissão dos enlaces ao longo do caminho, mas também do tráfego interveniente. Em especial, um enlace com uma alta taxa de transmissão pode, todavia, ser o enlace de gargalo para uma transferência de arquivo, caso muitos outros fluxos de dados estejam também passando por aquele enlace. Analisaremos mais detalhadamente vazão em redes de computadores nos exercícios de fixação e nos capítulos subsequentes.

1.5 Camadas de protocolo e seus modelos de serviço

Até aqui, nossa discussão demonstrou que a Internet é um sistema *extremamente* complicado e que possui muitos componentes: inúmeras aplicações e protocolos, vários tipos de sistemas finais e conexões entre eles, roteadores, além de vários tipos de meios físicos de enlace. Dada essa enorme complexidade, há alguma esperança de organizar a arquitetura de rede ou, ao menos, nossa discussão sobre ela? Felizmente, a resposta a ambas as perguntas é sim.

1.5.1 Arquitetura de camadas

Antes de tentarmos organizar nosso raciocínio sobre a arquitetura da Internet, vamos procurar uma analogia humana. Na verdade, lidamos com sistemas complexos o tempo todo em nosso dia a dia. Imagine se alguém pedisse que você descrevesse, por exemplo, o sistema de uma companhia aérea. Como você encontraria a estrutura para descrever esse sistema complexo que tem agências de emissão de passagens, pessoal para embarcar a bagagem para ficar no portão de embarque, pilotos, aviões, controle de tráfego aéreo e um sistema mundial de roteamento de aeronaves? Um modo de descrever esse sistema poderia ser apresentar a relação de uma série de ações que você realiza (ou que outros realizam para você) quando voa por uma empresa aérea. Você compra a passagem, despacha suas malas, dirige-se ao portão de embarque e, finalmente, entra no avião, que decola e segue uma rota até seu destino. Após a aterrissagem, você desembarca no portão designado e recupera suas malas. Se a viagem foi ruim, você reclama na agência que lhe vendeu a passagem (esforço em vão). Esse cenário é ilustrado na Figura 1.21.

Já podemos notar aqui algumas analogias com redes de computadores: você está sendo despachado da origem ao destino pela companhia aérea; um pacote é despachado da máquina de origem à máquina de destino na Internet. Mas essa não é exatamente a analogia que buscamos. Estamos tentando encontrar alguma *estrutura* na Figura 1.21. Observando essa figura, notamos que há uma função referente à passagem em cada ponta; há também uma função de bagagem para passageiros que já apresentaram a passagem e uma função de portão de embarque para passageiros que já apresentaram a passagem e despacharam a bagagem. Para passageiros que já passaram pelo portão de embarque (isto é, aqueles que já apresentaram a passagem, despacharam a bagagem e passaram pelo portão), há uma função de decolagem e de aterrissagem e, durante o voo, uma função de roteamento do avião. Isso sugere que podemos examinar a funcionalidade na Figura 1.21 na horizontal, como mostra a Figura 1.22.

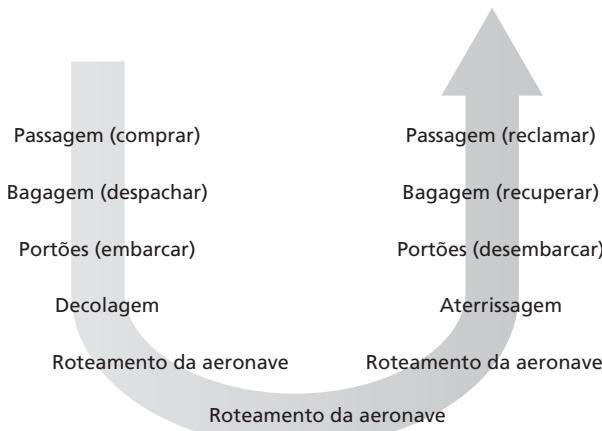


Figura 1.21 Uma viagem de avião: ações

A Figura 1.22 dividiu a funcionalidade da linha aérea em camadas, provendo uma estrutura com a qual podemos discutir a viagem aérea. Note que cada camada, combinada com as camadas abaixo dela, implementa alguma funcionalidade, algum serviço. Na camada da passagem aérea e abaixo dela, é realizada a transferência ‘balcão-de-linha-aérea-balcão-de-linha-aérea’ de um passageiro. Na camada de bagagem e abaixo dela, é realizada a transferência ‘despacho-de-bagagem–recuperação-de-bagagem’ de um passageiro e de suas malas. Note que a camada da bagagem provê esse serviço apenas para a pessoa que já apresentou a passagem. Na camada do portão, é realizada a transferência ‘portão-de-embarque-portão-de-desembarque’ de um passageiro e de sua bagagem. Na camada de decolagem/aterriagem, é realizada a transferência ‘pista-a-pista’ de passageiros e de suas bagagens. Cada camada provê seu serviço (1) realizando certas ações dentro da camada (por exemplo, na camada do portão, embarcar e desembarcar pessoas de um avião) e (2) utilizando os serviços da camada imediatamente inferior (por exemplo, na camada do portão, aproveitando o serviço de transferência ‘pista-a-pista’ de passageiros da camada de decolagem/aterriagem).

Uma arquitetura de camadas nos permite discutir uma parcela específica e bem definida de um sistema grande e complexo. Essa simplificação tem considerável valor intrínseco, pois provê modularidade fazendo com que fique muito mais fácil modificar a implementação do serviço prestado pela camada. Contanto que a camada forneça o mesmo serviço para a que está acima dela e use os mesmos serviços da camada abaixo dela, o restante do sistema permanece inalterado quando a sua implementação é modificada. (Note que modificar a implementação de um serviço é muito diferente de mudar o serviço em si!) Por exemplo, se as funções de portão fossem modificadas (digamos que passassem a embarcar e desembarcar passageiros por ordem de altura), o restante do sistema da linha aérea permaneceria inalterado, já que a camada do portão continuaria a prover a mesma função.

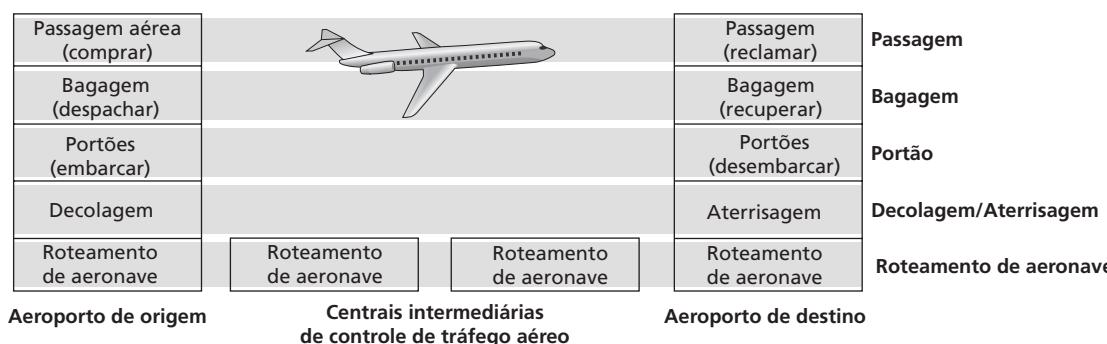


Figura 1.22 Camadas horizontais da funcionalidade de linha aérea

(embarcar e desembarcar passageiros); ela simplesmente implementaria aquela função de maneira diferente após a modificação. Para sistemas grandes e complexos que são atualizados constantemente, a capacidade de modificar a implementação de um serviço sem afetar outros componentes do sistema é outra vantagem importante da divisão em camadas.

Camadas de protocolo

Mas chega de linhas aéreas! Vamos agora voltar nossa atenção a protocolos de rede. Para prover uma estrutura para o projeto de protocolos de rede, projetistas de rede organizam protocolos — e o hardware e o software de rede que implementam os protocolos — em camadas. Cada protocolo pertence a uma das camadas, exatamente como cada função na arquitetura de linha aérea da Figura 1.22 pertence a uma camada. Novamente estamos interessados nos serviços que uma camada oferece à camada acima dela — denominado modelo de serviço de uma camada. Exatamente como no nosso exemplo da linha aérea, cada camada provê seu serviço (1) executando certas ações dentro da camada e (2) utilizando os serviços da camada diretamente abaixo dela. Por exemplo, os serviços providos pela camada n podem incluir entrega confiável de mensagens de uma extremidade da rede à outra, que pode ser implementada utilizando um serviço não confiável de entrega de mensagem fim a fim da camada $n - 1$ e adicionando funcionalidade da camada n para detectar e retransmitir mensagens perdidas.

Uma camada de protocolo pode ser implementada em software, em hardware, ou em uma combinação dos dois. Protocolos de camada de aplicação como HTTP e SMTP — quase sempre são implementados em software em sistemas finais; o mesmo acontece com protocolos de camada de transporte. Como a camada física e as camadas de enlace de dados são responsáveis pelo manuseio da comunicação por um enlace específico, normalmente são implementadas em uma placa de interface de rede (por exemplo, placas de interface Ethernet ou Wi-Fi) associadas a um determinado enlace. A camada de rede quase sempre é uma implementação mista de hardware e software. Note também que, exatamente como as funções na arquitetura em camadas da linha aérea eram distribuídas entre os vários aeroportos e centrais de controle de tráfego aéreo que compunham o sistema, também um protocolo de camada n é distribuído entre os sistemas finais, comutadores de pacote e outros componentes que formam a rede. Isto é, há sempre uma parcela de um protocolo de camada n em cada um desses componentes de rede.

O sistema de camadas de protocolos tem vantagens conceituais e estruturais. Como vimos, a divisão em camadas proporciona um modo estruturado de discutir componentes de sistemas. A modularidade facilita a atualização de componentes de sistema. Devemos mencionar, no entanto, que alguns pesquisadores e engenheiros de rede se opõem veementemente ao sistema de camadas [Wakeman, 1992]. Uma desvantagem potencial desse sistema é que uma camada pode duplicar a funcionalidade de uma camada inferior. Por exemplo, muitas pilhas de protocolos oferecem serviço de recuperação de erros na camada de enlace e também fim a fim. Uma segunda desvantagem potencial é que a funcionalidade em uma camada pode necessitar de informações (por exemplo, um valor de carimbo de tempo) que estão presentes somente em uma outra camada, o que infringe o objetivo de separação de camadas.

Quando tomados em conjunto, os protocolos das várias camadas são denominados pilha de protocolos, que é formada por cinco camadas: física, de enlace, de rede, de transporte e de aplicação, como mostra a Figura 1.23 (a). Se você verificar o sumário, verá que organizamos este livro utilizando as camadas da pilha de protocolos da Internet. Fazemos uma abordagem descendente, primeiro abordando as camadas de aplicação e, então, os processos.

Camada de aplicação

A camada de aplicação é onde residem aplicações de rede e seus protocolos. Ela inclui muitos protocolos, tais como o protocolo HTTP (que provê requisição e transferência de documentos pela Web), o SMTP (que provê transferência de mensagens de correio eletrônico) e o FTP (que provê a transferência de arquivos entre dois sistemas finais). Veremos que certas funções de rede, como a tradução de nomes fáceis de entender dados a sistemas finais da Internet (por exemplo, `gaia.cs.umass.edu`) para um endereço de rede de 32 bits, também são executadas com a ajuda de um protocolo de camada de aplicação, no caso, o sistema de nomes de domínio (*domain name system* — DNS). Veremos no Capítulo 2 que é muito fácil criar nossos próprios novos protocolos de camada de aplicação.

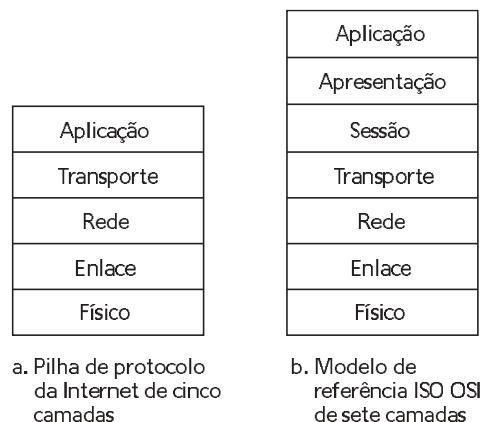


Figura 1.23 A pilha de protocolo da Internet (a) e o modelo de referência OSI (b)

Um protocolo de camada de aplicação é distribuído por diversos sistemas finais, sendo que a aplicação em um sistema final utiliza o protocolo para trocar pacotes de informação com a aplicação em outro sistema final. Denominaremos esse pacote de informação na camada de aplicação de mensagem.

Camada de transporte

A camada de transporte da Internet transporta mensagens da camada de aplicação entre os lados do cliente e servidor de uma aplicação. Há dois protocolos de transporte na Internet: TCP e UDP, e qualquer um deles pode levar mensagens de camada de aplicação. O TCP provê serviços orientados para conexão para suas aplicações. Alguns desses serviços são a entrega garantida de mensagens da camada de aplicação ao destino e controle de fluxo (isto é, compatibilização das velocidades do remetente e do receptor). O TCP também fragmenta mensagens longas em segmentos mais curtos e provê mecanismo de controle de congestionamento, de modo que uma origem regula sua velocidade de transmissão quando a rede está congestionada. O protocolo UDP provê serviço não orientado para conexão a suas aplicações. Este é um serviço econômico que fornece segurança, sem controle de fluxo e de congestionamento. Neste livro, um pacote de camada de transporte será denominado segmento.

Camada de rede

A camada de rede da Internet é responsável pela movimentação, de uma máquina para outra, de pacotes de camada de rede conhecidos como *datagramas*. O protocolo de camada de transporte da Internet (TCP ou UDP) em uma máquina de origem passa um segmento de camada de transporte e um endereço de destino à camada de rede, exatamente como você passaria ao serviço de correios uma carta com um endereço de destinatário. A camada de rede então provê o serviço de entrega do segmento à camada de transporte na máquina destinatária.

A camada de rede da Internet tem dois componentes principais. Um deles é um protocolo que define os campos no datagrama, bem como o modo como os sistemas finais e os roteadores agem nesses campos. Este é o famoso protocolo IP. Existe somente um único protocolo IP, e todos os componentes da Internet que têm uma camada de rede devem executar esse protocolo. O outro componente importante é o protocolo de roteamento que determina as rotas que os datagramas seguem entre origens e destinos. A Internet tem muitos protocolos de roteamento. Como vimos na Seção 1.3, a Internet é uma rede de redes e, dentro de uma delas, o administrador pode executar qualquer protocolo de roteamento que queira. Embora a camada de rede contenha o protocolo IP e também numerosos protocolos de roteamento, ela quase sempre é denominada simplesmente camada IP, refletindo o fato de que ele é o elemento fundamental que mantém a integridade da Internet.

Camada de enlace

A camada de rede da Internet roteia um datagrama por meio de uma série de roteadores entre a origem e o destino. Para levar um pacote de um nó (sistema final ou comutador de pacotes) ao nó seguinte na rota, a camada de rede depende dos serviços da camada de enlace. Em especial, em cada nó, a camada de rede passa o datagrama para a camada de enlace, que o entrega, ao longo da rota, ao nó seguinte, no qual o datagrama é passado da camada de enlace para a de rede.

Os serviços prestados pela camada de enlace dependem do protocolo específico empregado no enlace. Alguns protocolos de camada de enlace proveem entrega garantida entre enlaces, isto é, desde o nó transmissor, passando por um único enlace, até o nó receptor. Note que esse serviço confiável de entrega é diferente do serviço de entrega garantida do TCP, que provê serviço de entrega garantida de um sistema final a outro. Exemplos de protocolos de camadas de enlace são Ethernet, WiFi e PPP (*point-to-point protocol* — protocolo ponto-a-ponto). Como datagramas normalmente precisam transitar por diversos enlaces para irem da origem ao destino, serão manuseados por diferentes protocolos de camada de enlace em diferentes enlaces ao longo de sua rota, podendo ser manuseados por Ethernet em um enlace e por PPP no seguinte. A camada de rede receberá um serviço diferente de cada um dos variados protocolos de camada de enlace. Neste livro, pacotes de camada de enlace serão denominados quadros.

Camada física

Enquanto a tarefa da camada de enlace é movimentar quadros inteiros de um elemento da rede até um elemento adjacente, a da camada física é movimentar os *bits individuais* que estão dentro do quadro de um nó para o seguinte. Os protocolos nessa camada novamente dependem do enlace e, além disso, dependem do próprio meio de transmissão do enlace (por exemplo, fios de cobre trançado ou fibra ótica monomodal). Por exemplo, a Ethernet tem muitos protocolos de camada física: um para par de fios de cobre trançado, outro para cabo coaxial, um outro para fibra e assim por diante. Em cada caso, o bit é movimentado pelo enlace de um modo diferente.

O modelo OSI

Após discutir detalhadamente a pilha de protocolo da Internet, devemos mencionar que essa não é a única pilha de protocolo existente. Particularmente, no final dos anos 1970, a Organização Internacional para Padronização (ISO) propôs que as redes de computadores fossem organizadas em, aproximadamente, sete camadas, denominadas modelo de Interconexão de Sistemas Abertos (OSI) [ISO, 2009]. O modelo OSI tomou forma quando os protocolos que iriam se tornar protocolos da Internet estavam em amadurecimento; e eram um dos muitos conjuntos de protocolos em desenvolvimento; na verdade, os inventores do modelo OSI original provavelmente não tinham a Internet em mente ao criá-lo. No entanto, no final dos anos 1970, muitos cursos universitários e de treinamento obtiveram conhecimentos sobre a exigência do ISO e organizaram cursos voltados para o modelo de sete camadas. Em razão de seu impacto precoce na educação de redes, o modelo de sete camadas continua presente em alguns livros sobre redes e em cursos de treinamento.

As sete camadas do modelo de referência OSI, mostradas na Figura 1.23 (b), são: camada de aplicação, camada de apresentação, camada de sessão, camada de transporte, camada de rede, camada de enlace e camada física. A função de cinco dessas camadas é a mesma que seus componentes da Internet. Desse modo, vamos considerar as duas camadas adicionais presentes no modelo de referência OSI — a camada de apresentação e a camada de sessão. O papel da camada de apresentação é prover serviços que permitam que as aplicações de comunicação interpretem o significado dos dados trocados. Entre esses serviços estão a compressão de dados e a codificação de dados (o que não precisa de explicação), assim como a descrição de dados (que, como veremos no Capítulo 9, livram as aplicações da preocupação com o formato interno onde os dados estão sendo descritos/armazenados — formato esse que podem se diferenciar de um computador para o outro). A camada de sessão provê a delimitação e sincronização da troca de dados, incluindo os meios de construir um esquema de pontos de verificação e de recuperação.

O fato de a Internet ser desprovida de duas camadas encontradas no modelo de referência OSI faz surgir duas questões: os serviços fornecidos por essas camadas são irrelevantes? E se uma aplicação precisar de um desses

serviços? A resposta da Internet para essas perguntas é a mesma — depende do criador da aplicação. Cabe ao criador da aplicação decidir se um serviço é importante, e se o serviço *for* importante, cabe ao criador da aplicação desenvolver essa função para ela.

1.5.2 Mensagens, segmentos, datagramas e quadros

A Figura 1.24 apresenta o caminho físico que os dados percorrem: para baixo na pilha de protocolos de um sistema final emissor, para cima e para baixo nas pilhas de protocolos de um comutador de camada de enlace interveniente e de um roteador e então para cima na pilha de protocolos do sistema final receptor. Como discutiremos mais adiante neste livro, ambos, comutadores de camada de enlace e roteadores, são comutadores de pacotes. De modo semelhante a sistemas finais, roteadores e comutadores de camada de enlace organizam seu hardware e software de rede em camadas. Mas roteadores e comutadores de camada de enlace não implementam *todas* as camadas da pilha de protocolos; normalmente implementam apenas as camadas de baixo. Como mostra a Figura 1.24, comutadores de camada de enlace implementam as camadas 1 e 2; roteadores implementam as camadas 1, 2 e 3. Isso significa, por exemplo, que roteadores da Internet são capazes de implementar o protocolo IP (da camada 3), mas comutadores de camada de enlace não. Veremos mais adiante que, embora não reconheçam endereços IP, comutadores de camada de enlace são capazes de reconhecer endereços de camada 2, tais como endereços da Ethernet. Note que sistemas finais implementam todas as cinco camadas, o que é consistente com a noção de que a arquitetura da Internet concentra sua complexidade na periferia da rede.

A Figura 1.24 também ilustra o importante conceito de encapsulamento. Uma mensagem de camada de aplicação na máquina emissor (M na Figura 1.24) é passada para a camada de transporte. No caso mais simples, esta pega a mensagem e anexa informações adicionais (denominadas informações de cabeçalho de camada de transporte, H_t na Figura 1.24) que serão usadas pela camada de transporte do lado receptor. A mensagem de camada de aplicação e as informações de cabeçalho da camada de transporte, juntas, constituem o segmento da camada de transporte, que encapsula a mensagem da camada de aplicação. As informações adicionadas podem

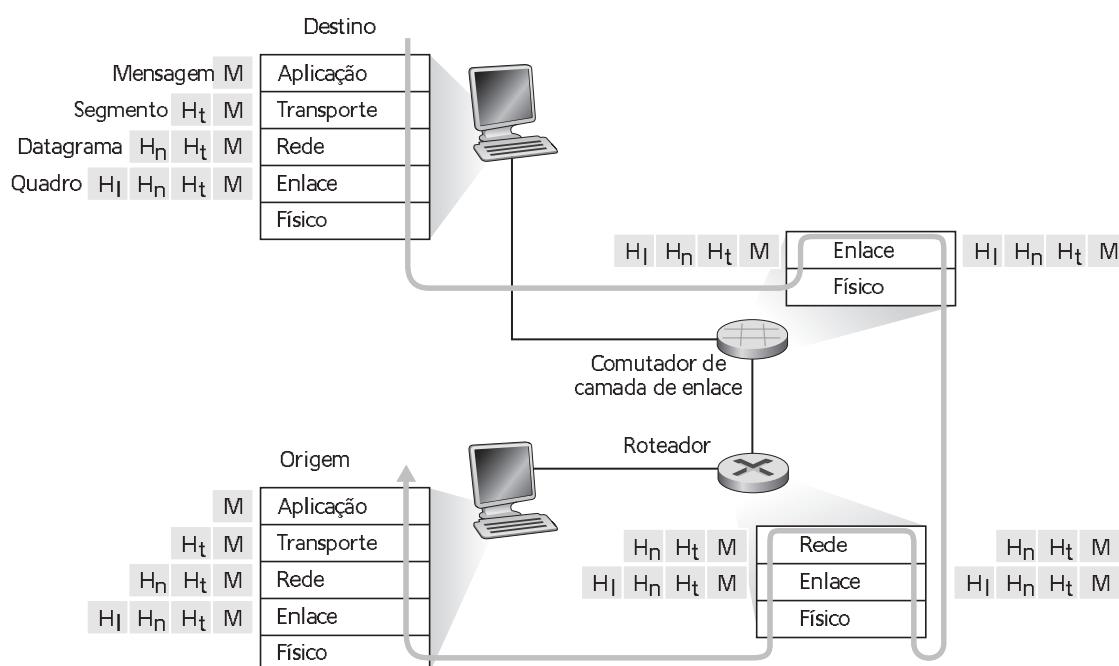


Figura 1.24 Hospedeiros, roteadores e comutadores de camada de enlace; cada um contém um conjunto diferente de camadas, refletindo suas diferenças em funcionalidade

incluir dados que habilitem a camada de transporte do lado do receptor a entregar a mensagem à aplicação apropriada, além de bits de detecção de erro que permitem que o receptor determine se os bits da mensagem foram modificados em trânsito. A camada de transporte então passa o segmento à camada de rede, que adiciona informações de cabeçalho de camada de rede (H_n na Figura 1.24), como endereços de sistemas finais de origem e de destino, criando um datagrama de camada de rede. Este é então passado para a camada de enlace, que (é claro!), adicionará suas próprias informações de cabeçalho e criará um quadro de camada de enlace. Assim, vemos que, em cada camada, um pacote possui dois tipos de campos: campos de cabeçalho e um campo de carga útil. A carga útil é normalmente um pacote da camada acima.

Uma analogia útil que podemos usar aqui é o envio de um memorando entre escritórios de um empresa pelo correio de uma filial corporativa a outra. Suponha que Alice, que está em uma filial, queira enviar um memorando a Bob, que está na outra filial. O memorando *representa* uma mensagem da camada de aplicação. Alice coloca o memorando em um envelope de correspondência interna em cuja face são escritos o nome e o departamento de Bob. O envelope de correspondência interna representa o segmento da camada de aplicação — contém as informações de cabeçalho (o nome de Bob e seu departamento) e encapsula a mensagem de camada de aplicação (o memorando). Quando a central de correspondência do escritório emissor recebe o envelope, ele é colocado dentro de outro envelope adequado para envio pelo correio. A central de correspondência emissora também escreve o endereço postal do remetente e do destinatário no envelope postal. Neste ponto, o envelope postal é análogo ao datagrama — encapsula o segmento de camada de transporte (o envelope de correspondência interna), que por sua vez encapsula a mensagem original (o memorando). O correio entrega o envelope postal à central de correspondência do escritório destinatário. Nesse local, o processo de reencapsulamento se inicia. A central de correspondência retira o memorando e o encaminha a Bob. Este, finalmente, abre o envelope e retira o memorando. O processo de encapsulamento pode ser mais complexo do que o descrito acima. Por exemplo, uma mensagem grande pode ser dividida em vários segmentos de camada de transporte (que também podem ser divididos em vários datagramas de camada de rede). Na extremidade receptora, cada segmento deve ser reconstruído a partir dos datagramas que o compõem.

1.6 Redes sob ameaça

A Internet se tornou essencial para muitas instituições hoje, incluindo empresas grandes e pequenas, universidades e órgãos do governo. Muitas pessoas também contam com a Internet para suas atividades profissionais, sociais e pessoais. Mas atrás de toda essa utilidade e entusiasmo, existe o lado negro, um lado no qual “vilões” tentam causar problemas em nosso cotidiano danificando nossos computadores conectados à Internet, violando nossa privacidade e tornando inoperantes os serviços da Internet dos quais dependemos [Skoudis, 2006].

A área de segurança de rede abrange como esses vilões podem ameaçar as redes de computadores e como nós, futuros especialistas no assunto, podemos defender a rede contra essas ameaças ou, melhor ainda, criar novas arquiteturas imunes a tais ameaças em primeiro lugar. Dadas a frequência e a variedade das ameaças existentes, bem como o perigo de novos e mais destrutivos futuros ataques, a segurança de rede se tornou, recentemente, um assunto principal na área de redes de computadores. Um dos objetivos deste livro é trazer as questões de segurança de rede para primeiro plano. Iniciaremos nossa discussão sobre redes de computadores nesta seção, com uma breve descrição de alguns dos ataques mais predominantes e prejudiciais na Internet de hoje. Então, à medida que retratarmos em detalhes as diversas tecnologias de redes de computadores e protocolos nos capítulos seguintes, analisaremos as diversas questões relacionadas à segurança associadas a essas tecnologias e protocolos. Finalmente, no Capítulo 8, munidos com nosso know-how em redes de computadores e protocolos da Internet recém-adquirido, estudaremos a fundo como as redes de computadores podem ser defendidas contra ameaças, ou projetadas e operadas para impossibilitá-las.

Visto que ainda não temos o know-how em rede de computadores e em protocolos da Internet, começaremos com uma análise de alguns dos atuais problemas mais predominantes relacionados à segurança. Isto irá aguçar nosso apetite para mais discussões importantes nos capítulos futuros. Começamos com a pergunta, o que pode dar errado? Como as redes de computadores são frágeis? Quais são alguns dos tipos de ameaças mais predominantes hoje?

Os vilões podem colocar "malware" em seu hospedeiro através da Internet

Conectamos aparelhos à Internet porque queremos receber/enviar dados de/para a Internet. Isso inclui todos os tipos de recursos vantajosos, como páginas da Web, mensagens de e-mail, MP3, chamadas telefônicas, vídeo em tempo real, resultados de mecanismo de busca etc. Porém, infelizmente, junto com esses recursos vantajosos aparecem os maliciosos — conhecidos conjuntamente como malware — que também podem entrar e infectar nossos aparelhos. Uma vez que o malware infecta nosso aparelho, ele é capaz de fazer coisas tortuosas, como apagar nossos arquivos; instalar spyware que coleta nossas informações particulares, como nosso número de cartão de crédito, senhas e combinação de teclas, e as envia (através da Internet, é claro!) de volta aos vilões. Nossa hospedeiro comprometido pode estar, também, envolvido em uma rede de milhares de aparelhos comprometidos, conhecidos como "botnet", o qual é controlado e influenciado pelos vilões para distribuição de spams ou ataques de recusa de serviço distribuídos (que serão brevemente discutidos) contra hospedeiros direcionados.

Muitos malwares existentes hoje são autorreprodutivos: uma vez que infecta um hospedeiro, a partir deste, ele faz a busca por entradas em outros hospedeiros através da Internet, e a partir de hospedeiros recém-infectados, ele faz a busca por entrada em mais hospedeiros. Desta maneira, o malware autorreprodutivo pode se disseminar rapidamente. Por exemplo, o número de aparelhos infectados pelo worm 2003 Saphire/Slammer dobrou a cada 8,5 segundos nos primeiros minutos após seu ataque, infectando mais de 90 por cento dos hospedeiros frágeis em 10 minutos [Moore, 2003]. O malware pode se espalhar na forma de vírus, worms ou cavalo de Troia [Skoudis, 2004]. Os vírus são malwares que necessitam de uma interação do usuário para infectar seu aparelho. O exemplo clássico é um anexo de e-mail contendo um código executável malicioso. Se o usuário receber e abrir tal anexo, o malware será executado em seu aparelho. Geralmente, tais vírus de e-mail se autorreproduzem: uma vez executado, o vírus pode enviar uma mensagem idêntica, com um anexo malicioso idêntico, para, por exemplo, todos os contatos da lista de endereços do usuário. Worms (como o Slammer) são malwares capazes de entrar em um aparelho sem qualquer interação do usuário. Por exemplo, um usuário pode estar executando uma aplicação de rede frágil para a qual um atacante pode enviar um malware. Em alguns casos, sem a intervenção do usuário, a aplicação pode aceitar o malware da Internet e executá-lo, criando um worm. Este, no aparelho recém-infectado, então, varre a Internet em busca de outros hospedeiros que estejam executando a mesma aplicação de rede frágil. Ao encontrar outros hospedeiros frágeis, ele envia uma cópia de si mesmo para eles. Por fim, um cavalo de Troia é uma parte oculta de um software funcional. Hoje, o malware é persuasivo e é caro para se criar uma proteção. À medida que trabalhar com este livro, sugerimos que pense na seguinte questão: O que os projetistas de computadores podem fazer para proteger os aparelhos que utilizam a Internet de ameaças de malware?

Os vilões pode atacar servidores e infraestrutura de redes

Um amplo grupo de ameaças à segurança pode ser classificado como ataques de recusa de serviços (DoS). Como o nome sugere, um ataque DoS torna uma rede, hospedeiro ou outra parte da infraestrutura inutilizável por usuários verdadeiros. Servidores da Web, de e-mails e DNS (discutidos no Capítulo 2), e redes institucionais podem estar sujeitos aos ataques DoS. Na Internet, esses ataques são extremamente comuns, com milhares deles ocorrendo todo ano [Moore, 2001; Mirkovic, 2005]. A maioria dos ataques DoS na Internet podem ser divididos em três categorias:

Ataque de vulnerabilidade. Envolve o envio de mensagens perfeitas a uma aplicação vulnerável ou a um sistema operacional sendo executado em um hospedeiro direcionado. Se a sequência correta de pacotes é enviada a uma aplicação vulnerável ou a um sistema operacional, o serviço pode parar ou, pior, o hospedeiro pode pifar.

Inundação na largura de banda. O atacante envia um grande número de pacotes ao hospedeiro direcionado — tantos pacotes que o enlace de acesso do alvo se entope, impedindo os pacotes legítimos de alcançarem o servidor.

Inundação na conexão. O atacante estabelece um grande número de conexões TCP semiabertas ou abertas (as conexões TCP são discutidas no Capítulo 3) no hospedeiro-alvo. O hospedeiro pode ficar tão atolado com essas conexões falsas que para de aceitar conexões legítimas.

Vamos agora explorar mais detalhadamente o ataque de inundação na largura de banda. Lembrando de nossa análise sobre atraso e perda na seção 1.4.2, é evidente que se o servidor possui uma taxa de acesso R bps, então o atacante precisará enviar tráfego a uma taxa de, aproximadamente, R bps para causar dano. Se R for muito grande, uma fonte de ataque única pode não ser capaz de gerar tráfego suficiente para prejudicar o servidor. Além disso, se todo o tráfego provier de uma fonte única, um roteador upstream pode conseguir detectar o ataque e bloquear todo o tráfego da fonte antes que ele se aproxime do servidor. Em um ataque DoS distribuído (DDoS), ilustrado na Figura 1.25, o atacante controla múltiplas fontes que sobrecarregam o alvo. Com tal abordagem, a taxa de tráfego agregada por todas as fontes controladas precisa ser, aproximadamente, R para incapacitar o serviço. Os ataques DDoS que potencializam botnets com centenas de hospedeiros comprometidos são uma ocorrência comum hoje em dia [Mirkovic, 2005]. Os ataques DDoS são muito mais difíceis de detectar e de prevenir do que um ataque DoS de um único hospedeiro.

Os vilões podem analisar pacotes

Muitos usuários hoje acessam à Internet por meio de aparelhos sem fio, como laptops conectados à tecnologia Wi-Fi ou aparelhos portáteis com conexões à Internet via telefone celular (abordado no Capítulo 6). Enquanto o acesso onipresente à Internet é extremamente conveniente e disponibiliza novas aplicações sensacionais aos usuários móveis, ele também cria uma grande vulnerabilidade de segurança — posicionando um receptor passivo nas proximidades do transmissor sem fio, o receptor pode obter uma cópia de cada pacote transmitido! Esses pacotes podem conter todo tipo de informações confidenciais, incluindo senhas, número de inscrição da previdência social, segredos comerciais e mensagens pessoais. Um receptor passivo que grava uma cópia de cada pacote que passa é denominado analisador de pacote.

Os analisadores também podem estar distribuídos em ambientes de conexão com fio. Nesses ambientes, como em muitas Ethernet LANs, um analisador de pacote pode obter cópias de todos os pacotes enviados pela LAN. Como descrito na Seção 1.2, as tecnologias de acesso a cabo também transmitem pacotes e são, dessa forma, vulneráveis à análise. Além disso, um vilão que quer ganhar acesso ao roteador de acesso de uma instituição ou enlace de acesso para a Internet pode instalar um analisador que faça uma cópia de cada pacote que vai para/de a empresa. Os pacotes farejados podem, então, ser analisados off-line em busca de informações confidenciais.

O software para analisar pacotes está disponível gratuitamente em diversos sites da Internet e em produtos comerciais. Professores que ministram um curso de redes passam exercícios que envolvem a composição de um programa de reconstrução de dados da camada de aplicação e um programa analisador de pacotes.

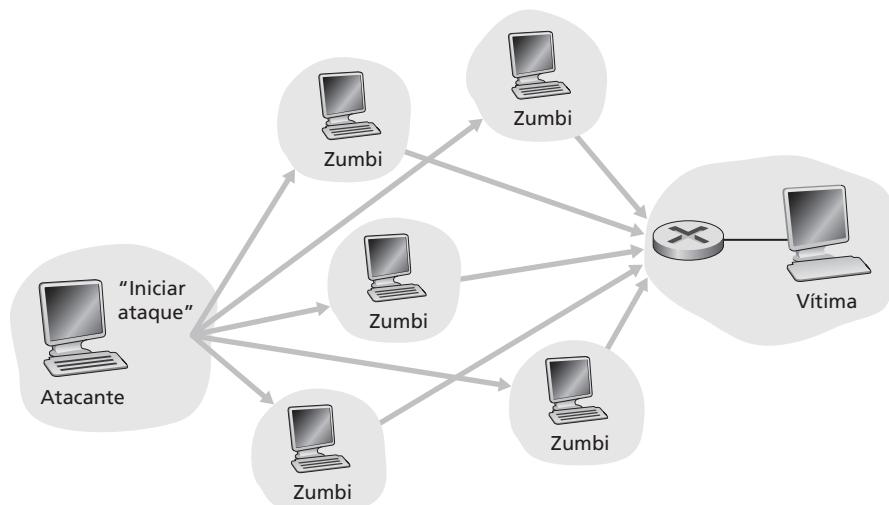


Figura 1.25 Um ataque de recusa de serviço distribuído (DDoS)

Como os analisadores de pacote são passivos — ou seja, eles não introduzem pacotes no canal — eles são difíceis de detectar. Portanto, quando enviamos pacotes para um canal sem fio, devemos aceitar a possibilidade de que alguém possa estar copiando nossos pacotes. Como você deve ter imaginado, uma das melhores defesas contra a análise de pacote envolve a criptografia, que será analisada no Capítulo 8, já que se aplica à segurança de rede.

Os vilões podem se passar por alguém de sua confiança

Por incrível que pareça, é extremamente fácil (você saberá como conforme ler este livro!) criar um pacote com um endereço de fonte arbitrário, conteúdo de pacote e um endereço de destino e, então, transmitir esse pacote feito à mão para a Internet, que, obedientemente, o encaminhará para seu destino. Imagine que um receptor inocente (digamos um roteador da Internet) que recebe tal pacote, acredita que a fonte (falsa) seja confiável e então executa um comando integrado aos conteúdos do pacote (digamos que modifica sua base de encaminhamento). A habilidade de introduzir pacotes na Internet com uma fonte falsa de endereço é conhecida como IP spoofing, e é uma das muitas maneiras pelas quais o usuário pode se passar por outro.

Para resolver esse problema, precisaremos de uma comprovação da fonte, ou seja, um mecanismo que nos permitirá determinar com certeza se uma mensagem se origina de onde pensamos. Mais uma vez, sugerimos que pense em como isso pode ser feito em aplicações de rede e protocolos à medida que avança sua leitura pelos capítulos deste livro. Exploraremos mais mecanismos para comprovação da fonte no Capítulo 8.

Os vilões podem alterar ou excluir mensagens

Encerramos esta breve análise de ataques na rede descrevendo os ataques *man-in-the-middle* (homem no meio). Nessa categoria de ataques, o vilão está infiltrado no percurso da comunicação entre duas entidades comunicantes. Vamos chamar essas entidades de Alice e Bob, que podem ser seres humanos reais ou entidades de rede como dois roteadores ou dois servidores de e-mail. O vilão pode ser, por exemplo, um roteador comprometido no percurso da comunicação, ou um módulo de software residente em um dos hospedeiros finais em uma camada inferior da pilha de protocolo. No ataque *man-in-the-middle*, o vilão não somente possui a possibilidade de analisar todos os pacotes que passam entre Bob e Alice, como também pode introduzir, alterar ou excluir pacotes. No jargão da segurança de rede, um ataque *man-in-the-middle* pode comprometer a integridade dos dados enviados entre Alice e Bob. Assim como veremos no Capítulo 8, os mecanismos que oferecem sigilo (proteção contra análise) e autenticação da origem (permitindo que o receptor verifique com certeza o gerador da mensagem) não necessariamente oferecem integridade dos dados.

Ao encerrar esta seção, deve-se considerar como a Internet se tornou um local inseguro, em primeiro lugar. A resposta resumidamente é que a Internet foi, a princípio, criada para ser assim, baseada no modelo de “um grupo de usuários de confiança mútua ligados a uma rede transparente” [Blumenthal, 2001] — um modelo no qual (por definição) não há necessidade de segurança. Muitos aspectos da arquitetura inicial da Internet refletem profundamente essa noção de confiança mútua. Por exemplo, a capacidade de um usuário enviar um pacote a qualquer outro é mais uma falha do que um recurso solicitado/concedido, e acredita-se piamente na identidade do usuário em vez de ela ser confirmada automaticamente.

Mas a Internet de hoje certamente não envolve “usuários de confiança mútua”. Contudo, os usuários atuais ainda precisam se comunicar mesmo quando não confiam um no outro, podem se comunicar anonimamente, podem se comunicar indiretamente através de terceiros (por exemplo, Web caches, que serão estudados no Capítulo 2, ou agentes móveis para assistência, que serão estudados no Capítulo 6), e podem desconfiar do hardware, software e até mesmo do ar pelo qual eles se comunicam. Temos agora muitos desafios relacionados à segurança perante nós à medida que prosseguimos com o livro: devemos procurar por proteção contra a análise, disfarce da origem, ataques *man-in-the-middle*, ataques DDoS, malware e mais. Devemos manter em mente que a comunicação entre usuários de confiança mútua é mais uma exceção do que uma regra.

Seja bem-vindo ao mundo da moderna rede de computadores!

1.7 História das redes de computadores e da Internet

Da seção 1.1 à 1.6, apresentamos um panorama da tecnologia de redes de computadores e da Internet. Agora, você já deve saber o suficiente para impressionar sua família e amigos. Contudo, se realmente quiser ser o maior sucesso do próximo coquetel, você deve rechear seu discurso com pérolas da fascinante história da Internet [Segaller, 1998].

1.7.1 Desenvolvimento da comutação de pacotes: 1961-1972

Os primeiros passos da disciplina de redes de computadores e da Internet podem ser traçados desde o início da década de 1960, quando a rede telefônica era a rede de comunicação dominante no mundo inteiro. Lembre-se de que na Seção 1.3 dissemos que a rede de telefonia usa comutação de circuitos para transmitir informações entre uma origem e um destino — uma escolha acertada, já que a voz é transmitida a uma taxa constante entre a origem e o destino. Dada a importância cada vez maior (e o alto custo) dos computadores no início da década de 1960 e o advento de computadores com multiprogramação (*time-sharing*), nada seria mais natural (agora que temos uma visão perfeita do passado) do que considerar a questão de como interligar computadores para que pudessem ser compartilhados entre usuários distribuídos em localizações geográficas diferentes. O tráfego gerado por esses usuários provavelmente era intermitente, por *rajadas* — períodos de atividade, como o envio de um comando a um computador remoto, seguidos de períodos de inatividade, como a espera por uma resposta ou o exame de uma resposta recebida.

Três grupos de pesquisa ao redor do mundo, sem que nenhum tivesse conhecimento do trabalho do outro [Leiner, 1998], começaram a inventar a comutação de pacotes como uma alternativa poderosa e eficiente à de circuitos. O primeiro trabalho publicado sobre técnicas de comutação de pacotes foi o de Leonard Kleinrock [Kleinrock, 1961, 1964], que, naquela época, era um doutorando do MIT. Usando a teoria de filas, o trabalho de Kleinrock demonstrou, com elegância, a eficácia da abordagem da comutação de pacotes para fontes de tráfego intermitentes (*em rajadas*). Em 1964, Paul Baran [Baran, 1964], do Rand Institute, começou a investigar a utilização de comutação de pacotes na transmissão segura de voz pelas redes militares, ao mesmo tempo que Donald Davies e Roger Scantlebury desenvolviam suas ideias sobre esse assunto no National Physical Laboratory, na Inglaterra.

Os trabalhos desenvolvidos no MIT, no Rand Institute e no National Physical Laboratory foram os alicerces do que hoje é a Internet. Mas a Internet também tem uma longa história de atitudes do tipo “construir e demonstrar”, que também data do início da década de 1960. J. C. R. Licklider [DEC, 1990] e Lawrence Roberts, ambos colegas de Kleinrock no MIT, foram adiante e lideraram o programa de ciência de computadores na ARPA (Advanced Research Projects Agency — Agência de Projetos de Pesquisa Avançada), nos Estados Unidos. Roberts publicou um plano geral para a ARPAnet [Roberts, 1967], a primeira rede de computadores por comutação de pacotes e uma ancestral direta da Internet pública de hoje. Os primeiros comutadores de pacotes eram conhecidos como processadores de mensagens de interface (*interface message processors* — IMPs), e o contrato para a fabricação desses comutadores foi entregue à empresa BBN. Em 1969, no Dia do Trabalho nos Estados Unidos, foi instalado o primeiro IMP na UCLA (Universidade da Califórnia em Los Angeles) sob a supervisão de Kleinrock. Logo em seguida foram instalados três IMPs adicionais no Stanford Research Institute (SRI), na Universidade da Califórnia em Santa Bárbara e na Universidade de Utah (Figura 1.26).

O incipiente precursor da Internet tinha quatro nós no final de 1969. Kleinrock recorda que a primeiríssima utilização da rede foi fazer um login remoto entre a UCLA e o SRI, derrubando o sistema [Kleinrock, 2004].

Em 1972, a ARPAnet tinha aproximadamente 15 nós e foi apresentada publicamente pela primeira vez por Robert Kahn na Conferência Internacional sobre Comunicação por Computadores (International Conference on Computer Communications) daquele ano. O primeiro protocolo fim a fim entre sistemas finais da ARPAnet, conhecido como protocolo de controle de rede (*network-control protocol* — NCP), estava concluído [RFC 001] e a partir desse momento a escrita de aplicações tornou-se possível. Em 1972, Ray Tomlinson, da BBN, escreveu o primeiro programa de e-mail.



Figura 1.26 Um dos primeiros processadores de mensagens de interface (IMP) e L. Kleinrock (Mark J. Terrill, AP/Wide World Photos)

1.7.2 Redes proprietárias e trabalho em rede: 1972-1980

A ARPAnet inicial era um rede isolada, fechada. Para se comunicar com uma máquina da ARPAnet, era preciso estar ligado a um outro IMP dessa rede. Do início a meados de 1970, surgiram novas redes independentes de comutação de pacotes:

- ALOHAnet, uma rede de micro-ondas ligando universidades das ilhas do Havaí [Abramson, 1970], bem como as redes de pacotes por satélite [RFC 829] e por rádio [Kahn, 1978] da DARPA [Kahn, 1978];
- Telenet, uma rede comercial de comutação de pacotes da BBN fundamentada na tecnologia ARPAnet;
- Cyclades, uma rede de comutação de pacotes pioneira na França, montada por Louis Pouzin [Think, 2002];
- redes de tempo compartilhado como a Tymnet e a rede GE Information Services, entre outras que surgiram no final da década de 1960 e início da década de 1970 [Schwartz, 1977];
- rede SNA da IBM (1969–1974), cujo trabalho comparava-se ao da ARPAnet [Schwartz, 1977].

O número de redes estava crescendo. Hoje, com perfeita visão do passado, podemos perceber que aquela era a hora certa para desenvolver uma arquitetura abrangente para conectar redes. O trabalho pioneiro de interconexão de redes, sob o patrocínio da DARPA (Defense Advanced Research Projects Agency — Agência de Projetos de Pesquisa Avançada de Defesa), criou em essência *uma rede de redes* e foi realizado por Vinton Cerf e Robert Kahn [Cerf, 1974]; o termo *internettting* foi cunhado para descrever esse trabalho.

Esses princípios de arquitetura foram incorporados ao TCP. As primeiras versões desse protocolo, contudo, eram muito diferentes do TCP de hoje. Aquelas versões combinavam uma entrega sequencial confiável de dados via retransmissão por sistema final (que ainda faz parte do TCP de hoje) com funções de envio (que hoje são desempenhadas pelo IP). As primeiras experiências com o TCP, combinadas com o reconhecimento da importância de um serviço de transporte fim a fim não confiável, sem controle de fluxo, para aplicações como voz em pacotes, levaram à separação entre IP e TCP e ao desenvolvimento do protocolo UDP. Os três protocolos fundamentais da Internet que temos hoje — TCP, UDP e IP — estavam conceitualmente disponíveis no final da década de 1970.

Além das pesquisas sobre a Internet realizadas pela DARPA, muitas outras atividades importantes relacionadas ao trabalho em rede estavam em curso. No Havaí, Norman Abramson estava desenvolvendo a ALOHAnet, uma rede de pacotes por rádio que permitia que vários lugares remotos das ilhas havaianas se comunicassem entre si. O ALOHA [Abramson, 1970] foi o primeiro protocolo de acesso múltiplo que permitiu que usuários distribuídos em diferentes localizações geográficas compartilhassem um único meio de comunicação broadcast (uma frequência de rádio). O trabalho de Abramson sobre protocolo de múltiplo acesso foi aprimorado por Metcalfe e Boggs com o desenvolvimento do protocolo Ethernet [Metcalfe, 1976] para redes compartilhadas de transmissão broadcast por fio; veja a Figura 1.27.

O interessante é que o protocolo Ethernet de Metcalfe e Boggs foi motivado pela necessidade de conectar vários PCs, impressoras e discos compartilhados [Perkins, 1994]. Há 25 anos, bem antes da revolução do PC e da explosão das redes, Metcalfe e Boggs estavam lançando as bases para as LANs de PCs de hoje. A tecnologia Ethernet representou uma etapa importante para o trabalho em redes interconectadas. Cada rede Ethernet local era, em si, uma rede, e, à medida que o número de LANs aumentava, a necessidade de interconectar essas redes foi se tornando cada vez mais importante. Discutiremos detalhadamente a tecnologia Ethernet, ALOHA e outras tecnologias de LAN no Capítulo 5.

1.7.3 Proliferação de redes: 1980-1990

Ao final da década de 1970, aproximadamente 200 máquinas estavam conectadas à ARPAnet. Ao final da década de 1980, o número de máquinas ligadas à Internet pública, uma confederação de redes muito parecida com a Internet de hoje, alcançaria cem mil. A década de 1980 seria uma época de formidável crescimento.

Grande parte daquele crescimento foi consequência de vários esforços distintos para criar redes de computadores para interligar universidades. A BITNET processava e-mails e fazia transferência de arquivos entre diversas universidades do nordeste dos Estados Unidos. A CSNET (computer science network — rede da ciência de com-

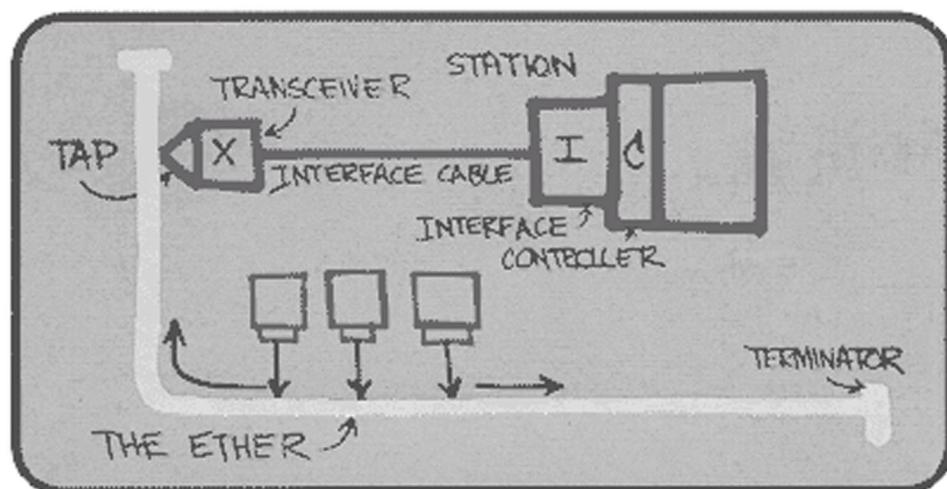


Figura 1.27 A concepção original de Metcalfe para a Ethernet

putadores) foi formada para interligar pesquisadores de universidades que não tinham acesso à ARPAnet. Em 1986, foi criada a NSFNET para prover acesso a centros de supercomputação patrocinados pela NSF. Partindo de uma velocidade inicial de 56 kbps, ao final da década o backbone da NSFNET estaria funcionando a 1,5 Mbps e servindo como backbone primário para a interligação de redes regionais.

Na comunidade da ARPAnet, já estavam sendo encaixados muitos dos componentes finais da arquitetura da Internet de hoje. No dia 1º de janeiro de 1983, o TCP/IP foi adotado oficialmente como o novo padrão de protocolo de máquinas para a ARPAnet (em substituição ao protocolo NCP). Devido à importância do evento, o dia da transição do NCP para o TCP/IP [RFC 801] foi marcado com antecedência — a partir daquele dia todas as máquinas tiveram de adotar o TCP/IP. No final da década de 1980, foram agregadas importantes extensões ao TCP para implementação do controle de congestionamento baseado em hospedeiros [Jacobson, 1988]. Também foi desenvolvido o sistema de nomes de domínios (DNS) utilizado para mapear nomes da Internet fáceis de entender (por exemplo, *gaia.cs.umass.edu*) para seus endereços IP de 32 bits [RFC 1034]. Paralelamente ao desenvolvimento da ARPAnet (que em sua maior parte deve-se aos Estados Unidos), no início da década de 1980 os franceses lançaram o projeto Minitel, um plano ambicioso para levar as redes de dados para todos os lares. Patrocinado pelo governo francês, o sistema Minitel consistia em uma rede pública de comutação de pacotes (baseada no conjunto de protocolos X.25, que usava circuitos virtuais), servidores Minitel e terminais baratos com modems de baixa velocidade embutidos. O Minitel transformou-se em um enorme sucesso em 1984, quando o governo francês forneceu, gratuitamente, um terminal para toda residência francesa que quisesse. O sistema Minitel incluía sites de livre acesso — como o da lista telefônica — e também sites particulares, que cobravam uma taxa de cada usuário baseada no tempo de utilização. No seu auge, em meados de 1990, o Minitel oferecia mais de 20 mil serviços, que iam desde home banking até bancos de dados especializados para pesquisa. Era usado por mais de 20 por cento da população da França, gerava receita de mais de um bilhão de dólares por ano e criou dez mil empregos. Estava presente em grande parte dos lares franceses dez anos antes de a maioria dos norte-americanos ouvir falar de Internet.

1.7.4 A explosão da Internet: a década de 1990

A década de 1990 estreou com vários eventos que simbolizaram a evolução contínua e a comercialização iminente da Internet. A ARPAnet, a progenitora da Internet, deixou de existir. Durante a década de 1980, a MILNET e a Defense Data Network (Rede de Dados de Defesa) cresceram e passaram a carregar a maior parte do tráfego do Departamento de Defesa dos Estados Unidos e a NSFNET começou a servir como uma rede de backbone conectando redes regionais nos Estados Unidos com nacionais no exterior. Em 1991, a NSFNET extinguiu as restrições que impunha à sua utilização com finalidades comerciais, mas, em 1995, perderia seu mandato quando o tráfego de backbone da Internet passou a ser carregado por provedores de serviços de Internet.

O principal evento da década de 1990, no entanto, foi o surgimento da World Wide Web, que levou a Internet para os lares e as empresas de milhões de pessoas no mundo inteiro. A Web serviu também como plataforma para a habilitação e a disponibilização de centenas de novas aplicações, inclusive negociação de ações e serviços bancários on-line, serviços multimídia em tempo real e serviços de recuperação de informações. Para um breve histórico dos primórdios da Web, consulte [W3C, 1995].

A Web foi inventada no CERN (European Center for Nuclear Physics — Centro Europeu para Física Nuclear) por Tim Berners-Lee entre 1989 e 1991 [Berners-Lee, 1989], com base em ideias originadas de trabalhos anteriores sobre hipertexto realizados por Bush [Bush, 1945], na década de 1940, e por Ted Nelson [Ziff-Davis, 1998], na década de 1960. Berners-Lee e seus companheiros desenvolveram versões iniciais de HTML, HTTP, um servidor para a Web e um browser — os quatro componentes fundamentais da Web. Os browsers originais do CERN ofereciam apenas uma interface de linha de comando. Perto do final de 1992 havia aproximadamente 200 servidores Web em operação, e esse conjunto de servidores era apenas uma amostra do que estava por vir. Nessa época, vários pesquisadores estavam desenvolvendo browsers da Web com interfaces GUI (*graphical user interface* — interface gráfica de usuário), entre eles Marc Andreessen, que liderou o desenvolvimento do popular browser Mosaic. Em 1994, Marc Andreessen e Jim Clark formaram a Mosaic Communications, que mais

tarde se transformou na Netscape Communications Corporation [Cusumano, 1998; Quittner, 1998]. Em 1995, estudantes universitários estavam usando browsers Mosaic e Netscape para navegar na Web diariamente. Nessa época, empresas — grandes e pequenas — começaram a operar servidores Web e a realizar transações comerciais pela Web. Em 1996, a Microsoft começou a fabricar browsers, dando início à guerra dos browsers entre Netscape e Microsoft, vencida pela última alguns anos mais tarde.

A segunda metade da década de 1990 foi um período de tremendo crescimento e inovação para a Internet, com grandes corporações e milhares de novas empresas criando produtos e serviços para a Internet. O correio eletrônico pela Internet (e-mail) continuou a evoluir com leitores ricos em recursos provendo agendas de endereços, anexos, hot links e transporte de multimídia. No final do milênio a Internet dava suporte a centenas de aplicações populares, entre elas quatro de enorme sucesso:

- e-mail, incluindo anexos e correio eletrônico com acesso pela Web;
- a Web, incluindo navegação pela Web e comércio pela Internet;
- serviço de mensagem instantânea, com listas de contato, cujo pioneiro foi o ICQ;
- compartilhamento peer-to-peer de arquivos MP3, cujo pioneiro foi o Napster.

O interessante é que as duas primeiras dessas aplicações de sucesso arrasador vieram da comunidade de pesquisas, ao passo que as duas últimas foram criadas por alguns jovens empreendedores.

No período de 1995 a 2001, a Internet realizou uma viagem vertiginosa nos mercados financeiros. Antes mesmo de se mostrarem lucrativas, centenas de novas empresas da Internet faziam suas ofertas públicas iniciais de ações e começavam a ser negociadas em bolsas de valores. Muitas empresas eram avaliadas em bilhões de dólares sem ter nenhum fluxo significativo de receita. As ações da Internet sofreram uma queda também vertiginosa em 2000-2001, e muitas novas empresas fecharam. Não obstante, várias empresas surgiram como grandes vencedoras no mundo da Internet (mesmo que os preços de suas ações tivessem sofrido com aquela queda), entre elas Microsoft, Cisco, Yahoo, e-Bay, Google e Amazon.

1.7.5 Desenvolvimentos recentes

A inovação na área de redes de computadores continua a passos largos. Há progressos em todas as frentes, incluindo desenvolvimento de novas aplicações, distribuição de conteúdo, telefonia por Internet, velocidades de transmissão mais altas em LANs e roteadores mais rápidos. Mas três desenvolvimentos merecem atenção especial: a proliferação de redes de acesso de alta velocidade (incluindo acesso sem fio), a segurança e as redes P2P.

Como discutimos na Seção 1.2, a penetração cada vez maior do acesso residencial de banda larga à Internet via modem a cabo e DSL montou o cenário para uma profusão de novas aplicações multimídia, entre elas voz e vídeo sobre IP [Skype, 2009], compartilhamento de vídeo [Youtube, 2009] e televisão sobre IP [PPLive, 2009]. A crescente onipresença de redes Wi-Fi públicas de alta velocidade (11 Mbps e maiores) e de acesso de média velocidade (centenas de kbps) à Internet por redes de telefonia celular não está apenas possibilitando conexão constante, mas também habilitando um novo conjunto muito interessante de serviços específicos para localizações determinadas. Abordaremos redes sem fio e mobilidade no Capítulo 6.

Em seguida a uma série de ataques de recusa de serviço em importantes servidores Web no final da década de 1990 e à proliferação de ataques de worms (por exemplo, o Blaster) que infectam sistemas finais e emperram a rede com tráfego excessivo, a segurança da rede tornou-se uma questão extremamente importante. Esses ataques resultaram no desenvolvimento de sistemas de detecção de intrusão capazes de prevenir ataques com antecedência, na utilização de firewalls para filtrar tráfego indesejado antes que entre na rede. Abordaremos vários tópicos importantes relacionados à segurança no Capítulo 8.

A última inovação que queremos destacar são as redes P2P. Uma aplicação de rede P2P explora os recursos de computadores de usuários — armazenagem, conteúdo, ciclos de CPU e presença humana — e tem significativa autonomia em relação a servidores centrais. A conectividade dos computadores de usuários (isto é, dos pares, ou peers) normalmente é intermitente. Há alguns anos, houve diversas histórias de sucesso com o P2P, incluindo o compartilha-

mento de arquivo P2P (Napster, Kazaa, Gnutella, eDonkey, Lime Wire etc.), distribuição de arquivos (BitTorrent), Voz sobre IP (Skype) e IPTV (PPLive, ppStream). Muitas dessas aplicações P2P serão discutidas no Capítulo 2.

1.8 Resumo

Neste capítulo, abordamos uma quantidade imensa de assuntos. Examinamos as várias peças de hardware e software que compõem a Internet, em especial, e redes de computadores, em geral. Começamos pela periferia da rede, examinando sistemas finais e aplicações, além do serviço de transporte fornecido às aplicações que executam nos sistemas finais. Também vimos as tecnologias de camada de enlace e meio físico normalmente encontrados na rede de acesso. Em seguida, mergulhamos no interior da rede e chegamos ao seu núcleo, identificando comutação de pacotes e comutação de circuitos como as duas abordagens básicas do transporte de dados por uma rede de telecomunicações, expondo os pontos fortes e fracos de cada uma delas. Examinamos, então, as partes inferiores (do ponto de vista da arquitetura) da rede — as tecnologias de camada de enlace e os meios físicos comumente encontrados na rede de acesso. Examinamos também a estrutura da Internet global e aprendemos que ela é uma rede de redes. Vimos que a estrutura hierárquica da Internet, composta de ISPs de níveis mais altos e mais baixos, permitiu que ela se expandisse e incluísse milhares de redes.

Na segunda parte deste capítulo introdutório, abordamos diversos tópicos fundamentais da área de redes de computadores. Primeiramente examinamos as causas de atrasos e perdas de pacotes em uma rede de comutação de pacotes. Desenvolvemos modelos quantitativos simples de atrasos de transmissão, de propagação e de fila; esses modelos de atrasos serão muito usados nos problemas propostos em todo o livro. Em seguida examinamos camadas de protocolo e modelos de serviço, princípios fundamentais de arquitetura de redes aos quais voltaremos a nos referir neste livro. Analisamos também alguns dos ataques mais comuns na Internet atualmente. Terminamos nossa introdução sobre redes com uma breve história das redes de computadores. O primeiro capítulo constitui um minicurso sobre redes de computadores.

Portanto, percorremos realmente um extraordinário caminho neste primeiro capítulo! Se você estiver um pouco assustado, não se preocupe. Abordaremos todas essas ideias em detalhes nos capítulos seguintes (é uma promessa, e não uma ameaça!). Por enquanto, esperamos que, ao encerrar este capítulo, você tenha adquirido uma noção, ainda que incipiente, das peças que formam uma rede, um domínio ainda em desenvolvimento do vocabulário (não se acanhe de voltar aqui para consulta) e um desejo cada vez maior de aprender mais sobre redes. Esta é a tarefa que nos espera no restante deste livro.

O guia deste livro

Antes de iniciarmos qualquer viagem, sempre é bom consultar um guia para nos familiarizar com as estradas principais e desvios que encontraremos pela frente. O destino final da viagem que estamos prestes a empreender é um entendimento profundo do como, do quê e do porquê das redes de computadores. Nossa guia é a sequência de capítulos deste livro:

1. Redes de computadores e a Internet
2. Camada de aplicação
3. Camada de transporte
4. Camada de rede
5. Camada de enlace e redes locais (LANs)
6. Sem fio e redes móveis
7. Redes multimídia
8. Segurança em redes de computadores
9. Gerenciamento de rede

Os capítulos 2 a 5 são os quatro capítulos centrais deste livro. Note que esses capítulos estão organizados segundo as quatro camadas superiores da pilha de cinco camadas de protocolos da Internet, com um capítulo para

cada camada. Note também que nossa jornada começará no topo da pilha de protocolos da Internet, a saber, a camada de aplicação, e prosseguirá daí para baixo. O princípio racional que orienta essa jornada de cima para baixo é que, entendidas as aplicações, podemos compreender os serviços de rede necessários para dar suporte a elas. Então, poderemos examinar, um por um, os vários modos como esses serviços poderiam ser implementados por uma arquitetura de rede. Assim, o estudo das aplicações logo no início dá motivação para o restante do livro.

A segunda metade deste livro — capítulos 6 a 9 — aborda quatro tópicos extremamente importantes (e de certa maneira independentes) de redes modernas. No Capítulo 6, examinamos tecnologia sem fio e mobilidade, incluindo LANs sem fio (Wi-Fi, WiMAX e Bluetooth), redes de telefonia celular (GSM) e mobilidade (nas redes IP e GSM). No Capítulo 7 (Redes multimídia), examinamos aplicações de áudio e vídeo, como telefone por Internet, videoconferência e recepção de mídia armazenada. Examinamos também como uma rede de comutação de pacotes pode ser projetada para prover serviço de qualidade consistente para aplicações de áudio e vídeo. No Capítulo 8 (Segurança em redes de computadores), analisamos, primeiramente, os fundamentos da criptografia e da segurança de redes e, em seguida, de que modo a teoria básica está sendo aplicada a um amplo leque de contextos da Internet. No último capítulo (Gerenciamento de redes), examinamos as questões fundamentais do gerenciamento de redes, bem como os protocolos primários da Internet utilizados para esse fim.



Exercícios de fixação

Capítulo 1 Questões de revisão

Seção 1.1

- Qual é a diferença entre um hospedeiro e um sistema final? Cite os tipos de sistemas finais. Um servidor Web é um sistema final?
- A palavra protocolo é muito usada para descrever relações diplomáticas. Dê um exemplo de um protocolo diplomático.

Seção 1.2

- O que é um programa cliente? O que é um programa servidor? Um programa servidor requisita e recebe serviços de um programa cliente?
- Cite seis tecnologias de acesso. Classifique cada uma delas nas categorias acesso residencial, acesso corporativo ou acesso móvel.
- A taxa de transmissão HFC é dedicada ou é compartilhada entre usuários? É possível haver colisões na direção provedor-usuário de um canal HFC? Por quê?
- Cite seis tecnologias de acesso residencial disponíveis em sua cidade. Para cada tipo de acesso, apresente a taxa downstream, a taxa upstream e o preço mensal anunciados.
- Qual é a taxa de transmissão de LANs Ethernet? Para uma dada taxa de transmissão, cada usuário da LAN pode transmitir continuamente a essa taxa?

- Cite alguns meios físicos utilizados para instalar a Ethernet.
- Modems discados, HFC e ADSL são usados para acesso residencial. Para cada uma dessas tecnologias de acesso, cite uma faixa de taxas de transmissão e comente se a largura de banda é compartilhada ou dedicada.
- Descreva as tecnologias de acesso sem fio mais populares atualmente. Faça uma comparação entre elas.

Seção 1.3

- Qual é a vantagem de uma rede de comutação de circuitos em relação a uma de comutação de pacotes? Quais são as vantagens da TDM sobre a FDM em uma rede de comutação de circuitos?
- Por que se afirma que a comutação de pacotes emprega multiplexação estatística? Compare a multiplexação estatística com a multiplexação que ocorre em TDM.
- Suponha que exista exatamente um comutador de pacotes entre um computador de origem e um de destino. As taxas de transmissão entre a máquina de origem e o comutador e entre este e a máquina de destino são R_1 e R_2 , respectivamente. Admitindo que um roteador use comutação de pacotes do tipo armazena e reenvia, qual é o atraso total fim a fim para enviar um pacote de comprimento L ? (Desconsidere

formação de fila, atraso de propagação e atraso de processamento).

- 14.** Qual é principal diferença que distingue ISPs de nível 1 e de nível 2?

Suponha que usuários compartilhem um enlace de 2 Mbps e que cada usuário transmita continuamente a 1 Mbps, mas cada um deles transmite apenas 20 por cento do tempo. (Veja a discussão sobre multiplexação estatística na Seção 1.3).

- Quando a comutação de circuitos é utilizada, quantos usuários podem usar o enlace?
- Para o restante deste problema, suponha que seja utilizada a comutação de pacotes. Por que não haverá atraso de fila antes de um enlace se dois ou menos usuários transmitirem ao mesmo tempo? Por que haverá atraso de fila se três usuários transmitirem ao mesmo tempo?
- Determine a probabilidade de um dado usuário estar transmitindo.
- Suponha agora que haja três usuários. Determine a probabilidade de, a qualquer momento, os três usuários transmitirem simultaneamente. Determine a fração de tempo durante o qual a fila cresce.

Seção 1.4

- 16.** Considere o envio de um pacote de uma máquina de origem a uma de destino por uma rota fixa. Relacione os componentes do atraso que formam o atraso fim a fim. Quais deles são constantes e quais são variáveis?
- 17.** Visite o applet "Transmission versus Propagation Delay" no Companion Website. Entre as taxas, o atraso de propagação e os tamanhos de pacote disponíveis, determine uma combinação para a qual o emissor termine de transmitir antes que o primeiro bit do pacote chegue ao receptor.
- 18.** Quanto tempo um pacote de 1.000 bytes leva para se propagar através de um enlace de 2.500 km de distância, com uma velocidade de propagação de $2,5 \cdot 10^8$ m/s e uma taxa de transmissão de 2 Mbps? Geralmente, quanto tempo um pacote de comprimento L leva para se propagar através de um enlace de distância d , velocidade de propagação s , e taxa de transmissão de R bps? Esse atraso depende do comprimento do pacote? Esse atraso depende da taxa de transmissão?
- 19.** Suponha que o Hospedeiro A queira enviar um arquivo grande para o Hospedeiro B. O percurso do Hospedeiro A para o Hospedeiro B possui três enlaces, de taxas $R_1 = 500$ kbps, $R_2 = 2$ Mbps, e $R_3 = 1$ Mbps.

a. Considerando que não haja nenhum outro tráfego na rede, qual é a vazão para a transferência de arquivo?

- b.** Suponha que o arquivo tenha 4 milhões de bytes. Dividindo o tamanho do arquivo pela vazão, quanto tempo levará a transferência para o Hospedeiro B?
- c.** Repita os itens "a" e "b", mas agora com R_2 reduzido a 100 kbps.

- 20.** Suponha que o sistema final A queira enviar um arquivo grande para o sistema B. Em um nível muito alto, descreva como o sistema A cria pacotes a partir do arquivo. Quando um desses arquivos chegar ao comutador de pacote, quais informações no pacote o comutador utiliza para determinar o enlace através do qual o pacote é encaminhado? Por que a comutação de pacote na Internet é análoga a dirigir de uma cidade para outra pedindo informações ao longo do caminho?
- 21.** Visite o applet "Queuing and Loss" no Companion Website. Qual é a taxa de emissão máxima e a taxa de emissão mínima? Com essas taxas, qual é a intensidade do tráfego? Execute o applet com essas taxas e determine o tempo que leva a ocorrência de uma perda de pacote. Repita o procedimento uma segunda vez e determine novamente o tempo de ocorrência para a perda de pacote. Os resultados são diferentes? Por quê? Por que não?

Seção 1.5

- 22.** Cite cinco tarefas que uma camada pode executar. É possível que uma (ou mais) dessas tarefas seja(m) realizada(s) por duas (ou mais) camadas?
- 23.** Quais são as cinco camadas da pilha de protocolo da Internet? Quais as principais responsabilidades de cada uma dessas camadas?
- 24.** O que é uma mensagem de camada de aplicação? Um segmento de camada de transporte? Um datagrama de camada de rede? Um quadro de camada de enlace?
- 25.** Que camadas da pilha do protocolo da Internet um roteador implementa? Que camadas um comutador de camada de enlace implementa? Que camadas um sistema final implementa?

Seção 1.6

- 26.** Qual é a diferença entre um vírus, um worm e um cavalo de Troia?
- 27.** Descreva como pode ser criado uma botnet e como ela pode ser utilizada no ataque DDoS.



28. Suponha que Alice e Bob estejam enviando pacotes um para o outro através de uma rede de computadores e que Trudy se posicione na rede para que ela consiga capturar todos os pacotes enviados por Alice

e enviar o que quiser para Bob; ela também consegue capturar todos os pacotes enviados por Bob e enviar o que quiser para Alice. Cite algumas atitudes maliciosas que Trudy pode fazer a partir de sua posição.



Problemas

1. Projete e descreva um protocolo de nível de aplicação para ser usado entre um caixa automático e o computador central de um banco. Esse protocolo deve permitir verificação do cartão e da senha de um usuário, consulta do saldo de sua conta (que é mantido no computador central) e saque de dinheiro da conta corrente (isto é, entrega de dinheiro ao usuário). As entidades do protocolo devem estar preparadas a resolver o caso comum em que não há dinheiro suficiente na conta do usuário para cobrir o saque. Faça uma especificação de seu protocolo relacionando as mensagens trocadas e as ações realizadas pelo caixa automático ou pelo computador central do banco na transmissão e recepção de mensagens. Esquematize a operação de seu protocolo para o caso de um saque simples sem erros, usando um diagrama semelhante ao da Figura 1.2. Descreva explicitamente o que seu protocolo espera do serviço de transporte fim a fim.
2. Considere uma aplicação que transmita dados a uma taxa constante (por exemplo, a origem gera uma unidade de dados de N bits a cada k unidades de tempo, onde k é pequeno e fixo). Considere também que, quando essa aplicação começa, continuará em funcionamento por um período de tempo relativamente longo. Responda às seguintes perguntas, dando uma breve justificativa para suas respostas:
 - a. O que seria mais apropriado para essa aplicação: uma rede de comutação de circuitos ou uma rede de comutação de pacotes? Por quê?
 - b. Suponha que seja usada uma rede de comutação de pacotes e que o único tráfego dessa rede venha de aplicações como a descrita anteriormente. Além disso, admita que a soma das velocidades de dados da aplicação seja menor do que a capacidade de cada um dos enlaces. Será necessário algum tipo de controle de congestionamento? Por quê?
3. Considere a rede de comutação de circuitos da Figura 1.12. Lembre-se de que há n circuitos em cada enlace.
 - a. Qual é o número máximo de conexões simultâneas que podem estar em curso a qualquer instante nessa rede?
4. Considere novamente a analogia do comboio de carros da Seção 1.4. Admita uma velocidade de propagação de 100 km/h.
 - a. Suponha que o comboio viaje 150 km, começando em frente ao primeiro dos postos de pedágio, passando por um segundo e terminando após um terceiro. Qual é o atraso fim a fim?
 - b. Repita o item 'a' admitindo agora que haja sete carros no comboio em vez de dez.
5. Este problema elementar começa a explorar atrasos de propagação e de transmissão, dois conceitos centrais em redes de computadores. Considere dois computadores, A e B, conectados por um único enlace de taxa R bps. Suponha que esses computadores estejam separados por m metros e que a velocidade de propagação ao longo do enlace seja de s metros/segundo. O computador A tem de enviar um pacote de L bits ao computador B.
 - a. Expressse o atraso de propagação, d_{prop} , em termos de m e s .
 - b. Determine o tempo de transmissão do pacote, d_{trans} , em termos de L e R .
 - c. Ignorando os atrasos de processamento e de fila, obtenha uma expressão para o atraso fim a fim.
 - d. Suponha que o computador A comece a transmitir o pacote no instante $t = 0$. No instante $t = d_{trans}$, onde estará o último bit do pacote?
 - e. Suponha que d_{prop} seja maior do que d_{trans} . Onde estará o primeiro bit do pacote no instante $t = d_{trans}$?
 - f. Suponha que d_{prop} seja menor do que d_{trans} . Onde estará o primeiro bit do pacote no instante $t = d_{trans}$?
 - g. Suponha $s = 2,5 \cdot 10^8$, $L = 100$ bits e $R = 56$ kbps. Encontre a distância m de forma que d_{prop} seja igual a d_{trans} .
6. Neste problema, consideramos o envio de voz em tempo real do computador A para o computador

- B por meio de uma rede de comutação de pacotes (VoIP). O computador A converte voz analógica para uma cadeia digital de bits de 64 kbps e, em seguida, agrupa os bits em pacotes de 56 bytes. Há apenas um enlace entre os computadores A e B; sua taxa de transmissão é de 2 Mbps e seu atraso de propagação, de 10 milissegundos. Assim que o computador A recolhe um pacote, ele o envia ao computador B. Quando recebe um pacote completo, o computador B converte os bits do pacote em um sinal analógico. Quanto tempo decorre entre o momento em que um bit é criado (a partir do sinal analógico no computador A) e o momento em que ele é decodificado (como parte do sinal analógico no computador B)?
7. Suponha que usuários compartilhem um enlace de 3 Mbps e que cada usuário precise de 150 kbps para transmitir, mas que transmita apenas durante 10 por cento do tempo. (Veja a discussão sobre multiplexação estatística na Seção 1.3.)
- Quando é utilizada comutação de circuitos, quantos usuários podem ter suporte?
 - Suponha que haja 120 usuários. Determine a probabilidade que, a um tempo dado, exatamente n usuários estejam transmitindo simultaneamente. (Dica: Use a distribuição binomial.)
 - Determine a probabilidade de haver 21 ou mais usuários transmitindo simultaneamente.
8. Considere a discussão na Seção 1.3 sobre multiplexação estatística, na qual é dado um exemplo com um enlace de 1 Mbps. Quando em atividade, os usuários estão gerando dados a uma taxa de 100 kbps; mas a probabilidade de estarem em atividade, gerando dados, é de $p = 0,1$. Suponha que o enlace de 1 Mbps seja substituído por um enlace de 1 Gbps.
- Qual é o número máximo de usuários, N , que pode ser suportado simultaneamente por comutação de pacotes?
 - Agora considere comutação de circuitos e um número M de usuários. Elabore uma fórmula (em termos de p , M , N) para a probabilidade de que mais de N usuários estejam enviando dados.
9. Considere um pacote de comprimento L que se inicia no sistema final A e percorre três enlaces até um sistema final de destino. Esses três enlaces estão conectados por dois comutadores de pacotes. Suponha que d_i , s_i e R_i representem o comprimento, a velocidade de propagação e a taxa de transmissão do enlace i , sendo $i = 1, 2, 3$. O comutador de pacote atrasa cada pacote por d_{proc} . Considerando que não haja nenhum atraso de fila, em relação a d_i , s_i e R_i , ($i = 1, 2, 3$) e L , qual é o atraso fim a fim total para o pacote? Suponha agora que o pacote tenha 1.500 bytes, a velocidade de propagação de ambos enlaces seja $2,5 \cdot 10^8$ m/s, as taxas de transmissão dos três enlaces sejam 2 Mbps, o atraso de processamento do comutador de pacote seja de 3 milissegundos, o comprimento do primeiro enlace seja 5.000 km, o comprimento do segundo seja 4.000 km e do último seja 1.000 km. Dados esses valores, qual é o atraso fim a fim?
10. No problema anterior, suponha que $R_1 = R_2 = R_3 = R$ e $d_{\text{proc}} = 0$. Suponha que o comutador de pacote não armazena e envia pacotes, mas transmite imediatamente cada bit recebido antes de esperar o pacote chegar. Qual é o atraso fim a fim?
11. Um comutador de pacote recebe um pacote e determina o enlace de saída pelo qual o pacote deve ser enviado. Quando o pacote chega, um outro já está sendo transmitido nesse enlace de saída e outros quatro já estão esperando para serem transmitidos. Os pacotes são transmitidos em ordem de chegada. Suponha que todos os pacotes tenham 1.500 bytes e que a taxa do enlace seja 2 Mbps. Qual é o atraso de fila para o pacote? Em um amplo sentido, qual é o atraso de fila quando todos os pacotes possuem comprimento L , o enlace possui uma taxa de transmissão $R \times$ bits do pacote sendo enviado já foram transmitidos e N pacotes já estão na fila?
12. Suponha que N pacotes cheguem simultaneamente ao enlace no qual não há pacotes sendo transmitidos e nem pacotes enfileirados. Cada pacote tem L de comprimento e é transmitido à taxa R . Qual é o atraso médio para os N pacotes?
13. Considere o atraso de fila em um buffer de roteador (antes de um enlace de saída). Suponha que todos os pacotes tenham L bits, que a taxa de transmissão seja de R bps e que N pacotes cheguem simultaneamente ao buffer a cada LN/R segundos. Determine o atraso de fila médio para um pacote. (Dica: o atraso de fila para o primeiro pacote é zero; para o segundo pacote, L/R ; para o terceiro pacote, $2L/R$. O pacote de ordem N já terá sido transmitido quando o segundo lote de pacotes chegar.)
14. Considere o atraso de fila em um buffer de roteador, sendo I a intensidade de tráfego; isto é, $I = La/R$. Suponha que o atraso de fila tome a forma de $IL/R(1 - I)$ para $I < 1$.
- Deduza uma fórmula para o atraso total, isto é, para o atraso de fila mais o atraso de transmissão.
 - Faça um gráfico do atraso total como uma função de L/R .
15. Sendo a a taxa de pacotes que chegam a um enlace em pacotes/s, e μ a taxa de transmissão de enlaces em pacotes/s, baseado na fórmula do atraso total (isto é, o atraso de fila mais o atraso de transmissão) do problema anterior, deduza uma fórmula para o atraso total em relação a a e μ .

- 16.** Considere um buffer de roteador antes de um enlace de saída. Neste problema, você usará a fórmula de Little, uma famosa fórmula da teoria das filas. Sendo N o número médio de pacotes no buffer mais o pacote sendo transmitido, a a taxa de pacotes que chegam no enlace, e d o atraso total médio (isto é, o atraso de fila mais o atraso de transmissão) sofrido pelo pacote. Dada a fórmula de Little $N = a \cdot d$, suponha que, na média, o buffer contenha 10 pacotes, o atraso de fila de pacote médio seja 10 milissegundos e a taxa de transmissão do enlace seja 100 pacotes/s. Utilizando tal fórmula, qual é a taxa média de chegada de pacote, considerando que não há perda de pacote?
- 17.** **a.** Generalize a fórmula para o atraso fim a fim na Seção 1.4.3 para taxas de processamento, atrasos de propagação e taxa de transmissão heterogêneos.
b. Repita o item “a”, mas suponha também que haja um atraso de fila médio d_{fila} em cada nó.
- 18.** Execute o programa Traceroute para verificar a rota entre uma origem e um destino, no mesmo continente, para três horários diferentes do dia.
- Determine a média e o desvio padrão dos atrasos de ida e volta para cada um dos três horários.
 - Determine o número de roteadores no caminho para cada um dos três. Os caminhos mudaram em algum dos horários?
 - Tente identificar o número de redes ISPs pelas quais o pacote do Traceroute passa entre origem e destino. Roteadores com nomes semelhantes e/ou endereços IP semelhantes devem ser considerados como parte do mesmo ISP. Em suas respostas, os maiores atrasos ocorrem nas interfaces de formação de pares entre ISPs adjacentes?
 - Faça o mesmo para uma origem e um destino em continentes diferentes. Compare os resultados dentro do mesmo continente com os resultados entre continentes diferentes.
- 19.** Considere o exemplo de vazão correspondente à Figura 1.20 (b). Agora imagine que haja M pares de cliente-servidor em vez de 10. R_s , R_c e R representam as taxas do enlace do servidor, enlaces do cliente e enlace da rede. Suponha que os outros enlaces possuam capacidade abundante e que não haja outro tráfego na rede além daquele gerado pelos M pares de cliente-servidor. Deduza uma expressão geral para a vazão em relação a R_s , R_c , R e M .
- 20.** Considere a Figura 1.19 (b). Agora suponha que haja M percursos entre o servidor e o cliente. Nenhum dos dois percursos compartilham qualquer enlace. O percurso k ($k = 1, \dots, M$) consiste em N enlaces com taxas de transmissão $R_1^k, R_2^k, \dots, R_N^k$. Se o servidor pode usar somente um percurso para enviar dados ao cliente, qual é a vazão máxima que ele pode atingir? Se o servidor pode usar todos os M percursos para enviar dados, qual é a vazão máxima que ele pode atingir?
- 21.** Considere a Figura 1.19 (b). Suponha que cada enlace entre o servidor e o cliente possua uma probabilidade de perda de pacote p , e que as probabilidades de perda de pacote para esses enlaces sejam independentes. Qual é a probabilidade de um pacote (enviado pelo servidor) ser recebido com sucesso pelo receptor? Se o pacote se perder no percurso do servidor para o cliente, então o servidor retransmitirá o pacote. Na média, quantas vezes o servidor retransmitirá o pacote para que o cliente o receba com sucesso?
- 22.** Considere a Figura 1.19 (a). Suponha que o enlace de gargalo ao longo do percurso do servidor para o cliente seja o primeiro com a taxa R_s bps. Imagine que enviamos um par de pacotes um após o outro do servidor para o cliente, e que não haja nenhum outro tráfego nesse percurso. Imagine também que cada pacote de tamanho L bits e os dois enlaces tenham o mesmo atraso de propagação d_{prop} .
- Qual é o tempo entre chegadas ao destino? Isto é, quanto tempo transcorre de quando o último bit do primeiro pacote chega até o último bit do segundo pacote chegar?
 - Agora suponha que o segundo enlace seja o de gargalo (isto é, $R_c < R_s$). É possível que o segundo pacote entre na fila de entrada do segundo enlace? Explique. Agora imagine que o servidor envie o segundo pacote T segundos após enviar o primeiro. Quão grande deve ser T para garantir que não haja nenhuma fila antes do segundo enlace? Explique.
- 23.** Suponha que você queira enviar, urgentemente, 40 terabytes de dados de Boston para Los Angeles. Você tem disponível um enlace dedicado de 100 Mbps para transferência de dados. Você escolheria transmitir os dados por meio desse enlace ou usar o serviço de entrega 24 horas FedEx? Explique.
- 24.** Suponha que dois computadores, A e B, estejam separados por uma distância de 20 mil quilômetros e conectados por um enlace direto de $R = 2$ Mbps. Suponha que a velocidade de propagação pelo enlace seja de $2,5 \cdot 10^8$ metros por segundo.
- Calcule o produto largura de banda-atraso $R \cdot d_{prop}$.
 - Considere o envio de um arquivo de 800 mil bits do computador A para o computador B. Suponha que o arquivo seja enviado continuamente, como se fosse uma única grande mensagem. Qual é o número máximo de bits que estará no enlace a qualquer dado instante?

- c. Interprete o produto largura de banda-atraso.
- d. Qual é o comprimento (em metros) de um bit no enlace? É maior do que a de um campo de futebol?
- e. Derive uma expressão geral para o comprimento de um bit em termos da velocidade de propagação s , da velocidade de transmissão R e do comprimento do enlace m .
25. Com referência ao problema 24, suponha que possamos modificar R . Para qual valor de R o comprimento de um bit será o mesmo que o comprimento do enlace?
26. Considere o problema 24, mas agora com um enlace de $R = 1$ Gbps.
- Calcule o produto largura de banda-atraso, $R \cdot d_{\text{prop}}$.
 - Considere o envio de um arquivo de 800 mil bits do computador A para o computador B. Suponha que o arquivo seja enviado continuamente, como se fosse uma única grande mensagem. Qual será o número máximo de bits que estará no enlace a qualquer dado instante?
 - Qual é o comprimento (em metros) de um bit no enlace?
27. Novamente com referência ao problema 24.
- Quanto tempo demora para enviar o arquivo, admitindo que ele seja enviado continuamente?
 - Suponha agora que o arquivo seja fragmentado em 20 pacotes e que cada pacote contenha 40 mil bits. Suponha que cada pacote seja verificado pelo receptor e que o tempo de transmissão de uma verificação de pacote seja desprezível. Finalmente, admita que o emissor não possa enviar um pacote até que o anterior tenha sido reconhecido. Quanto tempo demorará para enviar o arquivo?
 - Compare os resultados de 'a' e 'b'.
- Suponha que haja um enlace de microondas de 10 Mbps entre um satélite geoestacionário e sua estação-base na Terra. A cada minuto o satélite tira uma foto digital e a envia à estação-base. Admita uma velocidade de propagação de $2,4 \cdot 10^8$ metros por segundo.
- Qual é o atraso de propagação do enlace?
 - Qual é o produto largura de banda-atraso, $R \cdot d_{\text{prop}}$?
 - Seja x o tamanho da foto. Qual é o valor mínimo de x para que o enlace de micro-ondas transmita continuamente?
28. Considere a analogia da viagem aérea que utilizamos em nossa discussão sobre camadas na Seção 1.7, e a adição de cabeçalhos a unidades de dados de protocolo enquanto passam por sua pilha. Existe uma noção equivalente de adição de informações de cabeçalho à movimentação de passageiros e suas malas pela pilha de protocolos da linha aérea?
29. Em redes modernas de comutação de pacotes, a máquina de origem segmenta mensagens longas de camada de aplicação (por exemplo, uma imagem ou um arquivo de música) em pacotes menores e os envia pela rede. A máquina destinatária, então, monta novamente os pacotes restaurando a mensagem original. Denominamos esse processo *segmentação de mensagem*. A Figura 1.28 ilustra o transporte fim a fim de uma mensagem com e sem segmentação. Considere que uma mensagem de $8 \cdot 10^6$ bits de comprimento tenha de ser enviada da origem ao destino na Figura 1.28. Suponha que a velocidade de cada enlace da figura seja 2 Mbps. Ignore atrasos de propagação, de fila e de processamento.
- Considere o envio da mensagem da origem ao destino *sem segmentação*. Quanto tempo essa mensagem levará para ir da máquina de origem até o primeiro comutador de pacotes? Tendo em mente que cada comutador usa comutação de pacotes do tipo armazena-e-reenvia, qual é o tempo total para levar a mensagem da máquina de origem à máquina de destino?
 - Agora suponha que a mensagem seja segmentada em 4 mil pacotes, cada um com 2.000 bits de comprimento. Quanto tempo demorará para o primeiro pacote ir da máquina de origem até o primeiro comutador? Quando o primeiro pacote está sendo enviado do primeiro ao segundo comutador, o segundo pacote está sendo enviado da máquina de origem ao primeiro comutador. Em que instante o segundo pacote terá sido completamente recebido no primeiro comutador?
 - Quanto tempo demorará para movimentar o arquivo da máquina de origem até a máquina de destino quando é usada segmentação de mensagem? Compare este resultado com sua resposta na parte 'a' e comente.
 - Discuta as desvantagens da segmentação de mensagem.
30. Experimente o applet "Message Segmentation apresentado" no site Web deste livro. Os atrasos no applet correspondem aos atrasos obtidos no problema anterior? Como os atrasos de propagação no enlace afetam o atraso total fim a fim na comutação de pacotes (com segmentação de mensagem) e na comutação de mensagens?

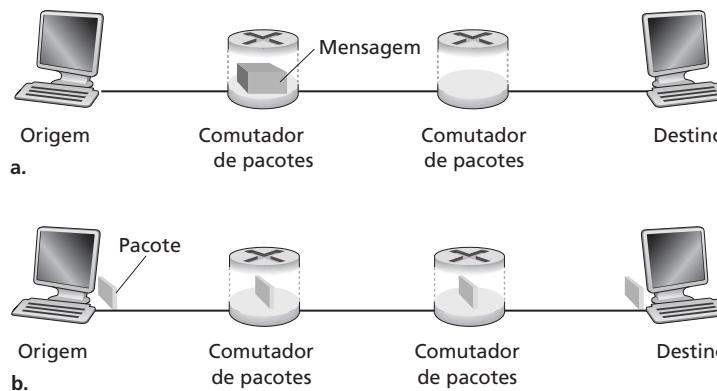


Figura 1.28 Transporte fin a fim de mensagem: a) sem segmentação de mensagem; (b) com segmentação de mensagem

31. Considere o envio de um arquivo grande de F bits do computador A para o computador B. Há dois enlaces (e um comutador) entre eles e os enlaces não estão congestionados (isto é, não há atrasos de fila). O computador A fragmenta o arquivo em segmentos de S bits
- cada e adiciona 80 bits de cabeçalho a cada segmento, formando pacotes de $L = 40 + S$ bits. Cada enlace tem uma taxa de transmissão de R bps. Qual é o valor de S que minimiza o atraso para levar o arquivo de A para B? Desconsidere o atraso de propagação.



Questões dissertativas

- Que tipos de serviços de telefone celular sem fio estão disponíveis em sua área?
- Usando tecnologia de LAN sem fio 802.11, elabore o projeto de uma rede doméstica para sua casa ou para a casa de seus pais. Relacione os modelos de produtos específicos para essa rede doméstica juntamente com seus custos.
- Descreva os serviços Skype PC para PC. Experimente o serviço de vídeo do Skype PC para PC e relate a experiência.
- O Skype oferece um serviço que permite realizar chamadas telefônicas de um computador para um computador comum. Isso significa que a chamada deve passar pela Internet e pela rede telefônica. Discuta como esse processo deve ser feito.
- O que é Short Message Service? Em quais países/continentes esse serviço é comum? É possível enviar uma mensagem SMS de um site Web para um telefone portátil?
- O que é recepção de vídeo armazenado? Quais são os sites Web mais populares que oferecem esse serviço hoje?
- O que é recepção de vídeo em tempo real por P2P? Quais são os sites Web mais populares que oferecem esse serviço hoje?
- Descubra cinco empresas que oferecem serviços de compartilhamento de arquivos P2P. Cite os tipos de arquivos (isto é, conteúdo) que cada empresa processa.
- Quem inventou o ICQ, o primeiro serviço de mensagem instantânea? Quando foi inventado e que idade tinham seus inventores? Quem inventou o Napster? Quando foi inventado e que idade tinham seus inventores?
- Quais são as semelhanças e diferenças entre as tecnologias Wi-Fi e 3G de acesso sem fio à Internet? Quais são as taxas de bits dos dois serviços? Quais são os custos? Discuta roaming e acesso de qualquer lugar.
- Por que o serviço original de compartilhamento de arquivo P2P Napster deixou de existir? O que é a RIAA e que providências está tomando para limitar o compartilhamento de arquivos P2P de conteúdo protegido por direitos autorais? Qual é a diferença entre violação de direitos autorais direta e indireta?
- O que é BitTorrent? No que ele difere de um serviço de compartilhamento de arquivo P2P, como eDonkey, LimeWire ou Kazaa?
- Você acha que daqui a dez anos redes de computadores ainda compartilharão amplamente arquivos protegidos por direitos autorais? Por que sim ou por que não? Justifique.



Wireshark Lab

*“Conte-me e eu esquecerei. Mostre-me e eu lembrarei.
Envolva-me e eu entenderei.”*

Provérbio chinês

A compreensão de protocolos de rede pode ser muito mais profunda se os virmos em ação e interagirmos com eles — observando a sequência de mensagens trocadas entre duas entidades de protocolo, pesquisando detalhes de sua operação, fazendo com que eles executem determinadas ações e observando essas ações e suas consequências. Isso pode ser feito em cenários simulados ou em um ambiente real de rede, tal como a Internet. Os applets Java apresentados (em inglês) no site deste livro adotam a primeira abordagem. Nos Wireshark labs adotaremos a última. Você executará aplicações de rede em vários cenários utilizando seu computador no escritório, em casa ou em um laboratório e observará também os protocolos de rede interagindo e trocando mensagens com entidades de protocolo que estão executando em outros lugares da Internet. Assim, você e seu computador serão partes integrantes desses laboratórios ao vivo e você observará — e aprenderá — fazendo.

A ferramenta básica para observar as mensagens trocadas entre entidades de protocolos em execução é denominada analisador de pacotes. Como o nome sugere, um analisador de pacotes recebe passivamente mensagens enviadas e recebidas por seu computador; também exibe o conteúdo dos vários campos de protocolo das mensagens que captura. Uma tela do analisador de pacotes Wireshark é mostrada na Figura 1.29. O Wireshark é um analisador de pacotes gratuito que funciona em computadores com sistemas operacionais Windows, Linux/Unix e Mac. Por todo o livro, você encontrará Wireshark labs que o habilitarão a explorar vários dos protocolos estudados em cada capítulo. Neste primeiro Wireshark lab, você obterá e instalará uma cópia do programa, acessará um site Web e examinará as mensagens de protocolo trocadas entre seu browser e o servidor Web.

Você encontrará detalhes completos, em inglês, sobre este primeiro Wireshark Lab (incluindo instruções sobre como obter e instalar o programa) no site www.aw.com/kurose_br.

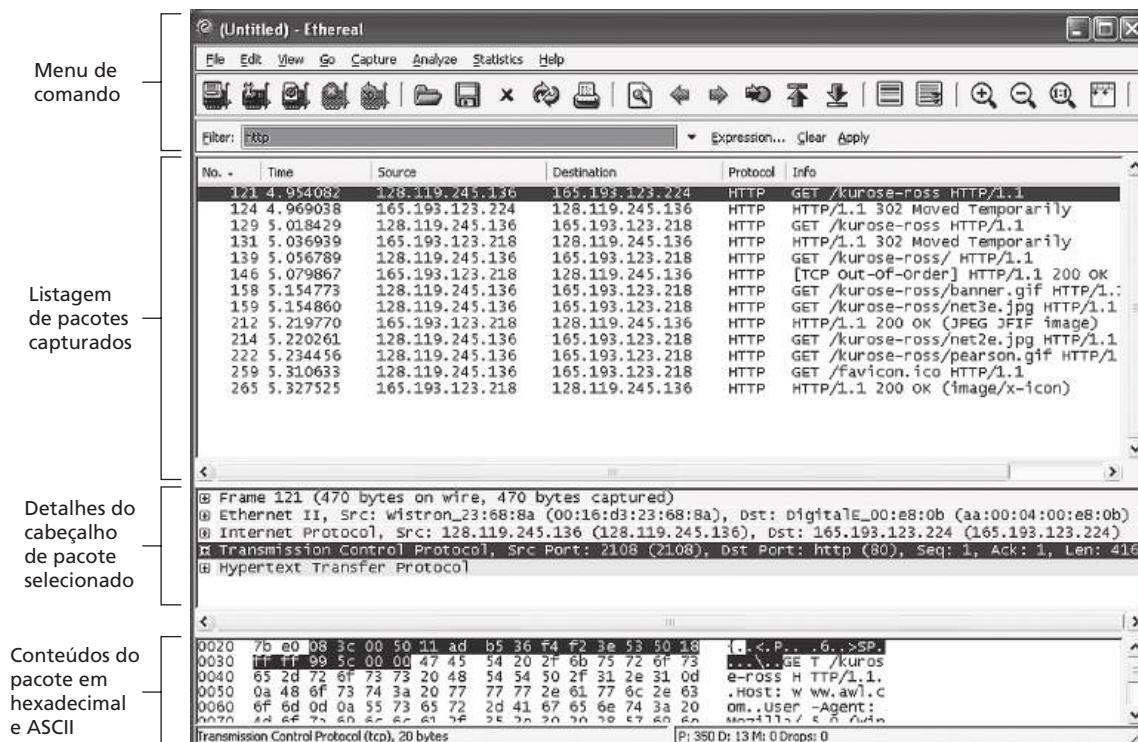


Figura 1.29 Uma amostra de tela do programa Wireshark

Entrevista

Leonard Kleinrock

Leonard Kleinrock é professor de ciência da computação da Universidade da Califórnia em Los Angeles. Em 1969, seu computador na UCLA se tornou o primeiro nó da Internet. Ele criou os princípios da comutação de pacotes em 1961, que se tornou a tecnologia básica da Internet. Leonard também é presidente e fundador da Nomadix, Inc., uma companhia cuja tecnologia oferece maior acessibilidade a serviços de Internet banda larga. Ele é bacharel em engenharia elétrica pela City College of New York (CCNY) e mestre e doutor em engenharia elétrica pelo Instituto de Tecnologia de Massachusetts (MIT).



O que o fez se decidir pela especialização em tecnologia de redes/Internet?

Como doutorando do MIT em 1959, percebi que a maioria dos meus colegas de turma estava fazendo pesquisas nas áreas de teoria da informação e de teoria da codificação. Havia no MIT o grande pesquisador Claude Shannon, que já tinha proposto estudos nessas áreas e resolvido a maior parte dos problemas importantes. Os problemas que restaram para pesquisar eram difíceis e de menor importância. Portanto, decidi propor uma nova área na qual até então ninguém tinha pensado. Lembre-se de que no MIT, eu estava cercado de computadores, e era evidente para mim que brevemente aquelas máquinas teriam de se comunicar umas com as outras. Na época, não havia nenhum meio eficaz de fazer isso; portanto, decidi desenvolver a tecnologia que permitiria a criação de redes de dados eficientes.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Frequentei o curso noturno de graduação em engenharia elétrica da CCNY de 1951 a 1957. Durante o dia, trabalhei inicialmente como técnico e depois como engenheiro em uma pequena empresa de eletrônica industrial chamada Photobell. Enquanto trabalhava lá, introduzi tecnologia digital na linha de produtos da empresa. Essencialmente, estávamos usando equipamentos fotoelétricos para detectar a presença de certos itens (caixas, pessoas etc.) e a utilização de um circuito conhecido na época como *multivibrator biestável* era exatamente o tipo de tecnologia de que precisávamos para levar o processamento digital a esse campo da detecção. Acontece que esses circuitos são os blocos de construção básicos dos computadores e vieram a ser conhecidos como *flip-flops* ou *comutadores* na linguagem coloquial de hoje.

O que passou por sua cabeça quando enviou a primeira mensagem computador a computador (da UCLA para o Stanford Research Institute)?

Francamente, eu não fazia a menor ideia da importância daquele acontecimento. Não havíamos preparado uma mensagem de significância histórica, como muitos criadores do passado o fizeram (Samuel Morse com “Que obra fez Deus.”, ou Alexandre Graham Bell, com “Watson, venha cá! Preciso de você.”, ou Neal Armstrong com “Este é um pequeno passo para o homem, mas um grande salto para a humanidade.”) Esses caras eram *inteligentes!* Eles entendiam de meios de comunicação e relações públicas. Nosso objetivo foi nos conectar ao computador do SRI. Então digitamos a letra “L”, que foi aceita corretamente, digitamos a letra “o”, que foi aceita, e depois digitamos a letra “g”, que fez o computador hospedeiro pifar! Então, nossa mensagem acabou sendo curta e, talvez, a mais profética de todas, ou seja, “Lo！”, como em “*Lo and behold*” (“*Pasmem!*”).

Anteriormente, naquele mesmo ano, fui citado em um comunicado de imprensa da UCLA por ter dito que, logo que a rede estivesse pronta e em funcionamento, seria possível ter acesso a utilidades de outros computadores a partir de nossa casa e escritório tão facilmente quanto tínhamos acesso à eletricidade e ao telefone. Portanto,

a visão que eu tinha da Internet naquela época era que ela seria onipresente, estaria sempre em funcionamento e sempre disponível, que qualquer pessoa que possuísse qualquer equipamento poderia se conectar com ela de qualquer lugar e que ela seria invisível. Mas jamais imaginei que minha mãe, aos 99 anos de idade, usaria a Internet — como ela realmente usou!

Em sua opinião, qual é o futuro das redes?

O fácil é predizer a infraestrutura por si mesma. Eu antecipo que vemos uma implantação considerável de computação nômade, aparelhos móveis e espaços inteligentes. Realmente, a disponibilidade de computação portátil, de alto desempenho, acessível e leve e de aparelho de comunicação (mais a onipresença da Internet) nos permitiu tornar nômades.

A computação nômade refere-se à tecnologia que permite aos usuários finais, que viajam de um lugar para o outro, ganhar acesso aos serviços da Internet de maneira transparente, não importando para onde vão e qual aparelho possuem ou ganham acesso. O difícil é predizer as aplicações e serviços, que nos surpreenderam consistentemente de formas dramáticas (e-mail, tecnologias de busca, a rede de alcance mundial, blogs, redes sociais, geração de usuários e compartilhamento de música, fotos, vídeos etc.). Estamos na margem de uma nova categoria de aplicações móveis, inovadoras e surpreendentes presentes em nossos aparelhos portáteis.

O passo seguinte vai nos capacitar a sair do mundo misterioso do ciberespaço para o mundo físico dos espaços inteligentes. A tecnologia dará vida a nossos ambientes (mesas, paredes, veículos, relógios e cintos, entre outros) por meio de atuadores, sensores, lógica, processamento, armazenagem, câmeras, microfones, alto-falantes, painéis e comunicação. Essa tecnologia embutida permitirá que nosso ambiente forneça os serviços IP que quisermos. Quando eu entrar em uma sala, ela saberá que entrei. Poderei me comunicar com meu ambiente naturalmente, como se estivesse falando o meu idioma nativo; minhas solicitações gerarão respostas apresentadas como páginas Web em painéis de parede, por meus óculos, por voz, por hologramas e assim por diante.

Analizando um panorama mais longínquo, vejo um futuro para as redes que inclui componentes fundamentais que ainda virão. Vejo agentes inteligentes de software distribuídos por toda a rede cuja função é fazer mineração de dados, agir sobre esses dados, observar tendências e adaptar e realizar tarefas dinamicamente. Vejo tráfego de rede consideravelmente maior gerado não tanto por seres humanos, mas por esses equipamentos embutidos e agentes inteligentes de software. Vejo grandes conjuntos de sistemas auto-organizáveis controlando essa rede imensa e veloz. Vejo quantidades enormes de informações zunindo por essa rede instantaneamente e passando por extraordinários processamentos e filtragens. A Internet será, essencialmente, um sistema nervoso de presença global. Vejo tudo isso e mais enquanto entramos de cabeça no século XXI.

Que pessoas o inspiraram profissionalmente?

Quem mais me inspirou foi Claude Shannon, do MIT, um brilhante pesquisador que tinha a capacidade de relacionar suas ideias matemáticas com o mundo físico de modo muitíssimo intuitivo. Ele fazia parte da banca examinadora de minha tese de doutorado.

Você pode dar algum conselho aos estudantes que estão ingressando na área de redes/Internet?

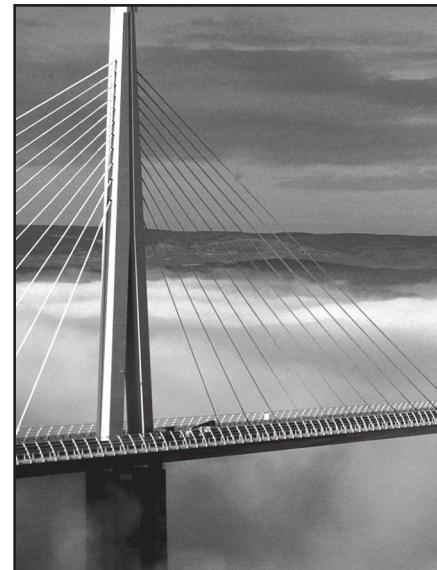
A Internet, e tudo o que ela habilita, é uma vasta fronteira nova, cheia de desafios surpreendentes. Há espaço para grandes inovações. Não fiquem limitados à tecnologia existente hoje. Soltem sua imaginação e pensem no que poderia acontecer e transformem isso em realidade.





Capítulo 2

Camada de aplicação



Aplicações são a razão de ser de uma rede de computadores. Se não fosse possível inventar aplicações úteis, não haveria necessidade de projetar protocolos de rede para suportá-las. Nos últimos 40 anos, foram criadas numerosas aplicações de rede engenhosas e maravilhosas. Entre elas estão as aplicações clássicas de texto, que se tornaram populares na década de 1970 e 1980: correio eletrônico, acesso a computadores remotos, transferência de arquivo, grupos de discussão e bate-papo e também uma aplicação que alcançou estrondoso sucesso em meados da década de 1990: a *World Wide Web*, abrangendo a navegação na Web, busca e o comércio eletrônico. Duas aplicações de enorme sucesso também surgiram no final do milênio — mensagem instantânea com lista de amigos e compartilhamento P2P de arquivos, assim como muitas aplicações de áudio e vídeo, incluindo a telefonia por Internet, transmissão e compartilhamento de vídeo, rádio via Internet e televisão sobre o protocolo IP (IPTV). Além disso, a penetração crescente de acesso residencial banda larga e a onipresença de acesso sem fio estão preparando o terreno para aplicações mais modernas e interessantes no futuro.

Neste capítulo estudamos os aspectos conceituais e de implementação de aplicações de rede. Começamos definindo conceitos fundamentais de camada de aplicação, incluindo serviços de rede exigidos por aplicações, clientes e servidores, processos e interfaces de camada de transporte. Examinamos detalhadamente várias aplicações de rede, entre elas a Web, e-mail, DNS, compartilhamento de arquivos P2P e telefonia por Internet P2P.

Em seguida, abordamos desenvolvimento de aplicação de rede por TCP e também por UDP. Em particular, estudamos o API socket e examinamos algumas aplicações cliente-servidor simples em Java. Apresentamos também vários exercícios divertidos e interessantes de programação de aplicações no final do capítulo.

A camada de aplicação é um lugar particularmente bom para iniciarmos o estudo de protocolos. É terreno familiar, pois conhecemos muitas das aplicações que dependem dos protocolos que estudaremos. Ela nos dará uma boa ideia do que são protocolos e nos apresentará muitos assuntos que encontraremos novamente quando estudarmos protocolos de camadas de transporte, de rede e de enlace.

2.1 Princípios de aplicações de rede

Suponha que você tenha uma grande ideia para uma nova aplicação de rede. Essa aplicação será, talvez, um grande serviço para a humanidade, ou agradará a seu professor, ou fará de você um homem (ou mulher) rico(a), ou simplesmente será divertido desenvolvê-la. Seja qual for sua motivação, vamos examinar agora como transformar a ideia em uma aplicação do mundo real.

O cerne do desenvolvimento de aplicação de rede é escrever programas que rodem em sistemas finais diferentes e se comuniquem entre si pela rede. Por exemplo, na aplicação Web há dois programas distintos que se comunicam um com o outro: o programa do browser, que roda na máquina do usuário (computador de mesa, laptop, PDA, telefone celular e assim por diante); e o programa do servidor Web, que roda na máquina do servidor Web. Outro exemplo é um sistema de compartilhamento de arquivos P2P no qual há um programa em cada máquina que participa da comunidade de compartilhamento de arquivos. Nesse caso, os programas de cada máquina podem ser semelhantes ou idênticos.

Portanto, ao desenvolver sua nova aplicação, você precisará escrever um software que rode em vários sistemas finais. Esse software poderia ser criado, por exemplo, em C, Java ou Python. Importante: você não precisará escrever programas que executem nos elementos do núcleo de rede, como roteadores e switches. Mesmo se quisesse, você não poderia escrever programas para esses elementos. Como aprendemos no Capítulo 1 e mostramos na Figura 1.24, equipamentos de núcleo de rede não funcionam na camada de aplicação, mas em camadas mais baixas, especificamente na de rede e abaixo dela. Esse projeto básico — a saber, confinar o software de aplicação nos sistemas finais, como mostra a Figura 2.1, facilitou o desenvolvimento e a proliferação rápidos de uma vasta gama de aplicações de Internet.

2.1.1 Arquiteturas de aplicação de rede

Antes de mergulhar na codificação do software, você deverá elaborar um plano geral para a arquitetura da sua aplicação. Tenha sempre em mente que a arquitetura de uma aplicação é distintamente diferente da arquitetura de rede (por exemplo, a arquitetura em cinco camadas da Internet que discutimos no Capítulo 1). Do ponto de vista do profissional que desenvolve a aplicação, a arquitetura de rede é fixa e provê um conjunto específico de serviços às aplicações. Por outro lado, a arquitetura da aplicação é projetada pelo desenvolvedor e determina como a aplicação é organizada nos vários sistemas finais. Ao escolher a arquitetura da aplicação, o desenvolvedor provavelmente aproveitará uma das duas arquiteturas mais utilizadas em aplicações modernas de rede: a arquitetura cliente-servidor ou a arquitetura P2P.

Em uma arquitetura cliente-servidor há um hospedeiro sempre em funcionamento, denominado *servidor*, que atende a requisições de muitos outros hospedeiros, denominados *clientes*. Estes podem estar em funcionamento às vezes ou sempre. Um exemplo clássico é a aplicação Web na qual um servidor Web que está sempre em funcionamento atende a requisições de browsers de hospedeiros clientes. Quando recebe uma requisição de um objeto de um hospedeiro cliente, um servidor Web responde enviando o objeto requisitado a ele. Observe que, na arquitetura cliente-servidor, os clientes não se comunicam diretamente uns com os outros; por exemplo, na aplicação Web, dois browsers não se comunicam diretamente. Outra característica da arquitetura cliente-servidor é que o servidor tem um endereço fixo, bem conhecido, denominado endereço IP (que discutiremos em breve). Devido a essa característica do servidor e devido ao fato de ele estar sempre em funcionamento, um cliente sempre pode contatá-lo, enviando um pacote ao endereço do servidor. Algumas das aplicações mais conhecidas que empregam a arquitetura cliente-servidor são Web, FTP, Telnet e e-mail. Essa arquitetura cliente-servidor é mostrada na Figura 2.2(a).

Em aplicações cliente-servidor, muitas vezes acontece de um único hospedeiro servidor ser incapaz de atender a todas as requisições de seus clientes. Por exemplo, um site Web popular pode ficar rapidamente saturado se tiver apenas um servidor para atender a todas as requisições. Por essa razão, um grande conjunto de hospedeiros — às vezes coletivamente chamado **data center** — frequentemente é usado para criar um servidor virtual poderoso em arquiteturas cliente-servidor. Os serviços de aplicação baseados na arquitetura cliente-servidor são geralmente de infraestrutura intensiva, uma vez que requerem que os provedores de serviço comprem, instalem e preservem o *server farm*. Além disso, os provedores de serviço devem pagar as despesas de interconexão recorrente e largura de banda para enviar e receber dados para e da Internet. Serviços populares como mecanismos de busca (por exemplo, o Google), comércio via Internet (por exemplo, Amazon e e-Bay), e-mail baseado na Web (por exemplo, Yahoo Mail), rede social (por exemplo, MySpace e Facebook) e compartilhamento de vídeo (por exemplo, YouTube) exigem muita infraestrutura e são de fornecimento dispendioso.

Em uma **arquitetura P2P**, há uma confiança mínima (ou nenhuma) nos servidores sempre em funcionamento. Em vez disso, a aplicação utiliza a comunicação direta entre pares de hospedeiros conectados

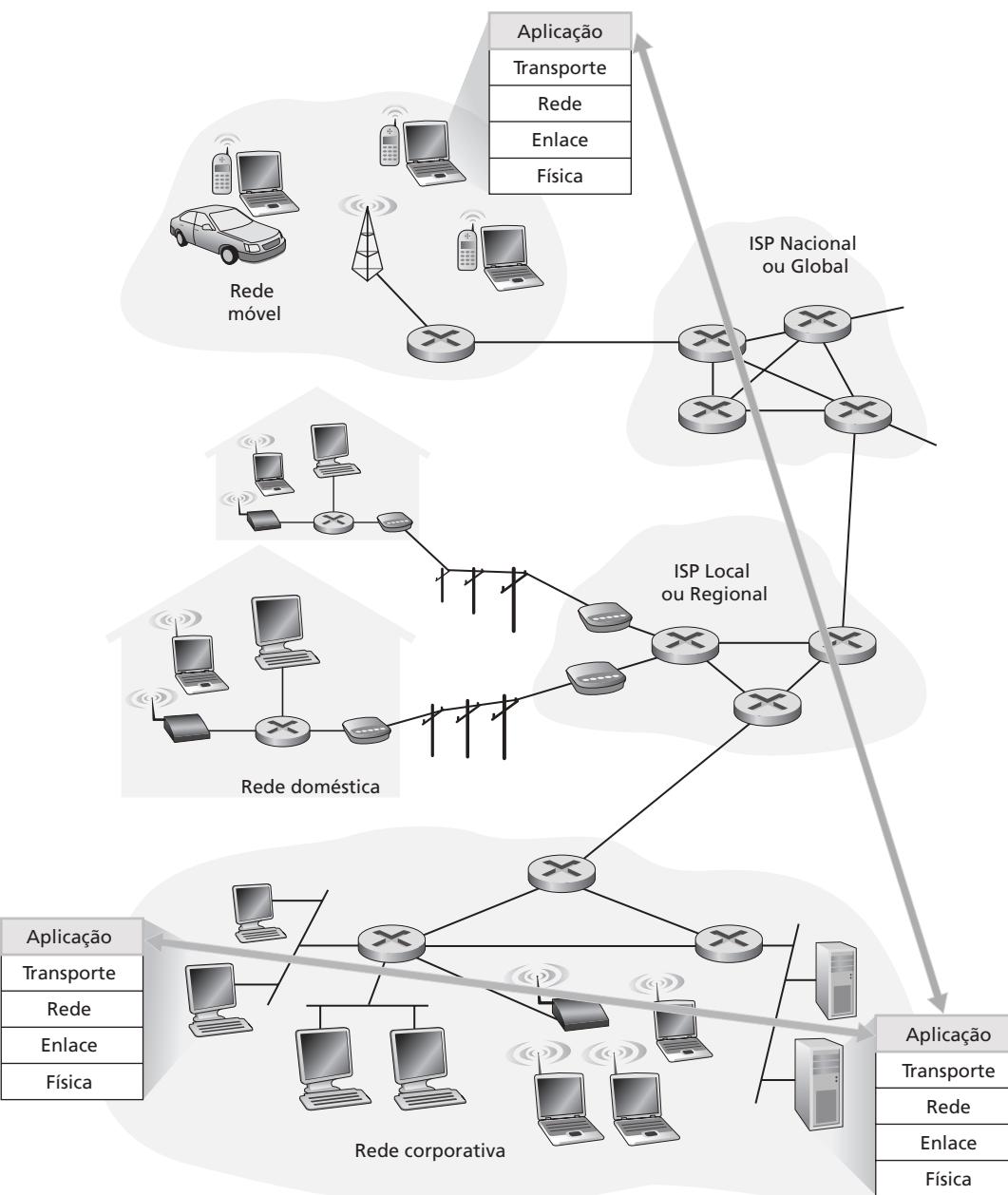


Figura 2.1 A comunicação de uma aplicação de rede ocorre entre sistemas finais na camada de aplicação

alternadamente, denominados *pares*. Os *pares* não são de propriedade dos provedores de serviço, mas são controlados por usuários de computadores de mesa e laptops, sendo que a maioria dos pares se aloja em residências, universidades e escritórios. Como os pares se comunicam sem passar por nenhum servidor dedicado, a arquitetura é denominada par-a-par (*par-to-par* — P2P). Muitas das aplicações de hoje mais populares e de intenso tráfego são baseadas nas arquiteturas P2P, incluindo distribuição de arquivos (por exemplo, BitTorrent), compartilhamento de arquivo (por exemplo, eMule e LimeWire), telefonia por Internet (por exemplo, Skype) e IPTV (por exemplo, PPLive). Essa arquitetura está ilustrada na Figura 2.2 (b). Mencionamos que algumas aplicações possuem arquiteturas híbridas, combinando elementos cliente-servidor e P2P. Por exemplo, para muitas aplicações de mensagem instantânea, os servidores costumam

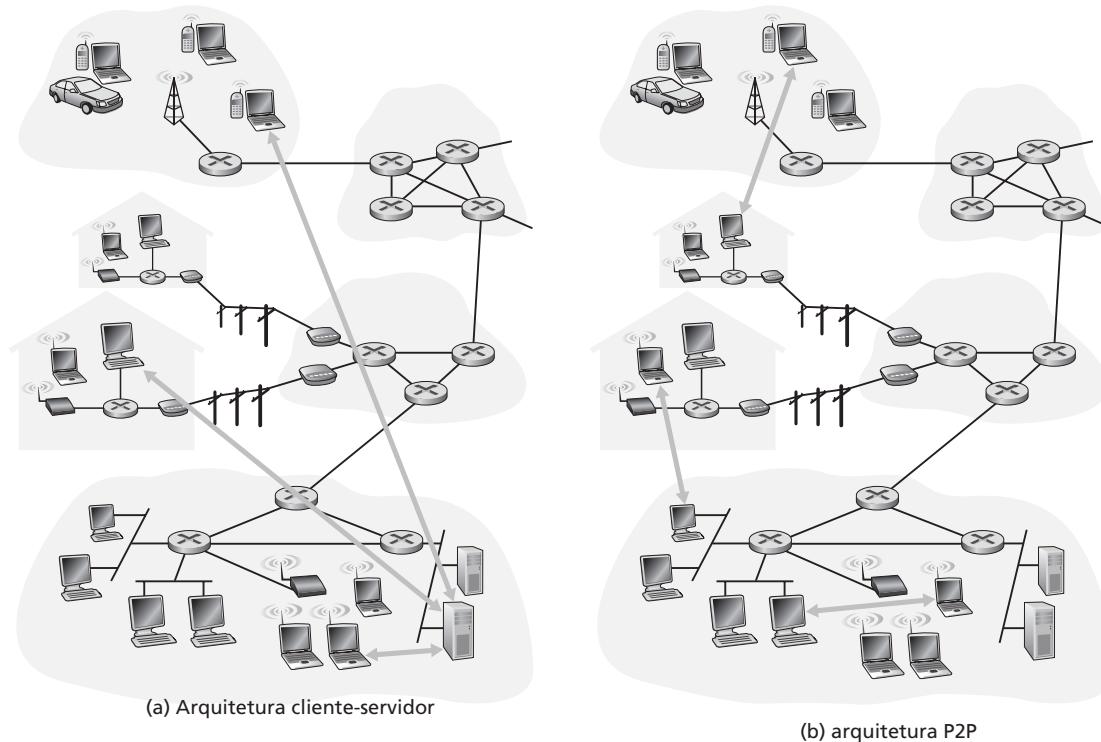


Figura 2.2 (a) Arquitetura cliente-servidor; (b) arquitetura P2P

rastrear o endereço IP dos usuários, mas as mensagens de usuário para usuário são enviadas diretamente entre os hospedeiros do usuário (sem passar por servidores intermediários). Uma das características mais fortes da arquitetura P2P é sua autoescalabilidade. Por exemplo, em uma aplicação de compartilhamento de arquivo P2P, embora cada par gere uma carga de trabalho solicitando arquivos, cada par também acrescenta capacidade de serviço ao sistema distribuindo arquivos a outros pares. As arquiteturas P2P também possuem uma boa relação custo-benefício, visto que normalmente não requerem uma infraestrutura de servidor significativa e uma largura de banda de servidor. A fim de reduzir os custos, os provedores de serviço (MSN, Yahoo etc.) estão cada vez mais interessados em utilizar arquiteturas P2P para suas aplicações [Chuang, 2007]. Entretanto, as futuras aplicações P2P estão diante de três principais desafios:

1. **ISP Amigável.** A maioria dos ISPs residenciais (incluindo o DSL e os ISPs a cabo) foram dimensionados para uso de largura de banda assimétrica, ou seja, para muito mais tráfego de entrada do que de saída. Mas a transmissão de vídeo P2P e as aplicações de distribuição de vídeo transferem o tráfego de saída dos servidores para ISPs residenciais, colocando, assim, uma pressão significativa nos ISPs. As futuras aplicações P2P precisam ser criadas para que sejam amigáveis aos ISPs [Xie, 2008].
2. **Segurança.** Em razão de sua natureza altamente distribuída e exposta, as aplicações P2P podem ser um desafio para proteger [Doucer, 2002; Yu, 2006; Liang, 2006; Naoumov, 2006; Dhungel, 2008].
3. **Incentivos.** O sucesso das futuras aplicações P2P também depende de usuários participativos para oferecer largura de banda, armazenamento e recursos da computação às aplicações, um projeto desafiador de incentivo [Feldman, 2005; Piatek, 2008; Aperjis, 2008].

2.1.2 Comunicação entre processos

Antes de construir sua aplicação de rede, você também precisará ter um entendimento básico de como programas que rodam em vários sistemas finais comunicam-se entre si. No jargão de sistemas

operacionais, na verdade não são programas, mas **processos que se comunicam**. Um processo pode ser imaginado como um programa que está rodando dentro de um sistema final. Quando os processos estão rodando no mesmo sistema final, eles comunicam-se entre si usando comunicação interprocessos, cujas regras são determinadas pelo sistema operacional do sistema final. Porém, neste livro, não estamos interessados em como se comunicam processos que estão no mesmo hospedeiro, mas em como se comunicam processos que rodam em sistemas finais *diferentes* (com sistemas operacionais potencialmente diferentes).

Eles se comunicam pela troca de mensagens por meio da rede de computadores. Um processo originador cria e envia mensagens para a rede; um processo destinatário recebe-as e possivelmente responde, devolvendo outras. A Figura 2.1 mostra que processos se comunicam usando a camada de aplicação da pilha de cinco camadas da arquitetura.

Processos clientes e processos servidores

Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede. Por exemplo, na aplicação Web, o processo browser de um cliente troca mensagens com o processo de um servidor Web. Em um sistema de compartilhamento de arquivos P2P, um arquivo é transferido de um processo que está em um par para outro que está em outro par. Para cada par de processos comunicantes normalmente rotulamos um dos dois processos de cliente e o outro, de servidor. Na Web, um browser é um processo cliente e um servidor Web é um processo servidor. No compartilhamento de arquivos P2P, o par que está enviando o arquivo é rotulado de cliente e o que está recebendo, de servidor.

Talvez você já tenha observado que, em algumas aplicações, tal como compartilhamento de arquivos P2P, um processo pode ser ambos, cliente e servidor. Realmente, um processo em um sistema de compartilhamento de arquivos P2P pode carregar e descarregar arquivos. Mesmo assim, no contexto de qualquer dada sessão entre um par de processos, ainda podemos rotular um processo de cliente e o outro de servidor. Definimos os processos cliente e servidor como segue:

*No contexto de uma sessão de comunicação entre um par de processos, o processo que inicia a comunicação (isto é, o primeiro a contatar o outro no início da sessão) é rotulado de **cliente**. O processo que espera ser contatado para iniciar a sessão é o **servidor**.*

Na Web, um processo do browser inicia o contato com um processo do servidor Web; por conseguinte, o processo do browser é o cliente e o processo do servidor Web é o servidor. No compartilhamento de arquivos P2P, quando o Par A solicita ao Par B o envio de um arquivo específico, o Par A é o cliente enquanto o Par B é o servidor no contexto dessa sessão específica de comunicação. Quando não houver possibilidade de confusão, às vezes usaremos também a terminologia “lado cliente e lado servidor de uma aplicação”. No final deste capítulo examinaremos passo a passo um código simples para ambos os lados de aplicações de rede: o lado cliente e o lado servidor.

A interface entre o processo e a rede de computadores

Como dissemos anteriormente, a maioria das aplicações consiste em pares de processos comunicantes, sendo que os dois processos de cada par enviam mensagens um para o outro. Qualquer mensagem enviada de um processo para um outro tem de passar pela rede subjacente. Um processo envia mensagens para a rede e recebe mensagens dela através de uma interface de software denominada socket. Vamos considerar uma analogia que nos auxiliará a entender processos e sockets. Um processo é análogo a uma casa e seu socket, à porta da casa. Quando um processo quer enviar uma mensagem a um outro processo em outro hospedeiro, ele empurra a mensagem pela porta (socket). Esse processo emissor admite que exista uma infraestrutura de transporte do outro lado de sua porta que transportará a mensagem pela rede até a porta do processo destinatário. Ao chegar ao hospedeiro destinatário, a mensagem passa através da porta (socket) do processo receptor, que então executa alguma ação sobre a mensagem.

A Figura 2.3 ilustra a comunicação por socket entre dois processos que se comunicam pela Internet. (A Figura 2.3 admite que o protocolo de transporte subjacente usado pelos processos é o TCP.) Como mostra essa figura, um socket é a interface entre a camada de aplicação e a de transporte dentro de uma máquina. É também denominado interface de programação da aplicação (*application programming interface — API*) entre a aplicação e a rede, visto que o socket é a interface de programação pela qual as aplicações de rede são inseridas na Internet. O desenvolvedor da aplicação controla tudo o que existe no lado da camada de aplicação do socket, mas tem pouco controle do lado da camada de transporte do socket. Os únicos controles que o desenvolvedor da aplicação tem do lado da camada de transporte são: (1) a escolha do protocolo de transporte e (2), talvez, a capacidade de determinar alguns parâmetros da camada de transporte, tais como tamanho máximo de buffer e de segmentos (a serem abordados no Capítulo 3). Uma vez escolhido um protocolo de transporte, (se houver escolha) o desenvolvedor constrói a aplicação usando os serviços da camada de transporte oferecidos por esse protocolo. Examinaremos sockets mais detalhadamente nas seções 2.7 e 2.8.

2.1.3 Serviços de transporte disponíveis para aplicações

Lembre-se de que um socket é a interface entre o processo da aplicação e o protocolo de camada de transporte. A aplicação do lado remetente envia mensagens através do socket. Do outro lado do socket, o protocolo de camada de transporte tem a responsabilidade de levar as mensagens pela rede até a “porta” do socket destinatário. Muitas redes, inclusive a Internet, oferecem mais de um protocolo de camada de transporte. Ao desenvolver uma aplicação, você deve escolher um dos protocolos de camada de transporte disponíveis. Como fazer essa escolha? O mais provável é que você avalie os serviços providos pelos protocolos de camada de transporte disponíveis e escolha o protocolo que melhor atenda às necessidades de sua aplicação. A situação é semelhante a escolher trem ou avião como meio de transporte entre duas cidades. Você tem de escolher um ou outro, e cada modalidade de transporte oferece serviços diferentes. Por exemplo, o trem oferece a facilidade da partida e da chegada no centro da cidade, ao passo que o avião oferece menor tempo de viagem.

Quais são os serviços que um protocolo da camada de transporte pode oferecer às aplicações que o chamem? Podemos classificar, de maneira geral, os possíveis serviços segundo quatro dimensões: transferência confiável de dados, vazão, temporização e segurança.

Transferência confiável de dados

Como discutido no Capítulo 1, os pacotes podem se perder dentro de uma rede de computador. Um pacote pode, por exemplo, exceder um buffer em um roteador, ou ser descartado por um hospedeiro ou um roteador

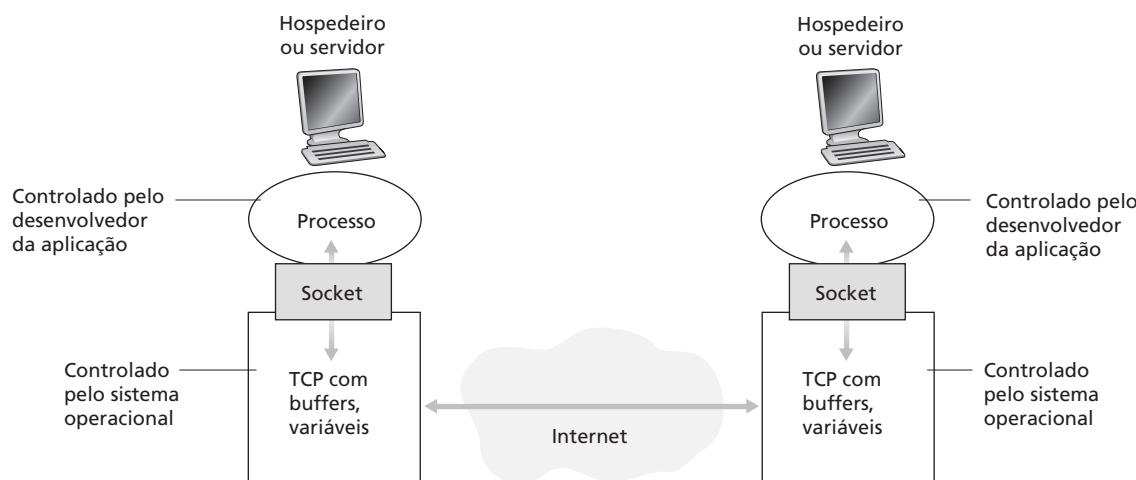


Figura 2.3 Processos de aplicação, sockets e protocolo de transporte subjacente

após alguns de seus bits terem sido corrompidos. Para muitas aplicações — como correio eletrônico, transferência de arquivo, acesso remoto, transferências de documentos da Web e aplicações financeiras — a perda de dados pode ter consequências devastadoras (no último caso, para o banco e para o cliente!). Assim, para suportar essas aplicações, algo deve ser feito para garantir que os dados enviados por uma extremidade da aplicação sejam enviados correta e completamente para a outra extremidade da aplicação. Se um protocolo fornecer um serviço de recebimento de dados garantidos, ele fornecerá uma transferência confiável de dados. Um importante serviço que o protocolo da camada de transporte pode oferecer para uma aplicação é a transferência confiável de dados processo a processo. Quando um protocolo de transporte oferece esse serviço, o processo remetente pode passar seus dados para um socket e saber com absoluta confiança que os dados chegarão sem erro ao processo destinatário.

Quando um protocolo da camada de transporte não oferece uma transferência confiável de dados, os dados enviados pelo processo remetente podem nunca chegar ao processo destinatário. Isso pode ser aceitável para aplicações tolerantes a perda, aplicações de multimídia como áudio/vídeo em tempo real ou áudio/vídeo armazenado, que podem tolerar uma quantidade de perda de dados. Nessas aplicações multimídia, dados perdidos podem resultar em uma pequena falha durante a execução do áudio/vídeo — o que não é um prejuízo crucial.

Vazão

No Capítulo 1 apresentamos o conceito de vazão disponível, que, no contexto de sessão da comunicação entre dois processos ao longo de um caminho de rede, é a taxa à qual o processo remetente pode enviar bits ao processo destinatário. Como outras sessões compartilharão a largura de banda no caminho da rede e estão indo e voltando, a vazão disponível pode oscilar com o tempo. Essas observações levam a outro serviço natural que um protocolo da camada de transporte pode oferecer, ou seja, uma vazão disponível garantida a uma taxa específica. Com tal serviço, a aplicação pode solicitar uma vazão garantida de r bits/s, e o protocolo de transporte garante, então, que a vazão disponível seja sempre r bits/s, pelo menos. Tal serviço de vazão garantida aparece em muitas aplicações.

Por exemplo, se uma aplicação de telefonia por Internet codifica voz a 32 bits/s, ela precisa enviar dados para a rede e fazer com que sejam entregues na aplicação receptora a essa mesma taxa. Se o protocolo de transporte não puder fornecer essa vazão, a aplicação precisará codificar a uma taxa menor (e receber vazão suficiente para sustentar essa taxa de codificação mais baixa) ou então desistir, já que receber metade da vazão de que precisa de nada adianta para essa **aplicação** de telefonia por Internet. Aplicações que possuam necessidade de vazão são conhecidas como aplicações sensíveis à largura de banda. Muitas aplicações de multimídia existentes são sensíveis à largura de banda, embora algumas poderão usar técnicas adaptativas de codificação para codificar a uma taxa que corresponda à vazão disponível na ocasião. Embora aplicações sensíveis à largura de banda possuam necessidades específicas de vazão, as **aplicações elásticas** podem fazer uso de qualquer quantidade mínima ou máxima que por acaso esteja disponível. Correio eletrônico, transferência de arquivos e transferências Web são todas aplicações elásticas. Evidentemente, quanto mais vazão, melhor. Há um ditado que diz que “dinheiro nunca é demais”; nesse caso, podemos dizer que vazão nunca é demais!

Temporização

Um protocolo da camada de transporte pode também oferecer garantias de temporização. Como na garantia de vazão, as garantias de temporização podem surgir em diversos aspectos e modos. Podemos citar como exemplo o fato de que cada bit que o remetente insere no socket chega ao socket destinatário em menos de 100 milissegundos depois. Esse serviço seria atrativo para aplicações interativas em tempo real, como a telefonia por Internet, ambientes virtuais, teleconferência e jogos multijogadores, que exigem restrições de temporização no envio de dados para garantir eficácia. (Veja Capítulo 7, [Gauthier, 1999; Ramjee, 1994].) Longos atrasos na telefonia por Internet, por exemplo, tendem a resultar em pausas artificiais na conversação; em um jogo multusuário ou ambiente virtual interativo, um longo atraso entre realizar uma ação e ver a reação do ambiente (por exemplo, a reação de um outro jogador na outra extremidade de uma conexão fim a fim) faz com que a aplicação pareça menos realista. Para aplicações que não são em tempo real, é sempre preferível um atraso menor a um maior, mas não há nenhuma limitação estrita aos atrasos fim a fim.

Segurança

Por fim, um protocolo de transporte pode oferecer uma aplicação com um ou mais serviços de segurança. Por exemplo, no hospedeiro remetente, um protocolo de transporte pode codificar todos os dados transmitidos pelo processo remetente e, no hospedeiro destinatário, o protocolo da camada de transporte pode codificar os dados antes de enviá-los ao processo destinatário. Tal serviço pode oferecer sigilo entre os dois processos, mesmo que os dados sejam, de algum modo, observados entre os processos remetente e destinatário. Um protocolo de transporte pode, além do sigilo, fornecer outros serviços de segurança, incluindo integridade dos dados e autenticação do ponto terminal, assuntos que serão abordados em detalhes no Capítulo 8.

2.1.4 Serviços de transporte providos pela Internet

Até aqui, consideramos serviços de transportes que uma rede de computadores *poderia* oferecer em geral. Vamos agora nos aprofundar mais no assunto e analisar o tipo de suporte de aplicação provido pela Internet. A Internet (e, em um amplo sentido, as redes TCP/IP) disponibiliza dois protocolos de transporte para aplicações, o UDP e o TCP. Quando você (como um criador de aplicação) cria uma nova aplicação de rede para a Internet, uma das primeiras decisões a ser tomada é usar o UDP ou o TCP. Cada um desses protocolos oferece um conjunto diferente de serviços para as aplicações solicitantes. A Figura 2.4 mostra os requisitos do serviço para algumas aplicações selecionadas.

Serviços do TCP

O modelo de serviço TCP inclui um serviço orientado para conexão e um serviço confiável de transferência de dados. Quando uma aplicação solicita o TCP como seu protocolo de transporte, recebe dele ambos os serviços.

Serviço orientado para conexão: o TCP faz com que o cliente e o servidor troquem informações de controle de camada de transporte antes que as mensagens de camada de aplicação comecem a fluir. Esse procedimento de apresentação, por assim dizer, alerta o cliente e o servidor, permitindo que eles se preparem para uma enxurrada de pacotes. Após a fase de apresentação, dizemos que existe uma **conexão TCP** entre os sockets dos dois processos. A conexão é full-duplex (simultânea), visto que os dois processos podem enviar mensagens um ao outro pela conexão ao mesmo tempo. Quando termina de enviar mensagens, a aplicação deve interromper a conexão. Esse serviço é chamado de serviço “orientado para conexão”, e não serviço “de conexão”, porque os dois processos estão conectados de um modo muito solto. No Capítulo 3, discutiremos detalhadamente serviço orientado para conexão e examinaremos como ele é implementado.

Serviço confiável de transporte: os processos comunicantes podem confiar no TCP para a entrega de todos os dados enviados sem erro e na ordem correta. Quando um lado da aplicação passa uma cadeia

Aplicação	Perda de dados	Largura de banda	Sensibilidade ao atraso
Transferência de arquivos	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns kbps)	Não
Telefonia via Internet/videoconferência	Tolerante à perda	Áudio: alguns kbps – 1 Mbps Vídeo: 10 kbps – 5 Mbps	Sim: décimos de segundo
Áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Alguns kbps – 10 Mbps	Sim: décimos de segundo
Mensagem instantânea	Sem perda	Elástica	Sim e não

Figura 2.4 Requisitos de aplicações de rede selecionadas

de bytes para dentro de um socket, pode contar com o TCP para entregar a mesma cadeia de dados ao socket receptor, sem falta de bytes nem bytes duplicados.

O TCP também inclui um mecanismo de controle de congestionamento, um serviço voltado ao bem-estar geral da Internet e não ao benefício direto dos processos comunicantes. O mecanismo de controle de congestionamento do TCP limita a capacidade de transmissão de um processo (cliente ou servidor) quando a rede está congestionada entre cliente e servidor. Em particular, como veremos no Capítulo 3, o controle de congestionamento do TCP tenta limitar cada conexão do TCP à sua justa porção de largura de banda de rede. A limitação da velocidade de transmissão pode ter um efeito muito prejudicial sobre aplicações de áudio e vídeo em tempo real que imponham uma limitação de largura de banda mínima. Além disso, aplicações em tempo real são tolerantes à perda e não necessitam de um serviço de transporte totalmente confiável. Por essas razões, desenvolvedores de aplicações em tempo real usualmente executam suas aplicações em UDP, e não em TCP.

Serviços do UDP

O UDP é um protocolo de transporte simplificado, leve, com um modelo de serviço minimalista. É um serviço não orientado para conexão; portanto, não há apresentação antes que os dois processos comecem a se comunicar. O UDP provê um serviço não confiável de transferência de dados — isto é, quando um processo envia uma mensagem para dentro de um socket UDP, o protocolo não oferece *nenhuma* garantia de que a mensagem chegará ao processo receptor. Além do mais, mensagens que realmente chegam ao processo receptor podem chegar fora de ordem.

O UDP não inclui um mecanismo de controle de congestionamento; portanto, um processo originador pode bombear dados para dentro de uma camada abaixo (a camada de rede) à taxa que quiser. (Observe, entretanto, que a vazão fim a fim pode ser menor do que essa taxa devido à largura de banda limitada de enlaces intervenientes ou ao congestionamento). Como aplicações em tempo real usualmente podem tolerar uma certa perda



Segurança em foco

Um conjunto impressionante de sistemas finais da Internet

protegendo o TCP

Nem o TCP ou o UDP fornecem qualquer codificação — os dados que o processo remetente transfere para seu socket são os mesmos que percorrem a rede até o processo destinatário. Então, por exemplo, se o processo destinatário enviar uma senha em *cleartext* (ou seja, não codificado) para seu socket, ela percorrerá por todos os enlaces entre o remetente e o destinatário, sendo analisada e descoberta em qualquer um dos enlaces intervenientes. Em razão da privacidade e outras questões de segurança terem se tornado importantes para muitas aplicações, a comunidade da Internet desenvolveu um aperfeiçoamento para o TCP, denominado Camada de Sockets Seguros (SSL). O aperfeiçoamento SSL para o TCP não somente faz tudo que o TCP tradicional faz, como também oferece serviços importantes de segurança para o processo, incluindo codificação, integridade dos dados e autenticação do ponto de chegada. Salientamos que o SSL não é um terceiro protocolo da Internet, no mesmo nível do TCP e do UDP, mas um aperfeiçoamento do TCP implementado na camada de aplicação. Particularmente, se uma aplicação quiser utilizar o serviço do SSL, é preciso incluir o código SSL (existente, bibliotecas altamente otimizadas e as classes) da aplicação em ambas as partes cliente e servidor. O SSL possui sua própria API de socket que é semelhante à tradicional API de socket TCP. Quando uma aplicação utiliza o SSL, o processo remetente transfere dados em *cleartext* para o socket SSL; no hospedeiro emissor, então, o SSL codifica os dados e os passa para o socket TCP. Os dados codificados percorrem a Internet até o socket TCP no processo remetente. O socket remetente passa os dados codificados ao SSL, que os decodifica. Por fim, o SSL passa os dados em *cleartext* por seu socket SSL até o processo destinatário. Abordaremos o SSL em mais detalhes no Capítulo 8.

de dados, mas exigem uma taxa mínima, desenvolvedores desse tipo de aplicações frequentemente preferem executá-las por UDP, evitando, assim, o controle de congestionamento e os cabeçalhos de pacotes do TCP. Por outro lado, como muitos firewalls estão configurados para bloquear (a maioria dos tipos de) tráfego UDP, os projetistas têm escolhido, cada vez mais, executar aplicações multimídia e em tempo real por meio do TCP [Sripanidkulchai, 2004].

Serviços não providos pelos protocolos de transporte da Internet

Organizamos os serviços do protocolo de transporte em quatro dimensões: transferência confiável de dados, vazão, temporização e segurança. Quais desses serviços são providos pelo TCP e pelo UDP? Já vimos que o TCP provê a transferência confiável de dados fim a fim, e sabemos também que ele pode ser aprimorado facilmente na camada de aplicação com o SSL para oferecer serviços de segurança. Mas em nossa breve descrição sobre o TCP e o UDP, faltou mencionar as garantias de vazão e de temporização — serviços não fornecidos pelos protocolos de transporte da Internet de hoje. Isso significa que as aplicações sensíveis ao tempo, como a telefonia por Internet não podem rodar na Internet atual? A resposta é claramente negativa — a Internet tem recebido essas aplicações por muitos anos. Tais aplicações muitas vezes funcionam bem, por terem sido desenvolvidas para lidar, na medida do possível, com essa falta de garantia. Analisaremos vários desses artifícios de criação no Capítulo 7. No entanto, a criação engenhosa possui suas limitações quando o atraso é excessivo, como é o caso frequente da Internet pública. Em resumo, a Internet de hoje pode oferecer serviços satisfatórios a aplicações sensíveis ao tempo, mas não garantias de temporização ou de largura de banda.

A Figura 2.5 mostra os protocolos de transporte usados por algumas aplicações populares da Internet. Vemos que e-mail, a Web, acesso a terminais remotos e transferência de arquivos usam o TCP. Essas aplicações escolheram o TCP primordialmente porque ele oferece um serviço confiável de transferência de dados, garantindo que todos eles mais cedo ou mais tarde cheguem a seu destino. Vemos também que a aplicação de telefonia por Internet normalmente funciona em UDP. Cada lado de uma aplicação de telefone por Internet precisa enviar dados pela rede a uma taxa mínima (veja a Figura 2.4), o que é mais provavelmente possível com UDP do que com TCP. E, também, aplicações de telefone por Internet são tolerantes às perdas, de modo que não necessitam do serviço confiável de transferência de dados provido pelo TCP.

Endereçamento de processos

Nossa discussão acima se focou nos serviços de transporte entre dois processos de comunicação. Mas como um processo indica qual processo ele quer para se comunicar usando esses serviços? Como um processo rodando em um hospedeiro em Amherst, Massachusetts, EUA, especifica que ele quer se comunicar com um processo em particular rodando em um hospedeiro em Bangkok, na Tailândia? Para isso, é preciso especificar duas informações: (1) o nome ou o endereço da máquina hospedeira e (2) um identificador que especifique o processo destinatário no hospedeiro de destino.

Aplicações	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP (RFC 5321)	TCP
Acesso a terminal remoto	Telnet (RFC 854)	TCP
Web	HTTP (RFC 2616)	TCP
Transferência de arquivos	FTP (RFC 959)	TCP
Multimídia em tempo real	HTTP (por exemplo, YouTube), RTP	UDP ou TCP
Telefonia por Internet	SIP, RTP ou proprietária (por exemplo, Skype)	Tipicamente UDP

Figura 2.5 Aplicações populares da Internet, seus protocolos de camada de aplicação e seus protocolos de transporte subjacentes



Na Internet, o hospedeiro é identificado por seu endereço IP. Discutiremos endereços IP em detalhes no Capítulo 4. Por enquanto, basta saber que o endereço IP é uma quantidade de 32 bits que identifica *exclusivamente* o sistema final. (Entretanto, como veremos no Capítulo 4, a implementação ampla do NAT (Tradução de Endereços de Rede) significa que, na prática, um endereço IP de 32 bits sozinho não endereça exclusivamente um hospedeiro.)

Além de saber o endereço do sistema final ao qual a mensagem se destina, o hospedeiro originador também deve identificar o processo que está rodando no outro hospedeiro. Essa informação é necessária porque, em geral, um hospedeiro poderia estar executando muitas aplicações de rede. Um **número de porta** de destino atende a essa finalidade. Aplicações populares receberam números de porta específicos. Por exemplo, um servidor Web é identificado pelo número de porta 80. Um processo servidor de correio (que usa o protocolo SMTP) é identificado pelo número de porta 25. Uma lista de números bem conhecidos de portas para todos os protocolos padronizados da Internet pode ser encontrada no site <http://www.iana.org>. Quando um desenvolvedor cria uma nova aplicação de rede, ela deve receber um novo número de porta. Examinaremos números de porta detalhadamente no Capítulo 3.

2.1.5 Protocolos de camada de aplicação

Acabamos de aprender que processos de rede comunicam-se entre si enviando mensagens para dentro de sockets. Mas, como essas mensagens são estruturadas? O que significam os vários campos nas mensagens? Quando os processos enviam as mensagens? Essas perguntas nos transportam para o mundo dos protocolos de camada de aplicação. Um **protocolo de camada de aplicação** define como processos de uma aplicação, que funcionam em sistemas finais diferentes, passam mensagens entre si. Em particular, um protocolo de camada de aplicação define:

- os tipos de mensagens trocadas, por exemplo, de requisição e de resposta;
- a sintaxe dos vários tipos de mensagens, tais como os campos da mensagem e como os campos são delineados;
- a semântica dos campos, isto é, o significado da informação nos campos;
- regras para determinar quando e como um processo envia mensagens e responde a mensagens.

Alguns protocolos de camada de aplicação estão especificados em RFCs e, portanto, são de domínio público. Por exemplo, o protocolo de camada de aplicação da Web, HTTP (HyperText Transfer Protocol [RFC 2616]), está à disposição como um RFC. Se um desenvolvedor de browser seguir as regras do RFC do HTTP, seu browser estará habilitado a extrair páginas de qualquer servidor Web que também tenha seguido as regras do RFC do HTTP. Muitos outros protocolos de camada de aplicação são proprietários e, intencionalmente, não estão disponíveis ao público. Por exemplo, muitos dos sistemas de compartilhamento de arquivos P2P existentes usam protocolos de camada de aplicação proprietários.

É importante distinguir aplicações de rede de protocolos de camada de aplicação. Um protocolo de camada de aplicação é apenas um pedaço (embora grande) de aplicação de rede. Examinemos alguns exemplos. A Web é uma aplicação cliente-servidor que permite aos usuários obter documentos de servidores Web por demanda. A aplicação Web consiste em muitos componentes, entre eles um padrão para formato de documentos (isto é, HTML), browsers Web (por exemplo, Firefox e Microsoft Internet Explorer), servidores Web (por exemplo, servidores Apache e Microsoft) e um protocolo de camada de aplicação. O protocolo de camada de aplicação da Web, HTTP, define o formato e a sequência das mensagens que são passadas entre o browser e o servidor Web. Assim, ele é apenas um pedaço (embora importante) da aplicação Web. Como outro exemplo, considere a aplicação correio eletrônico da Internet que também tem muitos componentes, entre eles servidores de correio que armazenam caixas postais de usuários, leitores de correio que permitem aos usuários ler e criar mensagens, um padrão que define como mensagens são passadas entre servidores e entre servidores e leitores de correio e como o conteúdo de certas partes da mensagem de correio (por exemplo, um cabeçalho) deve ser interpretado. O principal protocolo de camada de aplicação para o correio eletrônico é o SMTP — Simple Mail Protocol [RFC 5321]. Assim, o SMTP é apenas um pedaço (embora importante) da aplicação correio eletrônico.

2.1.6 Aplicações de rede abordadas neste livro

Novas aplicações de Internet de domínio público e proprietárias são desenvolvidas todos os dias. Em vez de tratarmos de um grande número dessas aplicações de maneira enciclopédica, preferimos focalizar um pequeno número de aplicações ao mesmo tempo importantes e populares. Neste capítulo, discutiremos cinco aplicações populares: a Web, a transferência de arquivos, o correio eletrônico, o serviço de diretório e aplicações P2P. Discutiremos primeiramente a Web não somente porque ela é uma aplicação imensamente popular, mas também porque seu protocolo de camada de aplicação, HTTP, é direto e fácil de entender. Após examinarmos a Web, examinaremos brevemente o FTP porque este protocolo oferece um ótimo contraste com o HTTP. Em seguida, discutiremos o correio eletrônico, a primeira aplicação de enorme sucesso da Internet. O correio eletrônico é mais complexo do que a Web, no sentido de que usa não somente um, mas vários protocolos de camada de aplicação. Após o e-mail, estudaremos o DNS, que provê um serviço de diretório para a Internet. A maioria dos usuários não interage diretamente com o DNS; em vez disso, eles o chamam indiretamente por meio de outras aplicações (inclusive a Web, a transferência de arquivos e o correio eletrônico). O DNS ilustra primorosamente como um componente de funcionalidade de núcleo de rede (tradução de nome de rede para endereço de rede) pode ser implementado na camada de aplicação da Internet. Finalmente, discutiremos o compartilhamento de arquivos P2P que, segundo algumas medições (por exemplo, tráfego de rede), é a classe de aplicações mais popular da Internet de hoje. Finalmente, discutiremos várias aplicações P2P, incluindo distribuição de arquivo, banco de dados distribuídos e telefonia IP.

2.2 A Web e o HTTP

Até a década de 1990, a Internet era usada primordialmente por pesquisadores, acadêmicos e estudantes universitários para se interligar com hospedeiros remotos, transferir arquivos de hospedeiros locais para hospedeiros remotos e vice-versa, enviar e receber notícias e enviar e receber correio eletrônico. Embora essas aplicações fossem (e continuem a ser) extremamente úteis, a Internet não era conhecida fora das comunidades acadêmicas e de pesquisa. Então, no início da década de 1990, entrou em cena uma nova aplicação importântissima — a World Wide Web [Berners-Lee, 1994]. A Web é a aplicação da Internet que chamou a atenção do público em geral. Ela transformou drasticamente a maneira como pessoas interagem dentro e fora de seus ambientes de trabalho. Alçou a Internet de apenas mais uma entre muitas redes de dados para, essencialmente, a única rede de dados.

Talvez o que mais atraia a maioria dos usuários da Web é que ela funciona *por demanda*. Usuários recebem o que querem, quando querem, o que é diferente da transmissão de rádio e de televisão, que força o usuário a sintonizar quando o provedor disponibiliza o conteúdo. Além de funcionar por demanda, a Web tem muitas outras características maravilhosas que as pessoas adoram. É muitíssimo fácil para qualquer indivíduo fazer com que informações fiquem disponíveis na Web; todo mundo pode se transformar em editor a um custo extremamente baixo. Hiperenlaces e buscadores nos ajudam a navegar pelo oceano dos sites Web. Dispositivos gráficos estimulam nossos sentidos. Formulários, applets Java e muitos outros dispositivos nos habilitam a interagir com páginas e sites. E, cada vez mais, a Web oferece um menu de interfaces para vastas quantidades de material de vídeo e áudio armazenadas na Internet — áudio e vídeo que podem ser acessados por demanda.

2.2.1 Descrição geral do HTTP

O **HTTP — Protocolo de Transferência de Hipertexto** (HyperText Transfer Protocol) — protocolo de camada de aplicação da Web, está no coração da Web e é definido no [RFC 1945] e no [RFC 2616]. O HTTP é implementado em dois programas: um programa cliente e outro servidor. Os dois programas, executados em sistemas finais diferentes, conversam um com o outro por meio da troca de mensagens HTTP. O HTTP define a

estrutura dessas mensagens e o modo como o cliente e o servidor as trocam. Antes de explicarmos detalhadamente o HTTP, devemos revisar a terminologia da Web.

Uma **página Web** (também denominada documento) é constituída de objetos. Um **objeto** é simplesmente um arquivo — tal como um arquivo HTML, uma imagem JPEG, um applet Java, ou um clipe de vídeo — que se pode acessar com um único URL. A maioria das páginas Web é constituída de um **arquivo-base HTML** e diversos objetos referenciados. Por exemplo, se uma página Web contiver um texto HTML e cinco imagens JPEG, então ela terá seis objetos: o arquivo-base HTML e mais as cinco imagens. O arquivo-base HTML referencia os outros objetos na página com os URLs dos objetos. Cada URL tem dois componentes: o nome do hospedeiro do servidor que abriga o objeto e o nome do caminho do objeto. Por exemplo, no URL

`http://www.someSchool.edu/someDepartment/picture.gif`

`www.someSchool.edu` é o nome do hospedeiro e `/someDepartment/picture.gif` é o do caminho. Como browsers Web também implementam o lado cliente do HTTP, podemos usar as palavras *browser* e *cliente* indiferentemente no contexto da Web. Os servidores Web também implementam o lado servidor do HTTP, abrigam objetos Web, cada um endereçado por um URL. São servidores Web populares o Apache e o Microsoft Internet Information Server.

O HTTP define como clientes Web requisitam páginas Web aos servidores e como eles as transferem a clientes. Discutiremos detalhadamente a interação entre cliente e servidor mais adiante, mas a ideia geral está ilustrada na Figura 2.6. Quando um usuário requisita uma página Web (por exemplo, clica sobre um hiperenlace), o browser envia ao servidor mensagens de requisição HTTP para os objetos da página. O servidor recebe as requisições e responde com mensagens de resposta HTTP que contêm os objetos.

O HTTP usa o TCP como seu protocolo de transporte subjacente (em vez de rodar em UDP). O cliente HTTP primeiramente inicia uma conexão TCP com o servidor. Uma vez estabelecida a conexão, os processos do browser e do servidor acessam o TCP por meio de suas interfaces sockets. Como descrito na Seção 2.1, no lado cliente a interface socket é a porta entre o processo cliente e a conexão TCP; no lado servidor, ela é a porta entre o processo servidor e a conexão TCP.

O cliente envia mensagens de requisição HTTP para sua interface socket e recebe mensagens de resposta HTTP de sua interface socket. De maneira semelhante, o servidor HTTP recebe mensagens de requisição de sua interface socket e envia mensagens de resposta para sua interface socket. Assim que o cliente envia uma mensagem para sua interface socket, a mensagem sai de suas mãos e “passa para as mãos do TCP”. Lembre-se de que na Seção 2.1 dissemos que o TCP provê ao HTTP um serviço confiável de transferência de dados, o que implica que toda mensagem de requisição HTTP emitida por um processo cliente chegará intacta ao servidor. De maneira semelhante, toda mensagem de resposta HTTP emitida pelo processo servidor chegará intacta ao cliente. Percebemos, nesse ponto, uma das grandes vantagens de uma arquitetura de camadas — o



Figura 2.6 Comportamento de requisição-resposta do HTTP

HTTP não precisa se preocupar com dados perdidos ou com detalhes de como o TCP recupera a perda de dados ou os reordena dentro da rede. Essa é a tarefa do TCP e dos protocolos das camadas mais inferiores da pilha de protocolos.

É importante notar que o servidor envia ao cliente os arquivos solicitados sem armazenar nenhuma informação de estado sobre este. Se um determinado cliente solicita o mesmo objeto duas vezes em um período de poucos segundos, o servidor não responde dizendo que acabou de enviar o objeto ao cliente; em vez disso, envia novamente o objeto, pois já esqueceu completamente o que fez antes. Como o servidor HTTP não mantém nenhuma informação sobre clientes, o HTTP é denominado um **protocolo sem estado**.

Salientamos também que a Web usa a arquitetura de aplicação cliente-servidor, como descrito na Seção 2.1. Um servidor Web está sempre em funcionamento, tem um endereço IP fixo e atende requisições de potencialmente milhões de browsers diferentes.

2.2.2 Conexões persistentes e não persistentes

Em muitas aplicações da Internet, o cliente e o servidor se comunicam por um período prolongado de tempo, em que o cliente faz uma série de requisições e o servidor responde a cada uma delas. Dependendo da aplicação e de como ela está sendo usada, a série de requisições pode ser feita de forma consecutiva, periodicamente em intervalos regulares ou esporadicamente. Quando a interação cliente-servidor acontece por meio de conexão TCP, o criador da aplicação precisa tomar uma importante decisão — cada par de requisição/resposta deve ser enviado por uma conexão TCP *distinta* ou todas as requisições e suas respostas devem ser enviadas por uma *mesma* conexão TCP? Na abordagem anterior, a aplicação utiliza conexões não persistentes; e na última abordagem, conexões persistentes. Para obter uma maior compreensão deste assunto, vamos analisar as vantagens e desvantagens das conexões não persistentes e das conexões persistentes no contexto de uma aplicação específica, o HTTP, que pode utilizar essas duas conexões. Embora o HTTP utilize conexões persistentes em seu modo padrão, os clientes e servidores HTTP podem ser configurados para utilizar a conexão não persistente.

O HTTP com conexões não persistentes

Vamos percorrer as etapas da transferência de uma página Web de um servidor para um cliente para o caso de conexões não persistentes. Vamos supor que uma página consista em um arquivo-base HTML e em dez imagens JPEG e que todos esses 11 objetos residam no mesmo servidor. Suponha também que o URL para o arquivo-base HTTP seja

`http://www.someSchool.edu/someDepartment/home.index`

Eis o que acontece:

1. O processo cliente HTTP inicia uma conexão TCP para o servidor `www.someSchool.edu` na porta número 80, que é o número de porta default para o HTTP. Associados à conexão TCP, haverá um socket no cliente e um socket no servidor.
2. O cliente HTTP envia uma mensagem de requisição HTTP ao servidor através de seu socket. Essa mensagem inclui o nome de caminho `/someDepartment/home.index`. (Discutiremos mensagens HTTP mais detalhadamente logo adiante.)
3. O processo servidor HTTP recebe a mensagem de requisição através de seu socket, extrai o objeto `/someDepartment/home.index` de seu armazenamento (RAM ou disco), encapsula o objeto em uma mensagem de resposta HTTP e a envia ao cliente através de seu socket.
4. O processo servidor HTTP ordena ao TCP que encerre a conexão TCP. (Mas, na realidade, o TCP só encerrará a conexão quando tiver certeza de que o cliente recebeu a mensagem de resposta intacta.)
5. O cliente HTTP recebe a mensagem de resposta e a conexão TCP é encerrada. A mensagem indica que o objeto encapsulado é um arquivo HTML. O cliente extrai o arquivo da mensagem de resposta, analisa o arquivo HTML e encontra referências aos dez objetos JPEG.
6. As primeiras quatro etapas são repetidas para cada um dos objetos JPEG referenciados.

À medida que recebe a página Web, o browser a apresenta ao usuário. Dois browsers diferentes podem interpretar (isto é, exibir ao usuário) uma página Web de modos ligeiramente diferentes. O HTTP não tem nada a ver com o modo como uma página Web é interpretada por um cliente. As especificações do HTTP [RFC 1945] e [RFC 2616] definem apenas o protocolo de comunicação entre o programa cliente HTTP e o programa servidor HTTP.

As etapas apresentadas ilustram a utilização de conexões não persistentes, nas quais cada conexão TCP é encerrada após o servidor enviar o objeto — a conexão não persiste para outros objetos. Note que cada conexão TCP transporta exatamente uma mensagem de requisição e uma mensagem de resposta. Assim, nesse exemplo, quando um usuário solicita a página Web, são geradas 11 conexões TCP.

Nas etapas descritas, fomos intencionalmente vagos sobre se os clientes obtêm as dez JPEGs por meio de dez conexões TCP em série ou se algumas delas são recebidas por conexões TCP paralelas. Na verdade, usuários podem configurar browsers modernos para controlar o grau de paralelismo. Nos modos default, a maioria dos browsers abre de cinco a dez conexões TCP paralelas e cada uma delas manipula uma transação requisição/resposta. Se o usuário preferir, o número máximo de conexões paralelas poderá ser fixado em um, caso em que as dez conexões são estabelecidas em série. Como veremos no próximo capítulo, a utilização de conexões paralelas reduz o tempo de resposta.

Antes de continuarmos, vamos fazer um cálculo simples para estimar o tempo que transcorre entre a requisição e o recebimento de um arquivo-base HTTP por um cliente. Para essa finalidade, definimos o **tempo de viagem de ida e volta** (*round-trip time* — RTT), ou seja, o tempo que leva para um pequeno pacote viajar do cliente ao servidor e de volta ao cliente. O RTT inclui atrasos de propagação de pacotes, de fila de pacotes em roteadores e comutadores intermediários e de processamento de pacotes. (Esses atrasos foram discutidos na Seção 1.4.) Considere, agora, o que acontece quando um usuário clica sobre um hiperenlace. Como ilustrado na Figura 2.7, isso faz com que o browser inicie uma conexão TCP entre ele e o servidor Web, o que envolve uma ‘apresentação de três vias’ — o cliente envia um pequeno segmento TCP ao servidor, o servidor o reconhece e responde com um pequeno segmento ao cliente que, por fim, o reconhece novamente para o servidor. As duas primeiras partes da apresentação de três vias representam um RTT. Após concluir essas partes, o cliente envia a mensagem de requisição HTTP combinada com a terceira parte da apresentação de três vias (o reconhecimento) para dentro da conexão TCP. Assim a mensagem de requisição chega ao servidor, este envia o arquivo HTML por meio

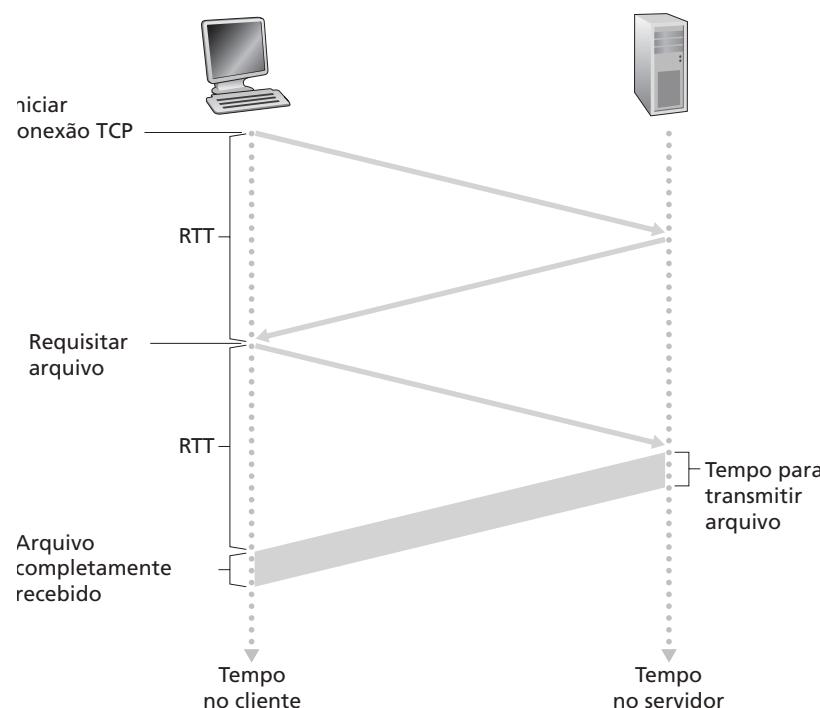


Figura 2.7 Cálculo simples para o tempo necessário para solicitar e receber um arquivo HTML

da conexão TCP. Essa requisição/resposta HTTP causa mais um RTT. Assim, aproximadamente, o tempo total de resposta são dois RTTs mais o tempo de transmissão do arquivo HTML no servidor.

O HTTP com conexões persistentes

Conexões não persistentes têm algumas desvantagens. Em primeiro lugar, uma nova conexão deve ser estabelecida e mantida para *cada objeto solicitado*. Para cada uma dessas conexões, devem ser alocados buffers TCP e conservadas variáveis TCP tanto no cliente quanto no servidor. Isso pode sobrecarregar seriamente o servidor Web, que poderá estar processando requisições de centenas de diferentes clientes ao mesmo tempo. Em segundo lugar, como acabamos de descrever, cada objeto sofre dois RTTs — um RTT para estabelecer a conexão TCP e um RTT para solicitar e receber um objeto.

Em conexões persistentes, o servidor deixa a conexão TCP aberta após enviar resposta. Requisições e respostas subsequentes entre os mesmos cliente e servidor podem ser enviadas por meio da mesma conexão. Em particular, uma página Web inteira (no exemplo anterior, o arquivo-base HTML e as dez imagens) pode ser enviada mediante uma única conexão TCP persistente. Essas requisições por objetos podem ser feitas consecutivamente sem ter de esperar por respostas a requisições pendentes (paralelismo). Normalmente, o servidor HTTP fecha uma conexão quando ela não é usada durante um certo tempo (um intervalo de pausa configurável). Quando o servidor recebe requisições consecutivas, os objetos são enviados de forma ininterrupta.

O modo default do HTTP usa conexões persistentes com paralelismo. Faremos uma comparação quantitativa entre os desempenhos de conexões persistentes e não persistentes nos exercícios de fixação dos capítulos 2 e 3. Aconselhamos o leitor interessado a consultar [Heidemann, 1997; Nielsen, 1997].

2.2.3 Formato da mensagem HTTP

As especificações do HTTP [RFC 2616] definem os formatos das mensagens HTTP. Há dois tipos de mensagens HTTP: de requisição e de resposta. Ambas serão discutidas a seguir.

Mensagem de requisição HTTP

Apresentamos a seguir uma mensagem de requisição HTTP típica:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

Podemos aprender bastante examinando essa simples mensagem de requisição. Em primeiro lugar, vemos que ela está escrita em texto ASCII comum, de modo que pode ser lida por qualquer um que conheça computadores. Em segundo lugar, vemos que ela é constituída de cinco linhas, cada uma seguida de um ‘carriage return’ e ‘line feed’ (fim de linha) para o início de uma nova linha. A última linha é seguida de um comando adicional de ‘carriage return’ e ‘line feed’. Embora essa mensagem de requisição específica tenha cinco linhas, uma mensagem de requisição pode ter muito mais do que isso e também menos do que isso, podendo conter até mesmo uma única linha. A primeira linha de uma mensagem de requisição HTTP é denominada **linha de requisição**; as subsequentes são denominadas **linhas de cabeçalho**. A linha de requisição tem três campos: o campo do método, o do URL e o da versão do HTTP. O campo do método pode assumir vários valores diferentes, entre eles GET, POST e HEAD. A grande maioria das mensagens de requisição HTTP emprega o método GET, o qual é usado quando o browser requisita um objeto e este é identificado no campo do URL. Nesse exemplo, o browser está requisitando o objeto /somedir/page.html. A versão é autoexplicativa. Nesse exemplo, o browser implementa a versão HTTP/1.1.

Vamos agora examinar as linhas de cabeçalho do exemplo. A linha de cabeçalho `Host: www.some-school.edu` especifica o hospedeiro no qual o objeto reside. Talvez você ache que ela é desnecessária, pois já existe uma conexão TCP para o hospedeiro. Mas, como veremos na Seção 2.2.5, a informação fornecida pela linha de cabeçalho do hospedeiro é requerida por armazenadores intermediários da Web. Ao incluir a linha de cabeçalho `Connection: close`, o browser está dizendo ao servidor que não quer usar conexões persistentes; quer que o servidor feche a conexão após o envio do objeto requisitado. A linha de cabeçalho `User-agent:` especifica o agente de usuário, isto é, o tipo de browser que está fazendo a requisição ao servidor. No caso em questão, o agente de usuário é o Mozilla/4.0, um browser da Netscape. Essa linha de cabeçalho é útil porque, na verdade, o servidor pode enviar versões diferentes do mesmo objeto a tipos diferentes de agentes de usuário. (Cada uma das versões é endereçada pelo mesmo URL.) Por fim, o cabeçalho `Accept-language:` mostra que o usuário prefere receber uma versão em francês do objeto se esse objeto existir no servidor; se não existir, o servidor deve enviar a versão default. O cabeçalho `Accept-language:` é apenas um dos muitos cabeçalhos de negociação de conteúdo disponíveis no HTTP.

Acabamos de examinar um exemplo. Vamos agora analisar o formato geral de uma mensagem de requisição, ilustrado na Figura 2.8.

Vemos que o formato geral de uma mensagem de requisição é muito parecido com nosso exemplo anterior. Contudo, você provavelmente notou que, após as linhas de cabeçalho (e após a linha adicional com ‘carriage return’ e ‘line feed’), há um ‘corpo de entidade’. O corpo de entidade fica vazio com o método GET, mas é utilizado com o método POST. Um cliente HTTP normalmente usa o método POST quando o usuário preenche um formulário — por exemplo, quando fornece palavras de busca a um site buscador. Com uma mensagem POST, o usuário continua solicitando uma página Web ao servidor, mas o conteúdo específico dela depende do que o usuário escreveu nos campos do formulário. Se o valor do campo de método for POST, então o corpo de entidade conterá o que o usuário digitou nos campos do formulário.

Seríamos omissos se não mencionássemos que uma requisição gerada com um formulário não utiliza necessariamente o método POST. Ao contrário, formulários HTML frequentemente utilizam o método GET e incluem os dados digitados (nos campos do formulário) no URL requisitado. Por exemplo, se um formulário usar o método GET, tiver dois campos e as entradas desses dois campos forem `monkeys` e `bananas`, então a estrutura do URL será `www.somesite.com/animalsearch?monkeys&bananas`. Ao navegar normalmente pela Web, você provavelmente já notou URLs extensos como esse.

O método HEAD é semelhante ao GET. Quando um servidor recebe uma requisição com o método HEAD, responde com uma mensagem HTTP, mas deixa de fora o objeto requisitado. Esse método é frequentemente usado pelos desenvolvedores de servidores HTTP para depuração.

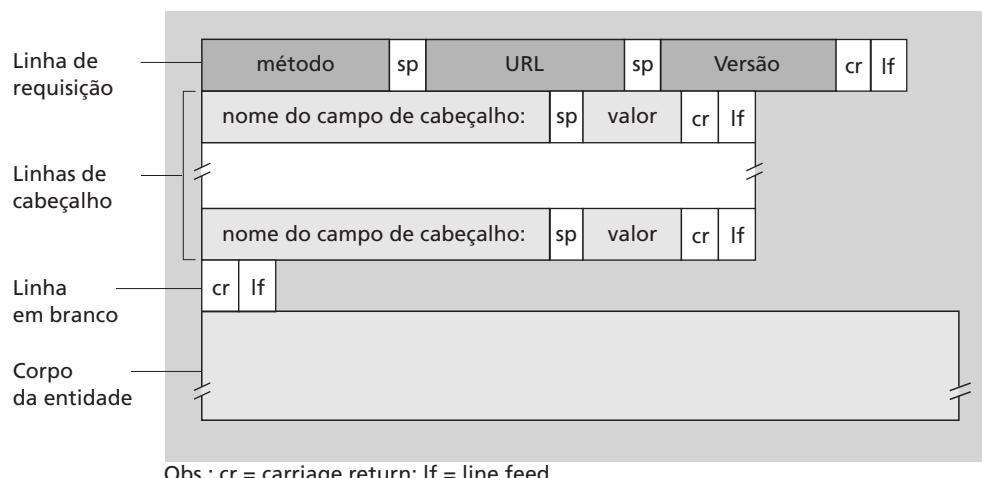


Figura 2.8 Formato geral de uma mensagem de requisição

O método PUT é frequentemente usado juntamente com ferramentas de edição da Web. Permite que um usuário carregue um objeto para um caminho específico (diretório) em um servidor Web específico. O método PUT também é usado por aplicações que precisam carregar objetos para servidores Web. O método DELETE permite que um usuário, ou uma aplicação, elimine um objeto em um servidor Web.

Mensagem de resposta HTTP

Apresentamos a seguir uma mensagem de resposta HTTP típica. Essa mensagem poderia ser a resposta ao exemplo de mensagem de requisição que acabamos de discutir.

```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 07 Jul 2007 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 6 May 2007 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

Vamos examinar cuidadosamente essa mensagem de resposta. Ela tem três seções: uma linha inicial ou **linha de estado**, seis **linhas de cabeçalho** e, em seguida, o **corpo da entidade**, que é o ‘filé mignon’ da mensagem — contém o objeto solicitado (representado por data data data data data ...). A linha de estado tem três campos: o campo de versão do protocolo, um código de estado e uma mensagem de estado correspondente. Nesse exemplo, ela mostra que o servidor está usando o HTTP/1.1 e que está tudo OK (isto é, o servidor encontrou e está enviando o objeto solicitado).

Agora, vamos examinar as linhas de cabeçalho. O servidor usa a linha de cabeçalho Connection: close para informar ao cliente que fechará a conexão TCP após enviar a mensagem. A linha de cabeçalho Date: indica a hora e a data em que a resposta HTTP foi criada e enviada pelo servidor. Note que esse não é o horário em que o objeto foi criado nem o de sua modificação mais recente; é a hora em que o servidor extraiu o objeto de seu sistema de arquivos, inseriu-o na mensagem de resposta e enviou-a. A linha de cabeçalho Server: mostra que a mensagem foi gerada por um servidor Web Apache, análoga à linha de cabeçalho User-agent: na mensagem de requisição HTTP. A linha de cabeçalho Last-Modified: indica a hora e a data em que o objeto foi criado ou sofreu a última modificação. Esse cabeçalho, que logo estudaremos mais detalhadamente, é fundamental para fazer cache do objeto, tanto no cliente local quanto em servidores de cache da rede (também conhecidos como servidores proxy). A linha de cabeçalho Content-Length: indica o número de bytes do objeto que está sendo enviado e a linha Content-Type: mostra que o objeto presente no corpo da mensagem é um texto HTML. (O tipo do objeto é oficialmente indicado pelo cabeçalho Content-Type: e não pela extensão do arquivo.)

Acabamos de ver um exemplo. Vamos agora examinar o formato geral de uma mensagem de resposta, ilustrado na Figura 2.9.

Esse formato geral de mensagem de resposta condiz com o exemplo anterior. Mas falemos um pouco mais sobre códigos de estado e suas frases, que indicam o resultado da requisição. Eis alguns códigos de estado e frases associadas comuns:

- 200 OK: requisição bem-sucedida e a informação é entregue com a resposta.
- 301 Moved Permanently: objeto requisitado foi removido permanentemente; novo URL é especificado no cabeçalho Location: da mensagem de resposta. O software do cliente recuperará automaticamente o novo URL.
- 400 Bad Request: código genérico de erro que indica que a requisição não pôde ser entendida pelo servidor.

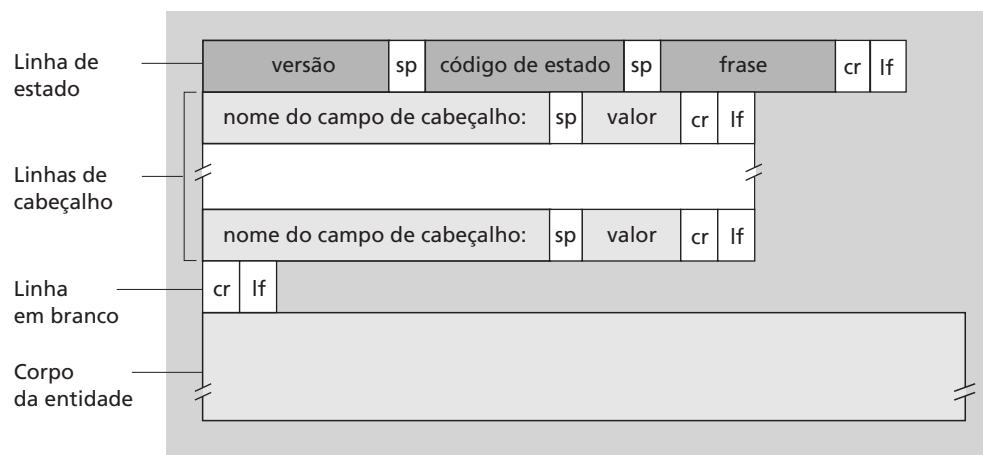


Figura 2.9 Formato geral de uma mensagem de resposta

404 Not Found: o documento requisitado não existe no servidor.

505 HTTP Version Not Supported: a versão do protocolo HTTP requisitada não é suportada pelo servidor.

Você gostaria de ver uma mensagem de resposta HTTP real? É muito recomendável e muito fácil! Primeiramente, dê um comando Telnet em seu servidor Web favorito. Em seguida, digite uma mensagem de requisição de uma linha solicitando algum objeto abrigado no servidor. Por exemplo, se você tem acesso a um prompt de comando, digite:

```
telnet cis.poly.edu 80
```

```
GET /~ross/ HTTP/1.1
```

```
Host: cis.poly.edu
```

(Clique duas vezes a tecla ‘enter’ após digitar a última linha.) Essa sequência de comandos abre uma conexão TCP para a porta número 80 do hospedeiro `cis.poly.edu` e, em seguida, envia a mensagem de requisição HTTP. Deverá aparecer uma mensagem de resposta que inclui o arquivo-base HTML da homepage do professor Ross. Se você preferir apenas ver as linhas da mensagem HTTP e não receber o objeto em si, substitua `GET` por `HEAD`. Finalmente, substitua `/~ross/` por `/~banana/` e veja que tipo de mensagem você obtém.

Nesta seção, discutimos várias linhas de cabeçalho que podem ser usadas em mensagens de requisição e de resposta HTTP. A especificação do HTTP define muitas outras linhas de cabeçalho que podem ser inseridas por browsers, servidores Web e servidores cache de redes. Vimos apenas um pouco do total de linhas de cabeçalho. Examinaremos mais algumas a seguir e mais um pouco quando discutirmos armazenagem Web na Seção 2.2.5. Uma discussão muito abrangente e fácil de ler sobre o protocolo HTTP, seus cabeçalhos e códigos de estado, pode ser encontrada em [Krishnamurty, 2001]; veja também [Luotonen, 1998] se estiver interessado no ponto de vista do desenvolvedor.

Como um browser decide quais linhas de cabeçalho serão incluídas em uma mensagem de requisição? Como um servidor Web decide quais linhas de cabeçalho serão incluídas em uma mensagem de resposta? Um browser vai gerar linhas de cabeçalho em função de seu tipo e versão (por exemplo, um browser HTTP/1.0 não vai gerar nenhuma linha de cabeçalho 1.1), da configuração do usuário para o browser (por exemplo, idioma preferido) e se o browser tem uma versão do objeto possivelmente ultrapassada em sua memória. Servidores Web se comportam de maneira semelhante: há diferentes produtos, versões e configurações, e todos influenciam as linhas de cabeçalho que são incluídas nas mensagens de resposta.

2.2.4 Interação usuário-servidor: cookies

Mencionamos anteriormente que um servidor HTTP não tem estado, o que simplifica o projeto do servidor e vem permitindo que engenheiros desenvolvam servidores Web de alto desempenho que podem manipular milhares de conexões TCP simultâneas. No entanto, é sempre bom que um site Web identifique usuários, seja porque o servidor deseja restringir acesso de usuário, seja porque quer apresentar conteúdo em função da identidade do usuário. Para essas finalidades, o HTTP usa cookies. Cookies, definidos no RFC 2965, permitem que sites monitorem seus usuários. Hoje, a maioria dos sites Web comerciais utiliza cookies.

Como ilustrado na Figura 2.10, a tecnologia dos cookies tem quatro componentes: (1) uma linha de cabeçalho de cookie na mensagem de resposta HTTP; (2) uma linha de cabeçalho de cookie na mensagem de requisição HTTP; (3) um arquivo de cookie mantido no sistema final do usuário e gerenciado pelo browser do usuário; (4) um banco de dados de apoio no site Web. Utilizando a Figura 2.10, vamos esmiuçar um exemplo de como os cookies são usados. Suponha que Susan, que sempre acessa a Web usando o Internet Explorer de seu PC, acesse o Amazon.com pela primeira vez, e que, no passado, ela já tenha visitado o site da eBay. Quando a requisição chega ao servidor Web da Amazon, ele cria um número de identificação exclusivo e uma entrada no seu banco de dados de apoio, que é indexado pelo número de identificação. Então, o servidor Web da Amazon responde ao browser de Susan, incluindo na resposta HTTP um cabeçalho Set-cookie: que contém o número de identificação. Por exemplo, a linha de cabeçalho poderia ser:

Set-cookie: 1678

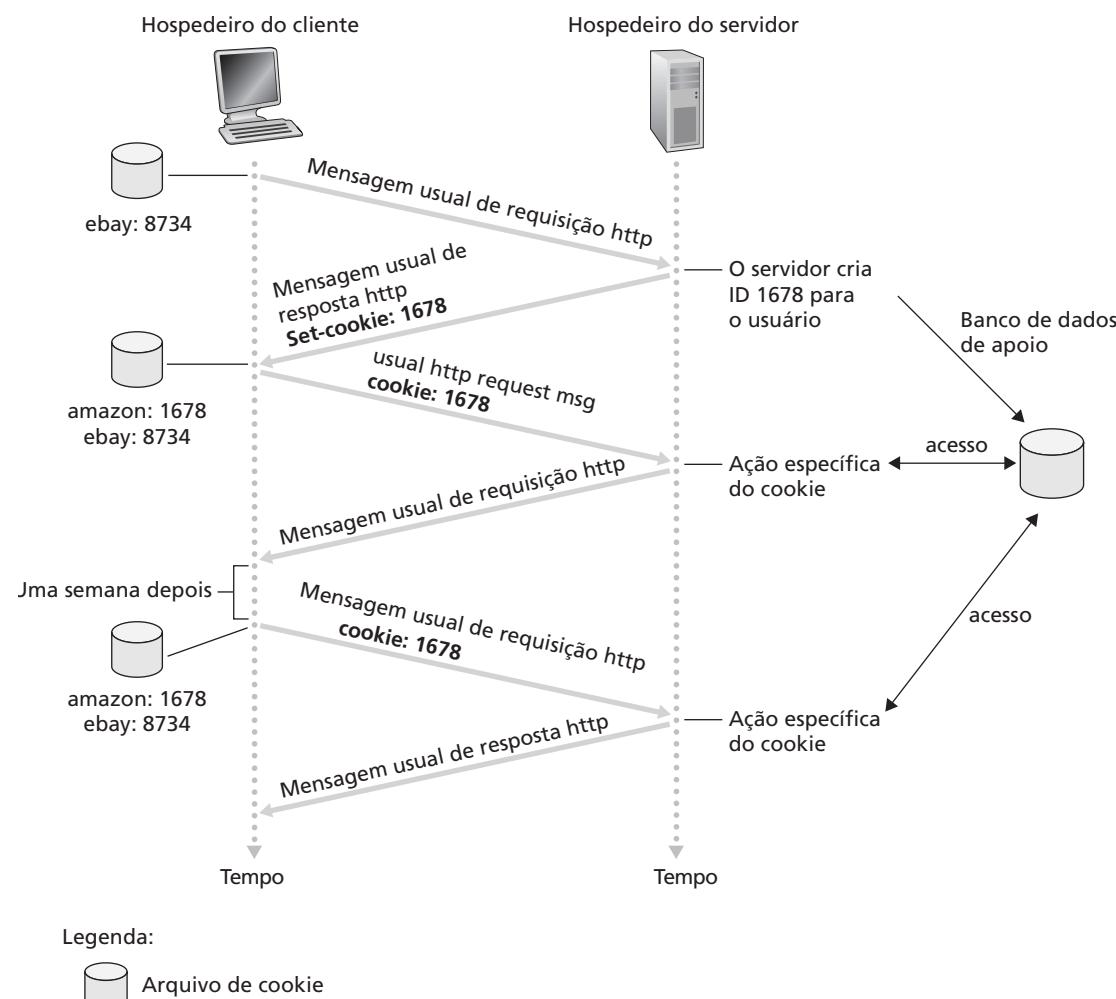


Figura 2.10 Mantendo o estado do usuário com cookies

Quando recebe a mensagem de resposta HTTP, o browser de Susan vê o cabeçalho `Set-cookie`: e, então, anexa uma linha ao arquivo especial de cookies que ele gerencia. Essa linha inclui o nome de hospedeiro do servidor e seu número de identificação nesse cabeçalho. Observe que o arquivo de cookie já possui um entrada para o eBay, uma vez que Susan já visitou esse site no passado. Toda vez que Susan requisita uma página Web enquanto navega pelo site da Amazon em questão, seu browser consulta seu arquivo de cookies, extrai seu número de identificação para esse site e insere na requisição HTTP uma linha de cabeçalho de cookie que inclui o número de identificação. Especificamente, cada uma de suas requisições HTTP ao servidor do site de comércio eletrônico inclui a linha de cabeçalho:

`Cookie: 1678`

Dessa maneira, o servidor da Amazon pode monitorar a atividade de Susan e, embora não saiba necessariamente que o nome dela é Susan, sabe exatamente quais páginas o usuário 1678 visitou, em que ordem e em que horários! Então, a Amazon pode utilizar cookies para oferecer um serviço de carrinho de compra — a Amazon pode manter uma lista de todas as compras de Susan, de modo que ela possa pagar por todas elas ao mesmo tempo, no final da sessão.

Se Susan voltar ao site, digamos, uma semana depois, seu browser continuará a inserir a linha de cabeçalho `Cookie: 1678` nas mensagens de requisição. A Amazon pode recomendar produtos a Susan com base nas páginas Web que ela visitou na Amazon anteriormente. Se ela também se registrar no site — fornecendo seu nome completo, endereço de e-mail, endereço postal e informações de cartão de crédito — a Amazon pode incluir essas informações em seu banco de dados e, assim, associar o nome de Susan com seu número de identificação (e com todas as páginas que ela consultou em suas visitas anteriores). É assim que a Amazon e outros sites de comércio eletrônico oferecem “compras com um só clique” — quando quiser comprar um item durante uma visita subsequente, Susan não precisará digitar novamente seu nome, número de seu cartão de crédito, nem endereço.

Essa discussão nos mostrou que cookies podem ser usados para identificar um usuário. Quando visitar um site pela primeira vez, um usuário pode fornecer dados de identificação (possivelmente seu nome). No decorrer das sessões subsequentes, o browser passa um cabeçalho de cookie ao servidor durante todas as visitas subsequentes ao site, identificando, desse modo, o usuário ao servidor. A discussão também deixou claro que cookies podem ser usados para criar uma camada de sessão de usuário sobre HTTP sem estado. Por exemplo, quando um usuário acessa uma aplicação de e-mail baseada na Web (como o Hotmail), o browser envia informações de cookie ao servidor que, por sua vez, identifica o usuário por meio da sessão do usuário com a aplicação.

Embora cookies frequentemente simplifiquem a experiência de compra pela Internet, continuam provocando muita controvérsia porque também podem ser considerados como invasão da privacidade de um usuário. Como acabamos de ver, usando uma combinação de cookies e informações de conta fornecidas pelo usuário, um site Web pode ficar sabendo muita coisa sobre esse usuário e, potencialmente, vender o que sabe para algum terceiro. O Cookie Central [Cookie Central, 2008] inclui informações abrangentes sobre a controvérsia dos cookies.

2.2.5 Caches Web

Um **cache Web** — também denominado **servidor proxy** — é uma entidade da rede que atende requisições HTTP em nome de um servidor Web de origem. O cache Web tem seu próprio disco de armazenagem e mantém, dentro dele, cópias de objetos recentemente requisitados. Como ilustrado na Figura 2.11, o browser de um usuário pode ser configurado de modo que todas as suas requisições HTTP sejam dirigidas primeiramente ao cache Web. Uma vez que esteja configurado um browser, cada uma das requisições de um objeto que o browser faz é primeiramente dirigida ao cache Web. Como exemplo, suponha que um browser esteja requisitando o objeto `http://www.someschool.edu/campus.gif`. Eis o que acontece:

1. O browser estabelece uma conexão TCP com o cache Web e envia a ele uma requisição HTTP para um objeto.
2. O cache Web verifica se tem uma cópia do objeto armazenada localmente. Se tiver, envia o objeto ao browser do cliente, dentro de uma mensagem de resposta HTTP.

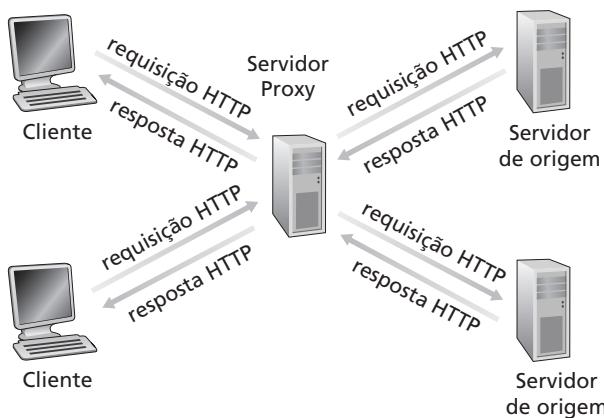


Figura 2.11 Clientes requisitando objetos por meio de um cache Web

3. Se não tiver o objeto, o cache Web abre uma conexão TCP com o servidor de origem, isto é, com `www.someschool.edu`. Então, envia uma requisição HTTP do objeto para a conexão TCP. Após receber essa requisição, o servidor de origem envia o objeto ao cache Web, dentro de uma resposta HTTP.
4. Quando recebe o objeto, o cache Web guarda uma cópia em seu armazenamento local e envia outra, dentro de uma mensagem de resposta, ao browser do cliente (pela conexão TCP existente entre o browser do cliente e o cache Web).

Note que um cache é, ao mesmo tempo, um servidor e um cliente. Quando recebe requisições de um browser e lhe envia respostas, é um servidor. Quando envia requisições para um servidor de origem e recebe respostas dele, é um cliente.

Em geral, é um ISP que compra e instala um cache Web. Por exemplo, uma universidade poderia instalar um cache na rede de seu *campus* e configurar todos os browsers apontando para esse cache. Ou um importante ISP residencial (como a AOL) poderia instalar um ou mais caches em sua rede e configurar antecipadamente os browsers que fornece apontando para os caches instalados.

O cache na Web tem sido utilizado amplamente na Internet por duas razões. Em primeiro lugar, um cache Web pode reduzir substancialmente o tempo de resposta para a requisição de um cliente, em particular se o gargalo da largura de banda entre o cliente e o servidor de origem for muito menor do que aquele entre o cliente e o cache. Se houver uma conexão de alta velocidade entre o cliente e o cache, como em geral é o caso, e se este tiver o objeto requisitado, então ele poderá entregar rapidamente o objeto ao cliente. Em segundo lugar, como logo ilustraremos com um exemplo, caches Web podem reduzir substancialmente o tráfego no enlace de acesso de uma instituição qualquer à Internet. Com a redução do tráfego, a instituição (por exemplo, uma empresa ou uma universidade) não precisa ampliar sua largura de banda tão rapidamente, o que diminui os custos. Além disso, caches Web podem reduzir substancialmente o tráfego na Internet como um todo, melhorando, assim, o desempenho para todas as aplicações.

Para entender melhor os benefícios dos caches, vamos considerar um exemplo no contexto da Figura 2.12. Essa figura mostra duas redes: uma rede institucional e a Internet pública. A rede institucional é uma LAN de alta velocidade. Um roteador da rede institucional e um roteador da Internet estão ligados por um enlace de 15 Mbps. Os servidores de origem estão todos ligados à Internet, mas localizados pelo mundo todo. Suponha que o tamanho médio do objeto seja 1 Mbit e que a taxa média de requisição dos browsers da instituição até os servidores de origem seja de 15 requisições por segundo. Imagine também que o tamanho das mensagens de requisição HTTP seja insignificante e, portanto, elas não criem tráfego nas redes ou no enlace de acesso (do roteador da instituição até o da Internet). Suponha ainda que o tempo entre o envio de uma requisição HTTP (dentro de um datagrama IP) pelo roteador do lado da Internet do enlace de acesso mostrado na Figura 2.12 e o recebimento da resposta (normalmente, dentro de muitos datagramas IPs) seja de 2 segundos em média. Esse último atraso é denominado informalmente ‘atraso da Internet’.

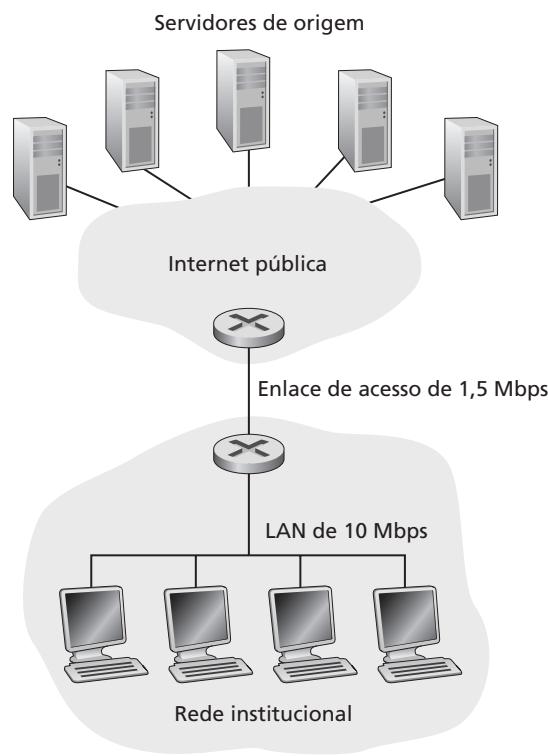


Figura 2.12 Gargalo entre uma rede institucional e a Internet

O tempo de resposta total — isto é, aquele transcorrido entre a requisição de um objeto feita pelo browser e o recebimento dele — é a soma do atraso da LAN, do atraso de acesso (isto é, o atraso entre os dois roteadores) e do atraso da Internet. Vamos fazer agora um cálculo bastante rudimentar para estimar esse atraso. A intensidade de tráfego na LAN (veja a Seção 1.4.2) é

$$(15 \text{ requisições/segundo}) \cdot (1 \text{ Mbits/requisição}) / (100 \text{ Mbps}) = 0,15$$

ao passo que a intensidade de tráfego no enlace de acesso (do roteador da Internet ao da instituição) é

$$(15 \text{ requisições/segundo}) = (1 \text{ Mbits/requisição}) / (15 \text{ Mbps}) = 1$$

Uma intensidade de tráfego de 0,15 em uma LAN resulta em, no máximo, dezenas de milissegundos de atraso; consequentemente, podemos desprezar o atraso da LAN. Contudo, como discutimos na Seção 1.4.2, à medida que a intensidade de tráfego se aproxima de 1 (como é o caso do enlace de acesso da Figura 2.12), o atraso em um enlace se torna muito grande e cresce sem limites. Assim, o tempo médio de resposta para atender requisições será da ordem de minutos, se não for maior, o que é inaceitável para os usuários da instituição. Evidentemente, algo precisa ser feito.

Uma possível solução seria aumentar a velocidade de acesso de 15 Mbps para, digamos, 100 Mbps. Isso reduziria a intensidade de tráfego no enlace de acesso a 0,15, o que se traduziria em atrasos desprezíveis entre os dois roteadores. Nesse caso, o tempo total de resposta seria aproximadamente 2 segundos, isto é, o atraso da Internet. Mas essa solução também significa que a instituição tem de atualizar seu enlace de acesso de 15 Mbps para 100 Mbps, o que pode ser muito dispendioso.

Considere agora a solução alternativa de não atualizar o enlace de acesso, mas, em vez disso, instalar um cache Web na rede institucional. Essa solução é ilustrada na Figura 2.13. A taxa de resposta local — a fração de requisições atendidas por um cache — em geral fica na faixa de 0,2 a 0,7 na prática. Para demonstrarmos isso, vamos supor que a taxa de resposta local do cache dessa instituição seja 0,4. Como os clientes e o cache estão conectados à mesma LAN de alta velocidade, 40 por cento das requisições serão atendidas quase

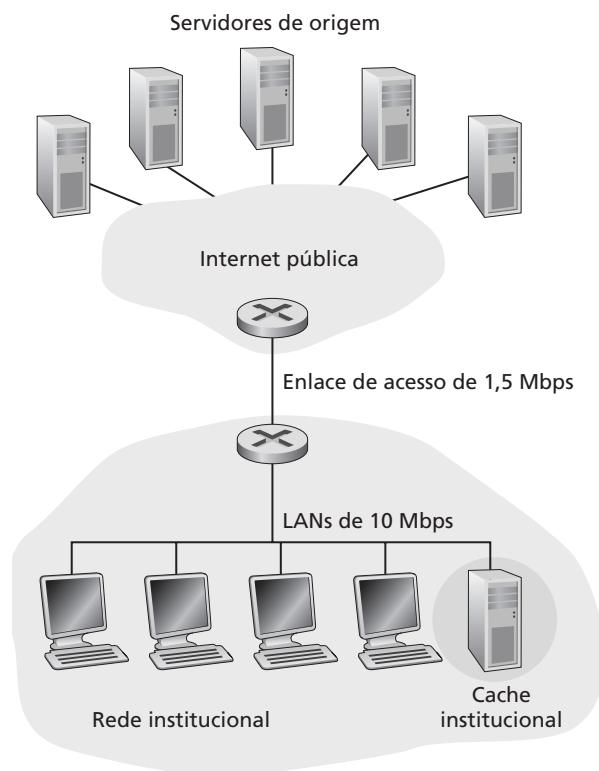


Figura 2.13 Adição de um cache à rede institucional

imediatamente, digamos, em 10 milissegundos. Mesmo assim, os demais 60 por cento das requisições ainda precisam ser atendidos pelos servidores de origem. Mas, com apenas 60 por cento dos objetos requisitados passando pelo enlace de acesso, a intensidade de tráfego neste diminui de 1,0 para 0,6. Em geral, uma intensidade de tráfego menor do que 0,8 corresponde a um atraso pequeno, digamos, dezenas de milissegundos, no caso de um enlace de 15 Mbps. Esse atraso é desprezível se comparado aos 2 segundos do atraso da Internet. Dadas essas considerações, o atraso médio é, por conseguinte, apenas ligeiramente maior do que 1,2 segundo.

$$0,4 \cdot (0,01 \text{ segundo}) + 0,6 \cdot (2,01 \text{ segundos}),$$

Assim, essa segunda solução resulta em tempo de resposta até menor do que o que se conseguiria com a troca do enlace de acesso e não requer que a instituição atualize seu enlace com a Internet. Evidentemente, a instituição terá de comprar e instalar um cache Web. Mas esse custo é baixo — muitos caches usam softwares de domínio público que rodam em PCs baratos.

2.2.6 GET condicional

Embora possa reduzir os tempos de resposta do ponto de vista do usuário, fazer cache introduz um novo problema — a cópia de um objeto existente no cache pode estar desatualizada. Em outras palavras, o objeto abrigado no servidor Web pode ter sido modificado desde a data em que a cópia entrou no cache no cliente. Felizmente, o HTTP tem um mecanismo que permite que um cache verifique se seus objetos estão atualizados. Esse mecanismo é denominado **GET condicional** (*conditional GET*). Uma mensagem de requisição HTTP é denominada uma mensagem GET condicional se (1) usar o método GET e (2) possuir uma linha de cabeçalho *If-Modified-Since*:

Para ilustrar como o GET condicional funciona, vamos examinar um exemplo. Em primeiro lugar, um cache proxy envia uma mensagem de requisição a um servidor Web em nome de um browser requisitante:



```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

Em segundo lugar, o servidor Web envia ao cache uma mensagem de resposta com o objeto requisitado:

```
HTTP/1.1 200 OK
Date: Sat, 7 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 4 Jul 2007 09:23:24
Content-Type: image/gif
```

```
(data data data data data ...)
```

O cache passa o objeto ao browser requisitante, mas também o guarda em sua memória cache local. O importante é que ele também guarda, juntamente com o objeto, a data da última modificação. Em terceiro lugar, uma semana depois, um outro browser requisita ao cache o mesmo objeto, que ainda está no cache. Como esse objeto pode ter sido modificado no servidor Web na semana anterior, o browser realiza uma verificação de atualização emitindo um GET condicional. Especificamente, o cache envia:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 4 Jul 2007 09:23:24
```

Note que o valor da linha de cabeçalho `If-modified-since:` é exatamente igual ao valor da linha de cabeçalho `Last-Modified:` que foi enviada pelo servidor há uma semana. Esse GET condicional está dizendo ao servidor para enviar o objeto somente se ele tiver sido modificado desde a data especificada. Suponha que o objeto não tenha sofrido modificação desde 4 Jul 2007 09:23:24. Então, em quarto lugar, o servidor Web envia uma mensagem de resposta ao cache:

```
HTTP/1.1 304 Not Modified
Date: Sat, 14 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)

(corpo de mensagem vazio)
```

Vemos que, em resposta ao GET condicional, o servidor Web ainda envia uma mensagem de resposta, mas não inclui nela o objeto requisitado, o que apenas desperdiçaria largura de banda e aumentaria o tempo de resposta do ponto de vista do usuário, particularmente se o objeto fosse grande. Note que, na linha de estado dessa última mensagem de resposta está inserido 304 Not Modified, que informa ao cache que ele pode seguir adiante e transmitir ao browser requisitante a cópia do objeto que está contida nele.

Finalizamos nossa discussão sobre HTTP, o primeiro protocolo da Internet (um protocolo da camada de aplicação) que estudamos em detalhes. Vimos o formato das mensagens HTTP e as ações tomadas pelo cliente e servidor Web quando essas mensagens são enviadas e recebidas. Também estudamos um pouco da infraestrutura da aplicação da Web, incluindo caches, cookies e banco de dados de apoio, todos associados, de algum modo, ao protocolo HTTP.

2.3 Transferência de arquivo: FTP

Em uma sessão FTP típica, o usuário, sentado à frente de um hospedeiro (o local), quer transferir arquivos de ou para um hospedeiro remoto. Para acessar a conta remota, o usuário deve fornecer uma identificação e uma senha. Após fornecer essas informações de autorização, pode transferir arquivos do sistema local de arquivos para o sistema remoto e vice-versa. Como mostra a Figura 2.14, o usuário interage com o FTP por meio de um agente de usuário FTP. Em primeiro lugar, ele fornece o nome do hospedeiro remoto, o que faz com que o processo cliente FTP do hospedeiro local estabeleça uma conexão TCP com o processo servidor FTP do hospedeiro remoto. O usuário então fornece sua identificação e senha, que são enviadas pela conexão TCP como parte dos

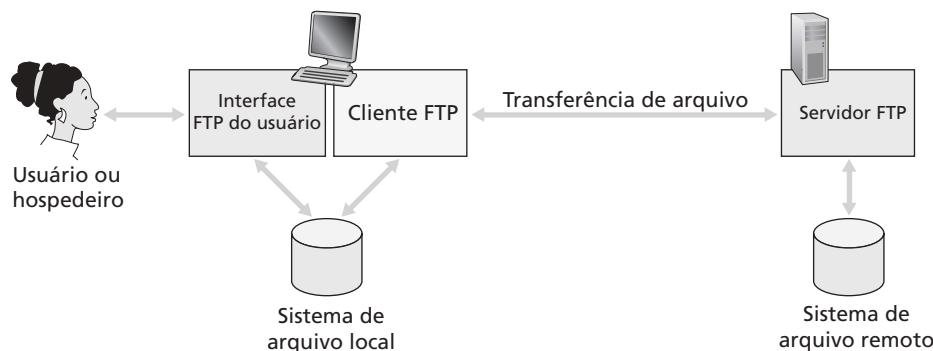


Figura 2.14 FTP transporta arquivos entre sistemas de arquivo local e remoto

comandos FTP. Assim que autorizado pelo servidor, o usuário copia um ou mais arquivos armazenados no sistema de arquivo local para o sistema de arquivo remoto (ou vice-versa).

O HTTP e o FTP são protocolos de transferência de arquivos e têm muitas características em comum; por exemplo, ambos utilizam o TCP. Contudo, esses dois protocolos de camada de aplicação têm algumas diferenças importantes. A mais notável é que o FTP usa duas conexões TCP paralelas para transferir um arquivo: uma **conexão de controle** e uma **conexão de dados**. A primeira é usada para enviar informações de controle entre os dois hospedeiros — como identificação de usuário, senha, comandos para trocar diretório remoto e comandos de ‘inserir’ e ‘pegar’ arquivos. A conexão de dados é a usada para efetivamente enviar um arquivo. Como o FTP usa uma conexão de controle separada, dizemos que ele envia suas informações de controle **fora da banda**. No Capítulo 7, veremos que o protocolo RTSP, usado para controlar a transferência de meios contínuos como áudio e vídeo, também envia suas informações de controle fora da banda. O HTTP, como você recorda, envia linhas de cabeçalho de requisição e de resposta pela mesma conexão TCP que carrega o próprio arquivo transferido. Por essa razão, dizemos que o HTTP envia suas informações de controle **na banda**. Na próxima seção, veremos que o SMTP, o principal protocolo para correio eletrônico, também envia suas informações de controle na banda. As conexões de controle e de dados do FTP estão ilustradas na Figura 2.15.

Quando um usuário inicia uma sessão FTP com um hospedeiro remoto, o lado cliente do FTP (usuário) inicia primeiramente uma conexão TCP de controle com o lado servidor (hospedeiro remoto) na porta número 21 do servidor e envia por essa conexão de controle a identificação e a senha do usuário, além de comandos para mudar o diretório remoto. Quando o lado servidor recebe, pela conexão de controle, um comando para uma transferência de arquivo (de ou para o hospedeiro remoto), abre uma conexão TCP de dados para o lado cliente. O FTP envia exatamente um arquivo pela conexão de dados e em seguida fecha-a. Se, durante a mesma sessão, o usuário quiser transferir outro arquivo, o FTP abrirá outra conexão de dados. Assim, com FTP, a conexão de controle permanece aberta durante toda a sessão do usuário, mas uma nova conexão de dados é criada para cada arquivo transferido dentro de uma sessão (ou seja, a conexão de dados é não persistente).

Durante uma sessão, o servidor FTP deve manter informações de **estado** sobre o usuário. Em particular, o servidor deve associar a conexão de controle com uma conta de usuário específica e também deve monitorar o

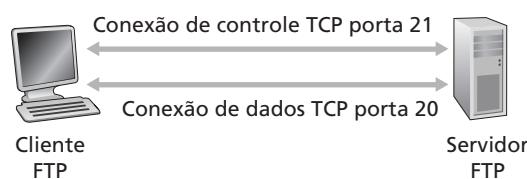


Figura 2.15 Conexões de controle e de dados



diretório corrente do usuário enquanto este passeia pela árvore do diretório remoto. Monitorar essas informações de estado para cada sessão de usuário em curso limita significativamente o número total de sessões que o FTP pode manter simultaneamente. Lembre-se de que o HTTP, por outro lado, é sem estado — não tem de monitorar o estado de nenhum usuário.

2.3.1 Comandos e respostas FTP

Encerraremos esta seção com uma breve discussão sobre alguns dos comandos mais comuns do FTP. Os comandos, do cliente para o servidor, e as respostas, do servidor para o cliente, são enviados por meio da conexão de controle no formato ASCII de 7 bits. Assim, tal como comandos HTTP, comandos FTP também podem ser lidos pelas pessoas. Para separar comandos sucessivos, um ‘carriage return’ e um ‘line feed’ encerram cada um deles. Cada comando é constituído de quatro caracteres ASCII maiúsculos, alguns com argumentos opcionais. Alguns dos comandos mais comuns são descritos a seguir:

- **USER username:** usado para enviar identificação do usuário ao servidor.
- **PASS password:** usado para enviar a senha do usuário ao servidor.
- **LIST:** usado para pedir ao servidor que envie uma lista com todos os arquivos existentes no atual diretório remoto. A lista de arquivos é enviada por meio de uma conexão de dados (nova e não persistente), e não pela conexão TCP de controle.
- **RETR filename:** usado para extrair (isto é, obter) um arquivo do diretório atual do hospedeiro remoto. Ativa o hospedeiro remoto para que abra uma conexão de dados e envie o arquivo requisitado por essa conexão.
- **STOR filename:** usado para armazenar (isto é, inserir) um arquivo no diretório atual do hospedeiro remoto.

Há, particularmente, uma correspondência unívoca entre o comando que o usuário gera e o comando FTP enviado pela conexão de controle. Cada comando é seguido de uma resposta, que é enviada do servidor ao cliente. As respostas são números de três dígitos com uma mensagem opcional após o número. Elas se assemelham, em estrutura, à codificação de estado e à frase da linha de estado da mensagem de resposta HTTP. Os inventores do HTTP incluíram intencionalmente essa similaridade nas mensagens de resposta HTTP. Algumas respostas típicas, junto com suas possíveis mensagens, são as seguintes:

- 331 Nome de usuário OK, senha requisitada
- 125 Conexão de dados já aberta; iniciando transferência
- 425 Não é possível abrir a conexão de dados
- 452 Erro ao escrever o arquivo

Para saber mais sobre outros comandos e respostas FTP, o leitor interessado pode consultar o RFC 959.

2.4 Correio eletrônico na Internet

O correio eletrônico existe desde o início da Internet. Era uma das aplicações mais populares quando ela ainda estava na infância [Segaller, 1998], e ficou mais e mais elaborado e poderoso ao longo dos anos. É uma das aplicações mais importantes e de maior uso da Internet.

Tal como o correio normal, o e-mail é um meio de comunicação assíncrono — as pessoas enviam e recebem mensagens quando for conveniente para elas, sem ter de estar coordenadas com o horário das outras pessoas. Ao contrário do correio normal, que anda a passos lentos, o correio eletrônico é rápido, fácil de distribuir e barato. O correio eletrônico moderno tem muitas características poderosas. Utilizando listas de mala direta, é possível enviar mensagens de e-mail, desejadas e indesejadas, a milhares de destinatários ao mesmo tempo. As mensagens do correio eletrônico moderno muitas vezes incluem anexos, hiperlinks, textos formatados em HTML e fotos.

Nesta seção, examinaremos os protocolos de camada de aplicação que estão no coração do correio eletrônico da Internet. Mas, antes de entrarmos nessa discussão, vamos tomar uma visão geral do sistema de correio da Internet e de seus componentes principais.

A Figura 2.16 apresenta uma visão do sistema de correio da Internet utilizando uma analogia com a correspondência por correio. Vemos, por esse diagrama, que há três componentes principais: **agentes de usuário**, **servidores de correio** e o **SMTP**. Descreveremos agora cada um desses componentes partindo do seguinte contexto: um remetente, Alice, está enviando uma mensagem de e-mail para um destinatário, Bob. Agentes de usuário permitem que usuários leiam, respondam, retransmitam, salvem e componham mensagens. (Às vezes, agentes de usuário de correio eletrônico são denominados *leitores de correio*, mas, neste livro, evitaremos essa expressão.) Quando Alice termina de compor sua mensagem, seu agente de usuário a envia a seu servidor de correio, onde ela é colocada na fila de saída de mensagens desse servidor. Quando Bob quer ler uma mensagem, seu agente de usuário a extrai da caixa de correio em seu servidor de correio. No final da década de 1990, agentes de usuário com interfaces gráficas de usuário (GUI) se tornaram populares, pois permitiam que usuários vissem e compusessem mensagens multimídia. Atualmente, o Outlook da Microsoft, o Apple Mail e o Mozilla Thunderbird são alguns dos agentes de usuário com interface gráfica populares para e-mail. Também há muitos agentes de usuário de texto de domínio público, (entre eles ‘mail’, ‘pine’ e ‘elm’), assim como interfaces baseadas na Web, como veremos rapidamente.

Servidores de correio formam o núcleo da infraestrutura do e-mail. Cada destinatário, como Bob, tem uma **caixa postal** localizada em um dos servidores de correio. A de Bob administra e guarda as mensagens que foram enviadas a ele. Uma mensagem típica inicia sua jornada no agente de usuário do remetente, vai até o servidor de correio dele e viaja até o servidor de correio do destinatário, onde é depositada na caixa postal. Quando Bob quer acessar as mensagens de sua caixa postal, o servidor de correio que contém sua caixa postal o autentica (com nome de usuário e senha). O servidor de correio de Alice também deve cuidar das falhas no servidor de

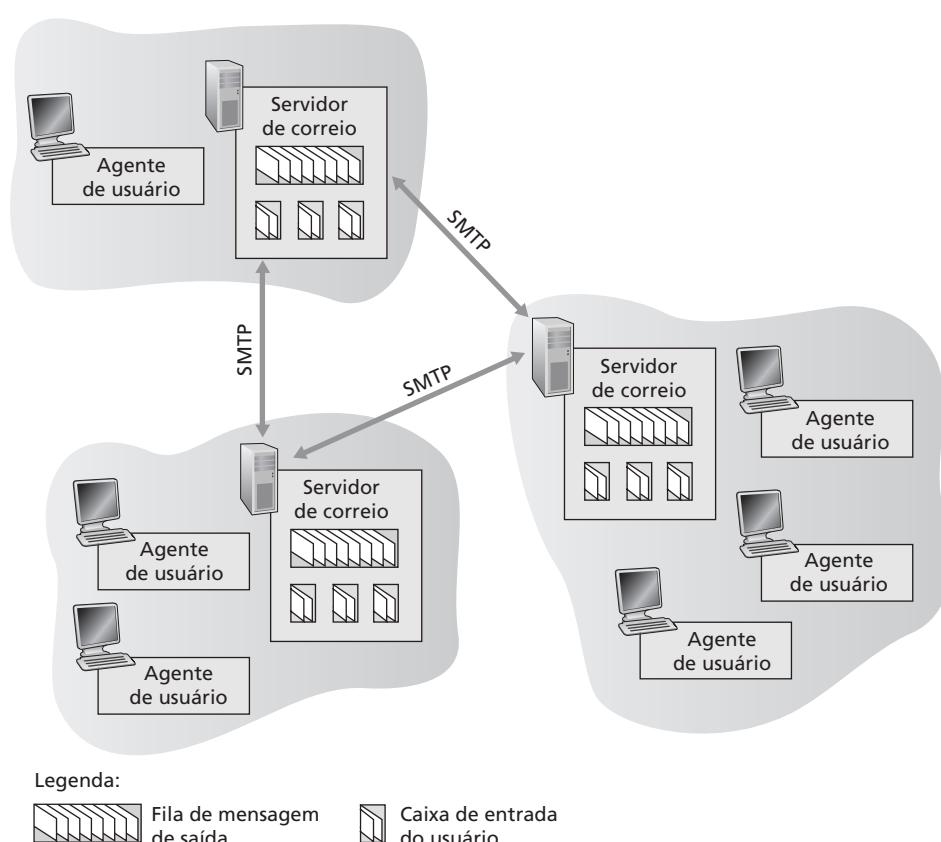


Figura 2.16 Uma visão do sistema de e-mail da Internet em analogia com a correspondência por correio



História

E-mail da Web

Em dezembro de 1995, após alguns anos depois da “invenção” da Web, Sabeer Bhatia e Jack Smith fizeram uma visita a Draper Fisher Jurvetson, um investidor em empreendimentos de Internet, e propuseram desenvolver um sistema de e-mail de livre acesso baseado na Web. A ideia era oferecer uma conta de e-mail grátis a quem quisesse e tornar essas contas acessíveis pela Web. Em troca de 15 por cento da empresa, Draper Fisher Jurvetson financiou Bhatia e Smith, que formaram uma empresa denominada Hotmail. Com três funcionários em tempo integral e mais 12 a 14 em tempo parcial, que trabalhavam em troca de opções de compra de ações da empresa, eles conseguiram desenvolver e lançar o serviço em julho de 1996. Um mês após o lançamento, a Hotmail tinha cem mil assinantes. Em dezembro de 1997, menos de 18 meses após seu lançamento, a Hotmail, já com mais de 12 milhões de assinantes, foi adquirida pela Microsoft, ao que se saiba, por 400 milhões de dólares.

O sucesso da Hotmail é muitas vezes atribuído à vantagem de ela ter sido a primeira a entrar no mercado e ao inerente ‘marketing viral’ do e-mail. (Talvez alguns dos estudantes que estão lendo este livro estarão entre os novos empreendedores que conceberão e desenvolverão serviços de Internet pioneiros no mercado e com marketing viral).

O e-mail da Web continua a prosperar, tornando-se, a cada ano, mais sofisticado e potente. Um dos serviços mais populares de hoje é o gmail do Google, que oferece livre armazenagem de gigabytes, filtro de spam e detector de vírus avançados, codificação de email opcional (utilizando SSL), coletores de e-mails e uma interface orientada a busca.

correio de Bob. Se o servidor de correio dela não puder entregar a correspondência ao servidor dele, manterá a mensagem em uma **fila de mensagens** e tentará transferi-la mais tarde. Em geral, novas tentativas serão feitas a cada 30 minutos aproximadamente; se não obtiver sucesso após alguns dias, o servidor removerá a mensagem e notificará o remetente por meio de uma mensagem de correio. O SMTP é o principal protocolo de camada de aplicação do correio eletrônico da Internet. Usa o serviço confiável de transferência de dados do TCP para transferir mensagens do servidor de correio do remetente para o do destinatário. Como acontece com a maioria dos protocolos de camada de aplicação, o SMTP tem dois lados: um lado cliente, que funciona no servidor de correio do remetente, e um lado servidor, que funciona no servidor de correio do destinatário. Ambos, o lado cliente e o lado servidor do SMTP, funcionam em todos os servidores de correio. Quando um servidor de correio envia correspondência para outros, age como um cliente SMTP. Quando o servidor de correio recebe correspondência de outros, age como um servidor SMTP.

2.4.1 SMTP

O SMTP, definido no RFC 5321, está no coração do correio eletrônico da Internet. Como mencionamos antes, esse protocolo transfere mensagens de servidores de correio remetentes para servidores de correio destinatários. O SMTP é muito mais antigo que o HTTP. (O RFC original do SMTP data de 1982, e ele já existia muito antes disso.) Embora tenha inúmeras qualidades maravilhosas, como evidencia sua ubiquidade na Internet, o SMTP é uma tecnologia antiga que possui certas características arcaicas. Por exemplo, restringe o corpo (e não apenas o cabeçalho) de todas as mensagens de correio ao simples formato ASCII de 7 bits. Essa restrição tinha sentido no começo da década de 1980, quando a capacidade de transmissão era escassa e ninguém enviava correio eletrônico com anexos volumosos nem arquivos grandes com imagens, áudio ou vídeo. Mas, hoje em dia, na era da multimídia, a restrição do ASCII de 7 bits é um tanto incômoda — exige que os dados binários de multimídia sejam codificados em ASCII antes de ser enviados pelo SMTP e que a mensagem correspondente em ASCII seja decodificada novamente para o sistema binário depois do transporte pelo SMTP. Lembre-se da Seção 2.2, na qual dissemos que o HTTP não exige que os dados de multimídia sejam codificados em ASCII antes da transferência.

Para ilustrar essa operação básica do SMTP, vamos percorrer um cenário comum. Suponha que Alice queira enviar a Bob uma simples mensagem ASCII:

1. Alice chama seu agente de usuário para e-mail, fornece o endereço de Bob (por exemplo, bob@someschool.edu), compõe uma mensagem e instrui o agente de usuário a enviar a mensagem.
2. O agente de usuário de Alice envia a mensagem para seu servidor de correio, onde ela é colocada em uma fila de mensagens.
3. O lado cliente do SMTP, que funciona no servidor de correio de Alice, vê a mensagem na fila e abre uma conexão TCP para um servidor SMTP, que funciona no servidor de correio de Bob.
4. Após alguns procedimentos iniciais de apresentação, o cliente SMTP envia a mensagem de Alice para dentro da conexão TCP.
5. No servidor de correio de Bob, o lado servidor do SMTP recebe a mensagem e a coloca na caixa postal dele.
6. Bob chama seu agente de usuário para ler a mensagem quando for mais conveniente para ele.

Esse cenário está resumido na Figura 2.17.

É importante observar que o SMTP normalmente não usa servidores de correio intermediários para enviar correspondência, mesmo quando os dois servidores estão localizados em lados opostos do mundo. Se o servidor de Alice está em Hong Kong, e o de Bob, em St. Louis, a conexão TCP é uma conexão direta entre os servidores em Hong Kong e St. Louis. Em particular, se o servidor de correio de Bob não estiver em funcionamento, a mensagem permanece no servidor de correio de Alice esperando por uma nova tentativa — a mensagem não é colocada em nenhum servidor de correio intermediário.

Vamos agora examinar mais de perto como o SMTP transfere uma mensagem de um servidor de correio remetente para um servidor de correio destinatário. Veremos que o protocolo SMTP tem muitas semelhanças com protocolos usados na interação humana cara a cara. Primeiramente, o cliente SMTP (que funciona no hospedeiro do servidor de correio remetente) faz com que o TCP estabeleça uma conexão na porta 25 com o servidor SMTP (que funciona no hospedeiro do servidor de correio destinatário). Se o servidor não estiver em funcionamento, o cliente tenta novamente mais tarde. Uma vez estabelecida a conexão, o servidor e o cliente trocam alguns procedimentos de apresentação de camada de aplicação — exatamente como os seres humanos, que frequentemente se apresentam antes de transferir informações, clientes e servidores SMTP também se apresentam antes de transferir informações. Durante essa fase, o cliente SMTP indica os endereços de e-mail do remetente (a pessoa que gerou a mensagem) e do destinatário. Assim que o cliente e o servidor SMTP terminam de se apresentar, o cliente envia a mensagem. O SMTP pode contar com o serviço confiável de transferência de dados do TCP para entregar a mensagem ao servidor sem erros. Então, o cliente repetirá esse processo, na mesma conexão TCP, se houver outras mensagens a enviar ao servidor; caso contrário, dará uma instrução ao TCP para encerrar a conexão.

Vamos examinar um exemplo de troca de mensagens entre um cliente (C) e um servidor SMTP (S). O nome do hospedeiro do cliente é crepes.fr e o nome do hospedeiro do servidor é hamburger.edu. As linhas de

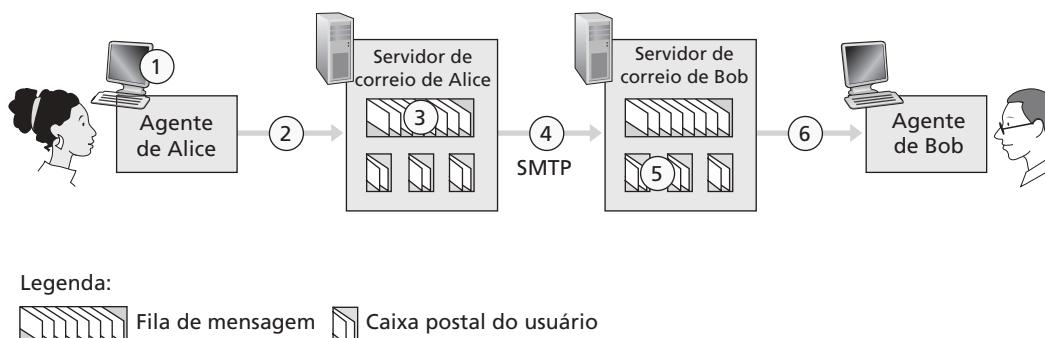


Figura 2.17 Alice envia uma mensagem a Bob

texto ASCII iniciadas com C: são exatamente as linhas que o cliente envia para dentro de seu socket TCP e as iniciadas com S: são exatamente as linhas que o servidor envia para dentro de seu socket TCP. A transcrição a seguir começa assim que a conexão TCP é estabelecida:

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu<
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with “.” on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Nesse exemplo, o cliente enviou uma mensagem (“Do you like ketchup? How about pickles?”) do servidor de correio crepes.fr ao servidor de correio hamburger.edu. Como parte do diálogo, o cliente emitiu cinco comandos: HELO (uma abreviação de HELLO), MAIL FROM, RCPT TO, DATA e QUIT. Esses comandos são autoexplicativos. O cliente também enviou uma linha consistindo em um único ponto final, que indica o final da mensagem para o servidor. (No jargão ASCII, cada mensagem termina com CRLF. CRLF, onde CR significa ‘carriage return’ e LF significa ‘line feed’). O servidor emite respostas a cada comando, e cada resposta tem uma codificação de resposta e algumas explicações (opcionais) em inglês. Mencionamos aqui que o SMTP usa conexões persistentes: se o servidor de correio remetente tiver diversas mensagens para enviar ao mesmo servidor de correio destinatário, poderá enviar todas pela mesma conexão TCP. Para cada mensagem, o cliente inicia o processo com um novo MAIL FROM: crepes.fr, indica o final da mensagem com um ponto final isolado e emite QUIT somente após todas as mensagens terem sido enviadas.

Recomendamos veementemente que você utilize o Telnet para executar um diálogo direto com um servidor SMTP. Para fazer isso digite

```
telnet serverName 25
```

em que serverName é o nome de um servidor de correio local. Ao fazer isso, você está simplesmente estabelecendo uma conexão TCP entre seu hospedeiro local e o servidor de correio. Após digitar essa linha, você deverá receber imediatamente do servidor a resposta 220. Digite, então, os comandos HELO, MAIL FROM, RCPT TO, DATA, CRLF.CRLF e QUIT nos momentos apropriados. Também recomendamos veementemente que você faça a Tarefa de Programação 2 no final deste capítulo. Nessa tarefa, você construirá um agente de usuário simples que implementa o lado cliente do SMTP. Esse agente permitirá que você envie uma mensagem de e-mail a um destinatário qualquer, por meio de um servidor de correio local.

2.4.2 Comparação com o HTTP

Agora, vamos fazer uma breve comparação entre o SMTP e o HTTP. Ambos os protocolos são usados para transferir arquivos de um hospedeiro para outro. O HTTP transfere arquivos (também denominados objetos) de um servidor Web para um cliente Web (normalmente um browser). O SMTP transfere arquivos (isto é, mensagens de e-mail) de um servidor de correio para outro. Ao transferir os arquivos, o HTTP persistente e o SMTP usam conexões persistentes. Assim, os dois protocolos têm características em comum. Existem, todavia, diferenças importantes. A primeira é que o HTTP é, principalmente, um **protocolo de recuperação de informações (pull)**

protocol) — alguém carrega informações em um servidor Web e os usuários utilizam o HTTP para recuperá-las do servidor quando quiserem. Em particular, a conexão TCP é ativada pela máquina que quer receber o arquivo. O SMTP, por sua vez, é, primordialmente, um **protocolo de envio de informações** (*push protocol*) — o servidor de correio remetente envia o arquivo para o servidor de correio destinatário. Em particular, a conexão TCP é ativada pela máquina que quer enviar o arquivo.

A segunda diferença, à qual aludimos anteriormente, é que o SMTP exige que cada mensagem, inclusive o corpo, esteja no formato ASCII de 7 bits. Se a mensagem contiver caracteres que não estejam nesse formato (por exemplo, caracteres em francês, com acento) ou dados binários (como um arquivo de imagem), terá de ser codificada em ASCII de 7 bits. Dados HTTP não impõem esta restrição.

A terceira diferença importante refere-se ao modo como um documento que contém texto e imagem (juntamente com outros tipos possíveis de mídia) é manipulado. Como vimos na Seção 2.2, o HTTP encapsula cada objeto em sua própria mensagem HTTP. O correio pela Internet, como discutiremos com maiores detalhes mais adiante, coloca todos os objetos de mensagem em um única mensagem.

2.4.3 Formatos de mensagem de correio e MIME

Quando Alice escreve uma carta a Bob e a envia pelo correio normal, ela pode incluir todos os tipos de informações periféricas no cabeçalho da carta, como seu próprio endereço, o endereço de Bob e a data. De modo semelhante, quando uma mensagem de e-mail é enviada, um cabeçalho contendo informações periféricas antecede o corpo da mensagem em si. Essas informações periféricas estão contidas em uma série de linhas de cabeçalho definidas no RFC 5322. As linhas de cabeçalho e o corpo da mensagem são separados por uma linha em branco (isto é, por CRLF). O RFC 5322 especifica o formato exato das linhas de cabeçalho das mensagens, bem como suas interpretações semânticas. Como acontece com o HTTP, cada linha de cabeçalho contém um texto legível, consistindo em uma palavra-chave seguida de dois-pontos e de um valor. Algumas palavras-chave são obrigatórias e outras, opcionais. Cada cabeçalho deve ter uma linha de cabeçalho *From:* e uma *To:* e pode incluir também uma *Subject:* bem como outras opcionais. É importante notar que essas linhas de cabeçalho são *diferentes* dos comandos SMTP que estudamos na Seção 2.4.1 (ainda que contenham algumas palavras em comum, como ‘from’ e ‘to’). Os comandos daquela seção faziam parte do protocolo de apresentação SMTP; as linhas de cabeçalho examinadas nesta seção fazem parte da própria mensagem de correio.

Um cabeçalho de mensagem típico é semelhante a:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

Após o cabeçalho da mensagem, vem uma linha em branco e, em seguida, o corpo da mensagem (em ASCII). Você pode usar o Telnet para enviar a um servidor de correio uma mensagem que contenha algumas linhas de cabeçalho, inclusive *Subject*. Para tal, utilize o comando `telnet serverName 25`, como discutido na Seção 2.4.1.

2.4.4 Protocolos de acesso ao correio

Quando o SMTP entrega a mensagem do servidor de correio de Alice ao servidor de correio de Bob, ela é colocada na caixa postal de Bob. Durante toda essa discussão, ficou tacitamente subentendido que Bob lê sua correspondência ao entrar no hospedeiro servidor e, em seguida, executa o leitor de correio que roda naquela máquina. Até o início da década de 1990, este era o modo padronizado de acessar o correio. Mas hoje o acesso ao correio usa uma arquitetura cliente-servidor — o usuário típico lê e-mails com um cliente que funciona em seu sistema final, por exemplo, um PC no escritório, um laptop ou um PDA. Quando executam um cliente de correio em PC local, usuários desfrutam de uma série de propriedades, inclusive a capacidade de ver mensagens e anexos multimídia.

Dado que Bob (o destinatário) executa seu agente de usuário em seu PC local, é natural que ele considere a instalação de um servidor de correio nessa máquina. Adotando essa abordagem, o servidor de correio de Alice

dialogaria diretamente com o PC de Bob. Porém, há um problema com essa abordagem. Lembre-se de que um servidor de correio gerencia caixas postais e executa os lados cliente e servidor do SMTP. Se o servidor de correio de Bob residisse em seu PC local, este teria de ficar sempre em funcionamento e ligado na Internet para poder receber novas correspondências que poderiam chegar a qualquer hora, o que não é prático para muitos usuários da Internet. Em vez disso, um usuário típico executa um agente de usuário no PC local, mas acessa sua caixa postal armazenada em um servidor de correio compartilhado que está sempre em funcionamento. Esse servidor de correio é compartilhado com outros usuários e, em geral, é mantido pelo ISP do usuário (por exemplo, universidade ou empresa).

Agora, vamos considerar o caminho que uma mensagem percorre quando é enviada de Alice para Bob. Acabamos de aprender que, em algum ponto ao longo do caminho, a mensagem de e-mail precisa ser depositada no servidor de correio de Bob. Essa tarefa poderia ser realizada simplesmente fazendo com que o agente de usuário de Alice enviasse a mensagem diretamente ao servidor de correio de Bob, o que pode ser feito com o SMTP — realmente, o SMTP foi projetado para enviar e-mail de um hospedeiro para outro. Contudo, normalmente o agente de usuário do remetente não dialoga diretamente com o servidor de correio do destinatário. Em vez disso, como mostra a Figura 2.18, o agente de usuário de Alice usa SMTP para enviar a mensagem de e-mail a seu servidor de correio. Em seguida, esse servidor usa SMTP (como um cliente SMTP) para retransmitir a mensagem de e-mail para o servidor de correio de Bob. Por que esse procedimento em duas etapas? Primordialmente porque, sem a retransmissão através do servidor de correio de Alice, o agente de usuário dela não dispõe de nenhum recurso para um servidor de correio de destinatário que não pode ser alcançado. Fazendo com que Alice primeiramente deposite o e-mail em seu próprio servidor de correio, este pode tentar, várias vezes, enviar a mensagem ao servidor de correio de Bob, digamos, a cada 30 minutos, até que esse servidor entre em operação. (E, se o servidor de correio de Alice não estiver funcionando, ela terá o recurso de se queixar ao administrador do seu sistema!) O RFC do SMTP define como os comandos do SMTP podem ser usados para retransmitir uma mensagem por vários servidores SMTP.

Mas ainda falta uma peça do quebra-cabeça! De que forma um destinatário como Bob, que executa um agente de usuário em seu PC local, obtém suas mensagens que estão em um servidor de correio dentro do seu ISP? Note que o agente de usuário de Bob não pode usar SMTP para obter as mensagens porque essa operação é de recuperação (*pull*), e o SMTP é um protocolo de envio (*push*). O quebra-cabeça é concluído com a introdução de um protocolo especial de acesso ao correio que transfere mensagens do servidor de correio de Bob para seu PC local. Atualmente, há vários protocolos populares de acesso a correio, entre eles **POP3** (Post Office Protocol versão 3), **IMAP** (Internet Mail Access Protocol) e **HTTP**.

A Figura 2.18 apresenta um resumo dos protocolos usados no correio da Internet: o SMTP é utilizado para transferir correspondência do servidor de correio remetente para o servidor de correio destinatário; também é usado para transferir correspondência do agente de usuário remetente para o servidor de correio remetente. Um protocolo de acesso de correio, como o POP3, é usado para transferir correspondência do servidor de correio destinatário para o agente de usuário destinatário.

POP3

O POP3 é um protocolo de acesso de correio extremamente simples. É definido no RFC 1939, que é curto e bem fácil de ler. Por esse protocolo ser tão simples, sua funcionalidade é bastante limitada. O POP3 começa

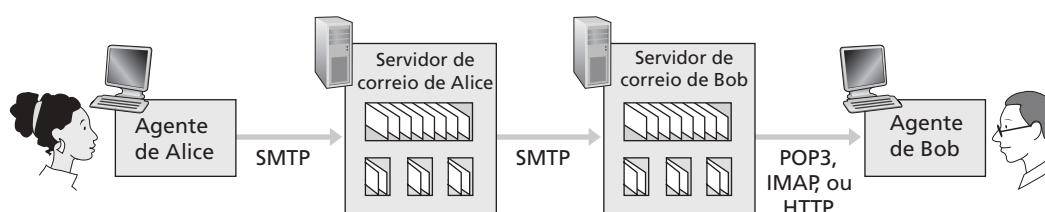


Figura 2.18 Protocolos de e-mail e suas entidades comunicantes

quando o agente de usuário (o cliente) abre uma conexão TCP com o servidor de correio (o servidor) na porta 110. Com a conexão TCP ativada, o protocolo passa por três fases: autorização, transação e atualização. Durante a primeira fase, autorização, o agente de usuário envia um nome de usuário e uma senha (às claras) para autenticar o usuário. Na segunda fase, transação, recupera mensagens; é também nessa fase que o agente de usuário pode marcar mensagens que devem ser apagadas, remover essas marcas e obter estatísticas de correio. A terceira fase, atualização, ocorre após o cliente ter dado o comando `quit` que encerra a sessão POP3. Nesse momento, o servidor de correio apaga as mensagens que foram marcadas.

Em uma transação POP3, o agente de usuário emite comandos e o servidor, uma resposta para cada um deles. Há duas respostas possíveis: `+OK` (às vezes seguida de dados do servidor para o cliente), usada pelo servidor para indicar que correu tudo bem com o comando anterior e `-ERR`, que o servidor usa para informar que houve algo errado com o comando anterior.

A fase de autorização tem dois comandos principais: `user <user name>` e `pass <password>`. Para ilustrar esses dois comandos, sugerimos que você realize uma sessão Telnet diretamente com um servidor POP3, usando a porta 110, e emita os dois comandos. Suponha que `mailServer` seja o nome de seu servidor de correio. O que você verá será algo parecido com:

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

Se você escrever um comando errado, o POP3 responderá com uma mensagem `-ERR`.

Agora, vamos examinar a fase de transação. Um agente de usuário que utiliza POP3 frequentemente pode ser configurado (pelo usuário) para ‘ler-e-apagar’ ou para ‘ler-e-guardar’. A sequência de comandos emitida por um agente de usuário POP3 depende do modo em que o agente de usuário estiver operando. No modo ler-e-apagar, o agente de usuário emite os comandos `list`, `retr` e `dele`. Como exemplo, suponha que o usuário tenha duas mensagens em sua caixa postal. No diálogo abaixo, **C:** (que representa o cliente) é o agente de usuário e **S:** (que representa o servidor), o servidor de correio. A transação será mais ou menos assim:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (blah blah ...
S: .....
S: .....blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: .....
S: .....blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

O agente de usuário primeiramente pede ao servidor de correio que apresente o tamanho de cada uma das mensagens armazenadas. Então, ele recupera e apaga cada mensagem do servidor. Note que, após a fase de

autorização, o agente de usuário empregou apenas quatro comandos: `list`, `retr`, `dele` e `quit`. A sintaxe para esses comandos é definida no RFC 1939. Depois de processar o comando de saída (`quit`), o servidor POP3 entra na fase de atualização e remove as mensagens 1 e 2 da caixa postal.

Há um problema com o modo ler-e-apagar: o destinatário, Bob, pode ser nômade e querer acessar seu correio de muitas máquinas, por exemplo, do PC de seu escritório, do PC de sua casa e de seu computador portátil. O modo ler-e-apagar reparte as mensagens de correio de Bob entre essas três máquinas; em particular, se ele ler primeiramente uma mensagem no PC de seu escritório, não poderá lê-la novamente mais tarde em seu computador portátil. No modo ler-e-guardar, o agente de usuário deixa as mensagens no servidor de correio após descarregá-las. Nesse caso, Bob pode reler mensagens em máquinas diferentes; pode acessar uma mensagem em seu local de trabalho e, uma semana depois, acessá-la novamente em casa.

Durante uma sessão POP3 entre um agente de usuário e o servidor de correio, o servidor POP3 mantém alguma informação de estado; em particular, monitora as mensagens do usuário marcadas para apagar. Contudo, não mantém informação de estado entre sessões POP3. Essa falta de informação simplifica a implementação de um servidor POP3.

IMAP

Usando POP3, assim que baixar suas mensagens na máquina local, Bob pode criar pastas de correspondência e transferir as mensagens baixadas para elas. Em seguida, pode apagar as mensagens, mudá-las de pastas e procurar mensagens (por nome de remetente ou assunto). Mas esse paradigma — pastas e mensagens na máquina local — apresenta um problema para o usuário nômade que gostaria de manter uma hierarquia de pastas em um servidor remoto que possa ser acessado de qualquer computador: com o POP3, isso não é possível. Esse protocolo não provê nenhum meio para um usuário criar pastas remotas e designar mensagens a pastas.

Para resolver esse e outros problemas, foi inventado o protocolo IMAP, definido no RFC 3501. Como o POP3, o IMAP é um protocolo de acesso a correio, porém com mais recursos, mas é também significativamente mais complexo. (E, portanto, também as implementações dos lados cliente e servidor são significativamente mais complexas.)

Um servidor IMAP associa cada mensagem a uma pasta. Quando uma mensagem chega a um servidor pela primeira vez, é associada com a pasta INBOX do destinatário, que, então, pode transferir a mensagem para uma nova pasta criada por ele, lê-la, apagá-la e assim por diante. O protocolo IMAP provê comandos que permitem que os usuários criem pastas e transfiram mensagens de uma para outra. O protocolo também provê comandos que os usuários podem usar para pesquisar pastas remotas em busca de mensagens que obedeçam a critérios específicos. Note que, diferentemente do POP3, um servidor IMAP mantém informação de estado de usuário entre sessões IMAP — por exemplo, os nomes das pastas e quais mensagens estão associadas a elas.

Outra característica importante do IMAP é que ele tem comandos que permitem que um agente de usuário obtenha componentes de mensagens. Por exemplo, um agente de usuário pode obter apenas o cabeçalho ou somente uma das partes de uma mensagem MIME multiparte. Essa característica é útil quando há uma conexão de largura de banda estreita entre o agente de usuário e seu servidor de correio, como, por exemplo, uma conexão de modem sem fio ou de baixa velocidade. Com uma conexão de pequena largura de banda, o usuário pode decidir não baixar todas as mensagens de sua caixa postal, evitando, em particular, mensagens longas que possam conter, por exemplo, um clipe de áudio ou de vídeo. Se quiser ler tudo sobre o IMAP, consulte o site oficial [IMAP, 2009].

E-mail pela Web

Hoje, um número cada vez maior de usuários está enviando e acessando e-mails por meio de seus browsers Web. O Hotmail lançou o e-mail com acesso pela Web em meados da década de 1990; agora, esse tipo de acesso é provido por praticamente todos os sites ISP, bem como universidades e empresas importantes. Com esse serviço, o agente de usuário é um browser Web comum e o usuário se comunica com sua caixa postal remota via HTTP. Quando um destinatário, por exemplo, Bob, quer acessar uma mensagem em sua caixa postal, ela é enviada do servidor de correio para o browser de Bob usando o protocolo HTTP, e não os protocolos POP3 ou IMAP. Quando

um remetente, por exemplo, Alice, quer enviar uma mensagem de e-mail, esta é enviada do browser de Alice para seu servidor de correio por HTTP, e não por SMTP. O servidor de correio de Alice, contudo, ainda envia mensagens para outros servidores de correio e recebe mensagens de outros servidores de correio usando o SMTP.

2.5 DNS: o serviço de diretório da Internet

Nós, seres humanos, podemos ser identificados por diversas maneiras. Por exemplo, podemos ser identificados pelo nome que aparece em nossa certidão de nascimento, pelo número do RG ou da carteira de motorista. Embora cada um desses números possa ser usado para identificar pessoas, em um dado contexto um pode ser mais adequado que outro. Por exemplo, os computadores da Receita Federal preferem usar o número do CPF (de tamanho fixo) ao nome que consta em nossa certidão de nascimento. Por outro lado, pessoas comuns preferem nosso nome de batismo, mais fácil de lembrar, ao número do CPF. (Realmente, você pode se imaginar dizendo: “Oi, meu nome é 132-67-9875. Este é meu marido, 178-87-1146”?)

Assim como seres humanos podem ser identificados de muitas maneiras, exatamente o mesmo acontece com hospedeiros da Internet. Um identificador é seu nome de **hospedeiro** (*hostname*). Nomes de hospedeiro — como `cnn.com`, `www.yahoo.com`, `gaias.cs.umass.edu` e `cis.poly.edu` — são fáceis de lembrar e, portanto, apreciados pelos seres humanos. Todavia, eles fornecem pouca, se é que alguma, informação sobre a localização de um hospedeiro na Internet. (Um nome como `www.eurecom.fr`, que termina com o código do país `.fr`, nos informa que o hospedeiro provavelmente está na França, mas não diz muito mais do que isso.) Além disso, como nomes de hospedeiros podem consistir em caracteres alfanuméricos de comprimento variável, seriam difíceis de ser processados por roteadores. Por essas razões, hospedeiros também são identificados pelo que denominamos **endereços IP**. Discutiremos endereços IP mais detalhadamente no Capítulo 4, mas é importante falar um pouco sobre eles agora. Um endereço IP é constituído de 4 bytes e sua estrutura hierárquica é rígida. Ele é semelhante a `121.7.106.83`, no qual cada ponto separa um dos bytes expressos em notação decimal de 0 a 255. Um endereço IP é hierárquico porque, ao examiná-lo da esquerda para a direita, obtemos gradativamente mais informações específicas sobre onde o hospedeiro está localizado na Internet (isto é, em qual rede, dentre as muitas que compõem a Internet). De maneira semelhante, quando examinamos um endereço postal de cima para baixo, obtemos informações cada vez mais específicas sobre a localização do destinatário.

2.5.1 Serviços fornecidos pelo DNS

Acabamos de ver que há duas maneiras de identificar um hospedeiro — por um nome de hospedeiro e por um endereço IP. As pessoas preferem o identificador nome de hospedeiro por ser mais fácil de lembrar, ao passo que roteadores preferem endereços IP de comprimento fixo e estruturados hierarquicamente. Para conciliar essas preferências, é necessário um serviço de diretório que traduza nomes de hospedeiro para endereços IP. Esta é a tarefa principal do DNS (*domain name system* — **sistema de nomes de domínio**) da Internet. O DNS é (1) um banco de dados distribuído implementado em uma hierarquia de servidores de nome (**servidores DNS**), e (2) um protocolo de camada de aplicação que permite que hospedeiros consultem o banco de dados distribuído. Os servidores de nome são frequentemente máquinas UNIX que executam o software BIND (Berkeley Internet Name Domain) [BIND, 2009]. O protocolo DNS utiliza UDP e usa a porta 53.

O DNS é comumente empregado por outras entidades da camada de aplicação — inclusive HTTP, SMTP e FTP — para traduzir nomes de hospedeiros fornecidos por usuários para endereços IP. Como exemplo, considere o que acontece quando um browser (isto é, um cliente HTTP), que executa na máquina de algum usuário, requisita o URL `www.someschool.edu/index.html`. Para que a máquina do usuário possa enviar uma mensagem de requisição HTTP ao servidor Web `www.someschool.edu`, ela precisa primeiramente obter o endereço IP de `www.someschool.edu`. Isso é feito da seguinte maneira:

1. A própria máquina do usuário executa o lado cliente da aplicação DNS.
2. O browser extrai o nome de hospedeiro, `www.someschool.edu`, do URL e passa o nome para o lado cliente da aplicação DNS.

3. O cliente DNS envia uma consulta contendo o nome do hospedeiro para um servidor DNS.
4. O cliente DNS finalmente recebe uma resposta, que inclui o endereço IP correspondente ao nome de hospedeiro.
5. Tão logo o browser receba o endereço do DNS, pode abrir uma conexão TCP com o processo servidor HTTP localizado naquele endereço IP.

Vemos, por esse exemplo, que o DNS adiciona mais um atraso — às vezes substancial — às aplicações de Internet que o usam. Felizmente, como discutiremos mais adiante, o endereço IP procurado frequentemente está no cache de um servidor DNS ‘próximo’, o que ajuda a reduzir o tráfego DNS na rede, bem como o atraso médio do DNS.

O DNS provê alguns outros serviços importantes além da tradução de nomes de hospedeiro para endereços IP:

Apelidos de hospedeiro. Um hospedeiro com nome complicado pode ter um ou mais apelidos. Um nome como `relay1.west-coast.enterprise.com` pode ter, por exemplo, dois apelidos, como `enterprise.com` e `www.enterprise.com`. Nesse caso, o nome de hospedeiro `relay1.west-coast.enterprise.com` é denominado **nome canônico**. Apelidos, quando existem, são em geral mais fáceis de lembrar do que o nome canônico. O DNS pode ser chamado por uma aplicação para obter o nome canônico correspondente a um apelido fornecido, bem como para obter o endereço IP do hospedeiro.

Apelidos de servidor de correio. Por razões óbvias, é adequado que endereços de e-mail sejam fáceis de lembrar. Por exemplo, se Bob tem uma conta no Hotmail, seu endereço de e-mail pode ser simplesmente `bob@hotmail.com`. Contudo, o nome de hospedeiro do servidor do Hotmail é mais complicado e muito mais difícil de lembrar do que simplesmente `hotmail.com` (por exemplo, o nome canônico pode ser algo parecido com `relay1.west-coast.hotmail.com`). O DNS pode ser chamado por uma aplicação de correio para obter o nome canônico a partir de um apelido fornecido, bem como o endereço IP do hospedeiro. Na verdade, o registro MX (veja adiante) permite que o servidor de correio e o servidor Web de uma empresa tenham nomes (apelidos) idênticos; por exemplo, o servidor Web e o servidor de correio de uma empresa podem ambos ser denominados `enterprise.com`.

Distribuição de carga. O DNS também é usado para realizar distribuição de carga entre servidores replicados, tais como os servidores Web replicados. Sites movimentados como `cnn.com` são replicados em vários servidores, sendo que cada servidor roda em um sistema final diferente e tem um endereço IP diferente. Assim, no caso de servidores Web replicados, um *conjunto* de endereços IP fica associado a um único nome canônico e contido no banco de dados do DNS. Quando clientes consultam um nome mapeado para um conjunto de endereços, o DNS responde com o conjunto inteiro de endereços IP, mas faz um rodízio da ordem dos endereços dentro de cada resposta. Como um cliente normalmente envia sua mensagem de requisição HTTP ao endereço IP que ocupa o primeiro lugar no conjunto, o rodízio



Princípios na prática

DNS: funções decisivas de rede via paradigma cliente-servidor

Como os protocolos HTTP, FTP e SMTP, o DNS é um protocolo da camada de aplicação, já que (1) roda entre sistemas finais comunicantes usando o paradigma cliente-servidor e (2) depende de um protocolo de transporte fim a fim subjacente para transferir mensagens DNS entre sistemas finais comunicantes. Em outro sentido, contudo, o papel do DNS é bastante diferente das aplicações Web, da transferência de arquivo e do e-mail. Diferentemente dessas aplicações, o DNS não é uma aplicação com a qual o usuário interage diretamente. Em vez disso, fornece uma função interna da Internet — a saber, a tradução de nomes de hospedeiro para seus endereços IP subjacentes, para aplicações de usuário e outros softwares da Internet. Notamos, na Seção 1.2, que muito da complexidade da arquitetura da Internet está localizada na ‘periferia’ da rede. O DNS, que implementa o processo crucial de tradução de nome para endereço usando clientes e servidores localizados nas bordas da rede, é mais um exemplo dessa filosofia de projeto.

de DNS distribui o tráfego entre os servidores replicados. O rodízio de DNS também é usado para e-mail, de modo que vários servidores de correio podem ter o mesmo apelido. Recentemente, empresas distribuidoras de conteúdo como a Akamai [Akamai, 2009] passaram a usar o DNS de maneira mais sofisticada para prover distribuição de conteúdo na Web (veja Capítulo 7).

O DNS está especificado no RFC 1034 e no RFC 1035 e atualizado em diversos RFCs adicionais. É um sistema complexo e, neste livro, apenas mencionamos os aspectos fundamentais de sua operação. O leitor interessado pode consultar os RFCs citados, o livro escrito por Abitz e Liu [Abitz, 1993] e também o artigo de [Mockapetris, 1988], que apresenta uma retrospectiva e uma ótima descrição do que e do porquê do DNS, e [Mockapetris, 2005].

2.5.2 Visão geral do modo de funcionamento do DNS

Apresentaremos, agora, uma visão panorâmica do modo de funcionamento do DNS. Nossa discussão focalizará o serviço de tradução de nome de hospedeiro para endereço IP.

Suponha que uma certa aplicação (como um browser Web ou um leitor de correio), que executa na máquina de um usuário, precise traduzir um nome de hospedeiro para um endereço IP. A aplicação chamará o lado cliente do DNS, especificando o nome de hospedeiro que precisa ser traduzido. (Em muitas máquinas UNIX, `gethostbyname()` é a chamada de função que uma aplicação invoca para realizar a tradução. Na Seção 2.7, mostraremos como uma aplicação Java pode chamar o DNS). A partir daí, o DNS do hospedeiro do usuário assume o controle, enviando uma mensagem de consulta para dentro da rede. Todas as mensagens de consulta e de resposta do DNS são enviadas dentro de datagramas UDP à porta 53. Após um atraso na faixa de milissegundos a segundos, o DNS no hospedeiro do usuário recebe uma mensagem de resposta DNS fornecendo o mapeamento desejado, que é, então, passado para a aplicação que está interessada. Portanto, do ponto de vista dessa aplicação, que está na máquina do cliente, o DNS é uma caixa-preta que provê um serviço de tradução simples e direto. Mas, na realidade, a caixa-preta que implementa o serviço é complexa, consistindo em um grande número de servidores de nomes distribuídos ao redor do mundo, bem como em um protocolo de camada de aplicação que especifica como se comunicam os servidores de nomes e os hospedeiros que fazem a consulta.

Um arranjo simples para DNS seria ter um servidor de nomes contendo todos os mapeamentos. Nesse projeto centralizado, os clientes simplesmente dirigiriam todas as consultas a esse único servidor de nomes, que responderia diretamente aos clientes que estão fazendo a consulta. Embora a simplicidade desse arranjo seja atraente, ele não é adequado para a Internet de hoje com seu vasto (e crescente) número de hospedeiros. Dentre os problemas de um arranjo centralizado, estão:

- **Um único ponto de falha.** Se o servidor de nomes quebrar, a Internet inteira quebrará!
- **Volume de tráfego.** Um único servidor de nomes teria de manipular todas as consultas DNS (para todas as requisições HTTP e mensagens de e-mail geradas por centenas de milhões de hospedeiros).
- **Banco de dados centralizado distante.** Um único servidor de nomes nunca poderia estar ‘próximo’ de todos os clientes que fazem consultas. Se colocarmos o único servidor de nomes na cidade de Nova York, todas as buscas provenientes da Austrália terão de viajar até o outro lado do globo, talvez por linhas lentas e congestionadas, o que pode resultar em atrasos significativos.
- **Manutenção.** O único servidor de nomes teria de manter registros de todos os hospedeiros da Internet. Esse banco de dados não somente seria enorme, mas também teria de ser atualizado frequentemente para atender a todos os novos hospedeiros.

Em resumo, um banco de dados centralizado em um único servidor DNS simplesmente não é escalável. Consequentemente, o DNS é distribuído por conceito de projeto. Na verdade, ele é um ótimo exemplo de como um banco de dados distribuído pode ser implementado na Internet.

Um banco de dados distribuído, hierárquico

Para tratar da questão da escala, o DNS usa um grande número de servidores, organizados de maneira hierárquica e distribuídos por todo o mundo. Nenhum servidor de nomes isolado tem todos os mapeamentos para

todos os hospedeiros da Internet. Em vez disso, os mapeamentos são distribuídos pelos servidores de nomes. Como uma primeira aproximação, há três classes de servidores de nomes: servidores de nomes raiz, servidores DNS de domínio de alto nível (*top-level domain* — TLD) e servidores DNS com autoridade — organizados em uma hierarquia, como mostra a Figura 2.19.

Para entender como essas três classes de servidores interagem, suponha que um cliente DNS queira determinar o endereço IP para o nome de hospedeiro `www.amazon.com`. Como uma primeira aproximação, ocorrerão os seguintes eventos. Em primeiro lugar, o cliente contatará um dos servidores raiz, que retornará endereços IP dos servidores TLD para o domínio de alto nível `.com`. Então, o cliente contatará um desses servidores TLD, que retornará o endereço IP de um servidor com autoridade para `amazon.com`. Finalmente, o cliente contatará um dos servidores com autoridade para `amazon.com`, que retornará o endereço IP para o nome de hospedeiro `www.amazon.com`. Mais adiante, analisaremos mais detalhadamente esse processo de consulta DNS. Mas, em primeiro lugar, vamos examinar mais de perto as três classes de servidores DNS.

Servidores de nomes raiz. Na Internet há 13 servidores de nomes raiz (denominados de A a M) e a maior parte deles está localizada na América do Norte. Um mapa de servidores de nomes raiz de outubro de 2006 é mostrado na Figura 2.20; uma lista dos servidores de nomes raiz existentes hoje está disponível em [Root-servers, 2009]. Embora tenhamos nos referido a cada um dos 13 servidores de nomes raiz como se fossem um servidor único, na realidade, cada um é um conglomerado de servidores replicados, para fins de segurança e confiabilidade.

Servidores de nomes de Domínio de Alto Nível (TLD). Esses servidores são responsáveis por domínios de alto nível como `.com`, `.org`, `.net`, `.edu` e `.gov`, e por todos os domínios de alto nível de países, tais como `.uk`, `.fr`, `.ca` e `.jp`. A empresa Network Solutions mantinha os servidores TLD para o domínio de alto nível `.com` e a empresa Educause mantinha os servidores para o domínio de alto nível `.edu`.

Servidores de nomes com autoridade. Toda organização que tiver hospedeiros que possam ser acessados publicamente na Internet (como servidores Web e servidores de correio) deve fornecer registros DNS também acessíveis publicamente que mapeiem os nomes desses hospedeiros para endereços IP. Um servidor DNS com autoridade de uma organização abriga esses registros. Uma organização pode preferir implementar seu próprio servidor DNS com autoridade para abrigar esses registros ou, como alternativa, pode pagar para armazená-los em um servidor DNS com autoridade de algum provedor de serviço. A maioria das universidades e empresas de grande porte implementam e mantêm seus próprios servidores DNS primário e secundário (backup) com autoridade.

Os servidores de nomes raiz, TLD e com autoridade pertencem à hierarquia de servidores DNS, como mostra a Figura 2.19. Há um outro tipo importante de DNS, denominado **servidor DNS local**, que não pertence, estreitamente, à hierarquia de servidores, mas, mesmo assim, é central para a arquitetura DNS.

Cada ISP — como o de uma universidade, de um departamento acadêmico, de uma empresa ou de uma residência — tem um servidor de nomes local (também denominado servidor de nomes default). Quando um hospedeiro se conecta com um ISP, o ISP fornece ao hospedeiro os endereços IP de um ou mais de seus servidores

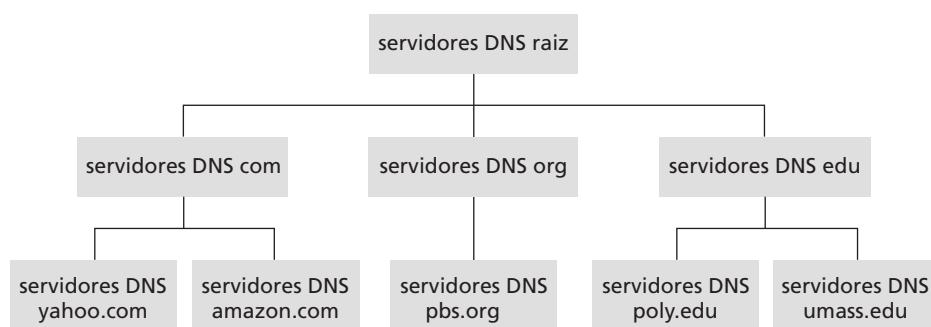


Figura 2.19 Parte da hierarquia de servidores DNSs

de nomes locais (normalmente por DHCP, que será discutido no Capítulo 4). Determinar o endereço IP do seu servidor de nomes local é fácil: basta acessar as janelas de estado da rede no Windows ou UNIX. O servidor de nomes local de um hospedeiro normalmente está ‘próximo’ dele. No caso de um ISP institucional, pode estar na mesma LAN do hospedeiro; já no caso de um ISP residencial, em geral o servidor de nomes está separado do hospedeiro por não mais do que alguns roteadores. Quando um hospedeiro faz uma consulta ao DNS, ela é enviada ao servidor de nomes local, que age como proxy e a retransmite para a hierarquia do servidor DNS, como discutiremos mais detalhadamente a seguir.

Vamos examinar um exemplo simples. Suponha que o hospedeiro `cis.poly.edu` deseje o endereço IP de `gaia.cs.umass.edu`. Suponha também que o servidor de nomes local da Polytechnic seja denominado `dns.poly.edu` e que um servidor de nomes com autoridade para `gaia.cs.umass.edu` seja denominado `dns.umass.edu`. Como mostra a Figura 2.21, o hospedeiro `cis.poly.edu` primeiramente envia uma mensagem de consulta DNS a seu servidor de nomes local `dns.poly.edu`. Essa mensagem contém o nome de hospedeiro a ser traduzido, isto é, `gaia.cs.umass.edu`. O servidor de nomes local transmite a mensagem de consulta a um servidor de nomes raiz, que percebe o sufixo `edu` e retorna ao servidor de nomes local uma lista de endereços IP contendo servidores TLD responsáveis por `edu`. Então, o servidor de nomes local retransmite a mensagem de consulta a um desses servidores TLD que, por sua vez, percebe o sufixo `umass.edu` e responde com o endereço IP do servidor de nomes com autoridade para a University of Massachusetts, a saber, `dns.umass.edu`. Finalmente, o servidor de nomes local reenvia a mensagem de consulta diretamente a `dns.umass.edu`, que responde com o endereço IP de `gaia.cs.umass.edu`. Note que, nesse exemplo, para poder obter o mapeamento para um único nome de hospedeiro, foram enviadas oito mensagens DNS: quatro mensagens de consulta e quatro mensagens de resposta! Em breve veremos como o cache de DNS reduz esse tráfego de consultas.

Nosso exemplo anterior afirmou que o servidor TLD conhece o servidor de nomes com autoridade para o nome de hospedeiro, o que nem sempre é verdade. Ele pode conhecer apenas um servidor de nomes intermediário que, por sua vez, conhece o servidor de nomes com autoridade para o nome de hospedeiro. Por exemplo, suponha novamente que a Universidade de Massachusetts tenha um servidor de nomes para a universidade denominado `dns.umass.edu`. Imagine também que cada um dos departamentos da universidade tenha seu próprio servidor de nomes e que cada servidor de nomes departamental seja um servidor de nomes com autoridade para todos os hospedeiros do departamento. Nesse caso, quando o servidor de nomes intermediário `dns.umass.edu` receber uma consulta para um hospedeiro cujo nome termina com `cs.umass.edu`, ele retornará a `dns.poly.edu` o endereço IP de `dns.cs.umass.edu`, que tem autoridade para todos os nomes de hospedeiro que terminam com `cs.umass.edu`. Então, o servidor de nomes local `dns.poly.edu` enviará a consulta ao servidor de nomes com



Figura 2.20 Servidores DNS raiz em 2009 (nome, organização, localização)

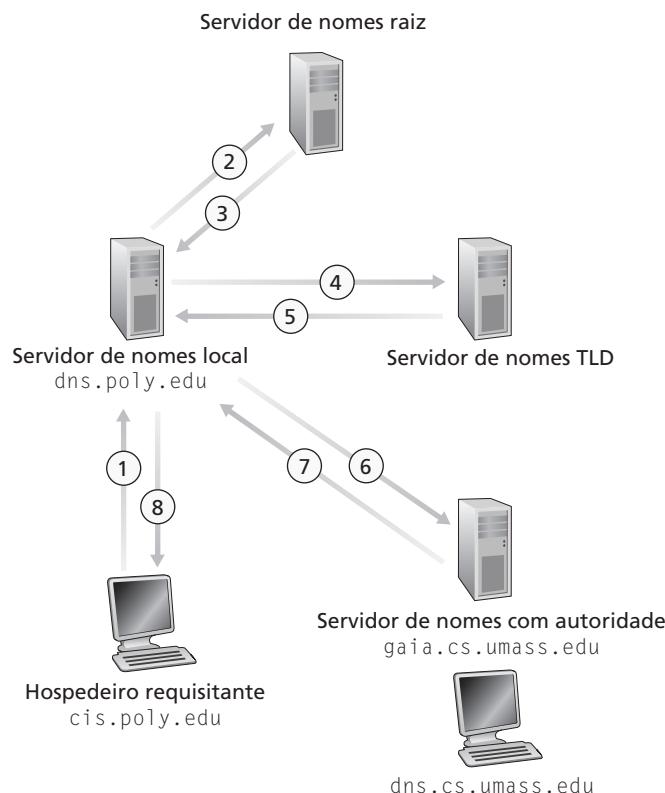


Figura 2.21 Intereração de vários servidores DNS

autoridade, que retornará o mapeamento desejado para o servidor de nomes local e que, por sua vez, o repassará ao hospedeiro requisitante. Nesse caso, serão enviadas dez mensagens DNS no total!

O exemplo mostrado na Figura 2.21 usa **consultas recursivas** e **consultas iterativas**. A consulta enviada de `cis.poly.edu` para `dns.poly.edu` é recursiva, visto que pede a `dns.poly.edu` que obtenha o mapeamento em seu nome. Mas as três consultas subsequentes são iterativas, visto que todas as respostas são retornadas diretamente a `dns.poly.edu`. Teoricamente, qualquer consulta DNS pode ser iterativa ou recursiva. Por exemplo, a Figura 2.22 mostra uma cadeia de consultas DNS na qual todas elas são recursivas. Na prática, as consultas normalmente seguem o padrão mostrado na Figura 2.21: a consulta do hospedeiro requisitante ao servidor de nomes local é recursiva e todas as outras são iterativas.

Cache DNS

Até aqui, nossa discussão ignorou o **cache DNS**, uma característica muito importante do sistema DNS. Na realidade, o DNS explora extensivamente o cache para melhorar o desempenho quanto ao atraso e reduzir o número de mensagens DNS que ricocheteia pela Internet. A ideia que está por trás do cache DNS é muito simples. Em uma cadeia de consultas, quando um servidor de nomes recebe uma resposta DNS (contendo, por exemplo, o mapeamento de um nome de hospedeiro para um endereço IP), ele pode fazer cache das informações da resposta em sua memória local. Por exemplo, na Figura 2.21, toda vez que o servidor de nomes local `dns.poly.edu` recebe uma resposta de algum servidor DNS, pode fazer cache de qualquer informação contida na resposta. Se um par nome de hospedeiro/endereço IP estiver no cache de um servidor DNS e outra consulta chegar ao mesmo servidor para o mesmo nome de máquina, o servidor de nomes poderá fornecer o endereço IP desejado, mesmo que não tenha autoridade para esse nome. Como hospedeiros e mapeamentos entre hospedeiros e endereços IP não são, de modo algum, permanentes, após um período de tempo (frequentemente dois dias), os servidores DNS descartam as informações armazenadas em seus caches.

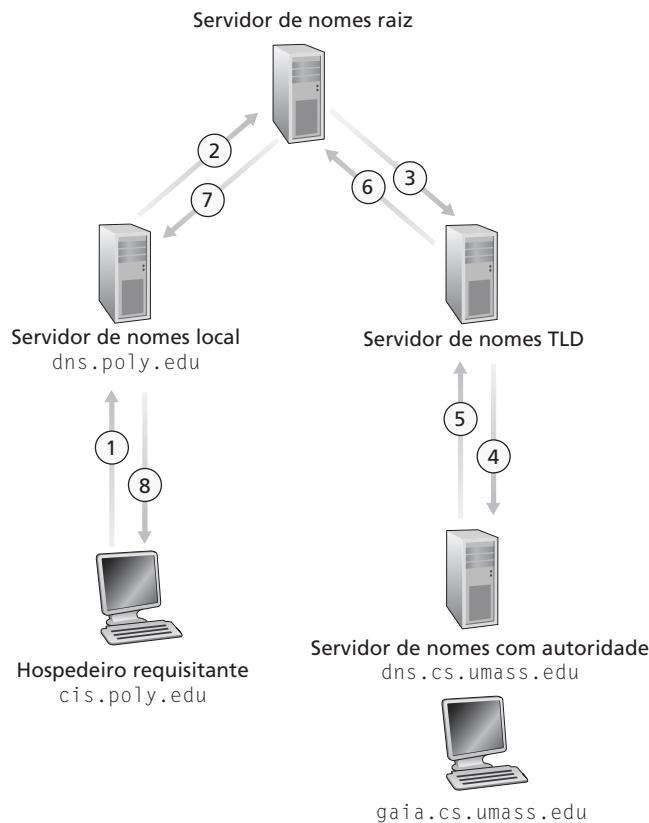


Figura 2.22 Consultas recursivas em DNS

Como exemplo, imagine que um hospedeiro `apricot.poly.edu` consulte `dns.poly.edu` para o endereço IP da máquina `cnn.com`. Além disso, suponha que algumas horas mais tarde outra máquina da Polytechnic University, digamos, `kiwi.poly.fr` também consulte `dns.poly.edu` para o mesmo nome de hospedeiro. Por causa do cache, o servidor local poderá imediatamente retornar o endereço IP de `cnn.com` a esse segundo hospedeiro requisitante, sem ter de consultar quaisquer outros servidores DNS. Um servidor de nomes local também pode fazer cache de endereços IP de servidores TLD, permitindo, assim, que servidores de nomes locais evitem os servidores de nomes raiz em uma cadeia de consultas (isso acontece frequentemente).

2.5.3 Registros e mensagens DNS

Os servidores de nomes que juntos implementam o banco de dados distribuído do DNS armazenam registros de recursos (RR) que fornecem mapeamentos de nomes de hospedeiros para endereços IP. Cada mensagem de resposta DNS carrega um ou mais registros de recursos. Nesta seção e na subsequente, apresentaremos uma breve visão geral dos registros de recursos e mensagens DNS. Para mais detalhes, consulte [Abitz, 1993] ou [RFC 1034; RFC 1035].

Um registro de recurso é uma tupla de quatro elementos que contém os seguintes campos:

(Name, Value, Type, TTL)

TTL é o tempo de vida útil do registro de recurso; determina quando um recurso deve ser removido de um cache. Nos exemplos de registros dados a seguir, ignoramos o campo TTL. Os significados de Name e Value dependem de Type:

Se Type=A, então Name é um nome de hospedeiro e Value é o endereço IP para o nome de hospedeiro. Assim, um registro Type A fornece o mapeamento padrão de nomes hospedeiros para endereços IP. Como exemplo, (`relay1.bar.foo.com`, `145.37.93.126`, A) é um registro com Type igual a A.



Se Type=NS, então Name é um domínio (como foo.com) e Value é o nome de um servidor de nomes com autoridade que sabe como obter os endereços IP para hospedeiros do domínio. Esse registro é usado para encaminhar consultas DNS ao longo da cadeia de consultas. Como exemplo, (foo.com, dns.foo.com, NS) é um registro com Type igual a NS.

Se Type=CNAME, então Value é um nome canônico de hospedeiro para o apelido de hospedeiro contido em Name. Esse registro pode fornecer aos hospedeiros consultantes o nome canônico correspondente a um apelido de hospedeiro. Como exemplo, (foo.com, relay1.bar.foo.com, CNAME) é um registro CNAME.

Se Type=MX, então Value é o nome canônico de um servidor de correio cujo apelido de hospedeiro está contido em Name. Como exemplo, (foo.com. mail.bar.foo.com, MX) é um registro MX. Registros MX permitem que os nomes de hospedeiros de servidores de correio tenham apelidos simples. Note que usando o registro MX, uma empresa pode ter o mesmo apelido para seu servidor de arquivo e para um de seus outros servidores (tal como seu servidor Web). Para obter o nome canônico do servidor de correio, um cliente DNS consultaria um registro MX; para obter o nome canônico do outro servidor, o cliente DNS consultaria o registro CNAME.

Se um servidor de nomes tiver autoridade para um determinado nome de hospedeiro, então conterá um registro Type A para o nome de hospedeiro. (Mesmo que não tenha autoridade, o servidor de nomes pode conter um registro Type A em seu cache.) Se um servidor não tiver autoridade para um nome de hospedeiro, conterá um registro Type NS para o domínio que inclui o nome e um registro Type A que fornece o endereço IP do servidor de nomes no campo Value do registro NS. Como exemplo, suponha que um servidor TLD edu não tenha autoridade para o hospedeiro gaia.cs.umass.edu. Nesse caso, esse servidor conterá um registro para um domínio que inclui o hospedeiro gaia.cs.umass.edu, por exemplo (umass.edu, dns.umass.edu, NS). O servidor TLD edu conterá também um registro Type A, que mapeia o servidor de nome dns.umass.edu para um endereço IP, por exemplo (dns.umass.edu, 128.119.40.111, A).

Mensagens DNS

Abordamos anteriormente nesta seção mensagens de consulta e de resposta DNS, que são as duas únicas espécies de mensagem DNS. Além disso, tanto as mensagens de consulta como as de resposta têm o mesmo formato, como mostra a Figura 2.23. A semântica dos vários campos de uma mensagem DNS é a seguinte:

Os primeiros 12 bytes formam a *seção de cabeçalho*, que tem vários campos. O primeiro campo é um número de 16 bits que identifica a consulta. Esse identificador é copiado para a mensagem de resposta a uma consulta, permitindo que o cliente combine respostas recebidas com consultas enviadas. Há várias flags no campo de flag. Uma flag de consulta/resposta de 1 bit indica se a mensagem é uma consulta (0) ou uma resposta (1). Uma flag de autoridade de 1 bit é marcada em uma mensagem de resposta quando o servidor de nomes é um servidor com autoridade para um nome consultado. Uma flag de recursão desejada de 1 bit é estabelecida quando um cliente (hospedeiro ou servidor de nomes) quer que um servidor de nomes proceda recursivamente sempre que não tem o registro. Um campo de recursão disponível de 1 bit é marcado em uma resposta se o servidor de nomes suporta buscas recursivas. No cabeçalho, há também quatro campos de ‘número de’. Esses campos indicam o número de ocorrências dos quatro tipos de seção de dados que se seguem ao cabeçalho.

A *seção de pergunta* contém informações sobre a consulta que está sendo feita. Essa seção inclui (1) um campo de nome que contém o nome que está sendo consultado e (2) um campo de tipo que indica o tipo de pergunta que está sendo feito sobre o nome — por exemplo, um endereço de hospedeiro associado a um nome (Type A) ou o servidor de correio para um nome (Type MX).

Em uma resposta de um servidor de nomes, a *seção de resposta* contém os registros de recursos para o nome que foi consultado originalmente. Lembre-se de que em cada registro de recurso há o Type (por exemplo, A, NS, CSNAME e MX), o Value e o TTL. Uma resposta pode retornar vários RRs, já que

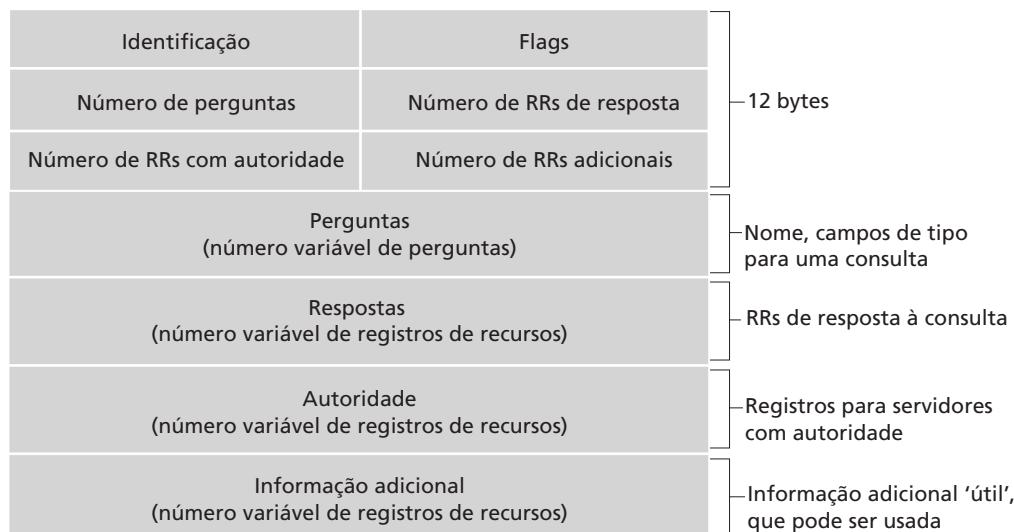


Figura 2.23 Formato da mensagem DNS

um nome de hospedeiro pode ter diversos endereços IP (por exemplo, para servidores Web replicados, como já discutimos anteriormente nesta seção).

A seção de autoridade contém registros de outros servidores com autoridade.

A seção adicional contém outros registros úteis. Por exemplo, o campo resposta em uma resposta a uma consulta MX conterá um registro de recurso que informa o nome canônico de um servidor de correio. A seção adicional conterá um registro Type A que fornece o endereço IP para o nome canônico do servidor de correio.

Você gostaria de enviar uma mensagem de consulta DNS diretamente de sua máquina a algum servidor DNS? Isso pode ser feito facilmente com o **programa nslookup**, que está disponível na maioria das plataformas Windows e UNIX. Por exemplo, se um hospedeiro executar Windows, abra o Command Prompt e chame o programa nslookup simplesmente digitando ‘nslookup’. Depois de chamar o programa, você pode enviar uma consulta DNS a qualquer servidor de nomes (raiz, TLD ou com autoridade). Após receber a mensagem de resposta do servidor de nomes, o nslookup apresentará os registros incluídos na resposta (em formato que pode ser lido normalmente). Como alternativa a executar nslookup na sua própria máquina, você pode visitar um dos muitos sites Web que permitem o emprego remoto do programa. (Basta digitar ‘nslookup’ em um buscador e você será levado a um desses sites.)

Para inserir registros no banco de dados do DNS

A discussão anterior focalizou como são extraídos registros do banco de dados DNS. É possível que você esteja se perguntando como os registros entraram no banco de dados em primeiro lugar. Vamos examinar como isso é feito no contexto de um exemplo específico. Imagine que você acabou de criar uma nova empresa muito interessante denominada Network Utopia. A primeira coisa que você certamente deverá fazer é registrar o nome de domínio networkutopia.com em uma entidade registradora. Uma **entidade registradora** é uma entidade comercial que verifica se o nome de domínio é exclusivo, registra-o no banco de dados do DNS (como discutiremos mais adiante) e cobra uma pequena taxa por seus serviços. Antes de 1999, uma única entidade registradora, a Network Solutions, detinha o monopólio do registro de nomes para os domínios .com, .net e .org. Mas agora existem muitas entidades registradoras credenciadas pela Internet Corporation for Assigned Names and Numbers (ICANN) competindo por clientes. Uma lista completa dessas entidades está disponível em <http://www.internic.net>.



Segurança em foco

Vulnerabilidades do DNS

Vimos que o DNS é um componente fundamental da infraestrutura da Internet, com muitos serviços importantes — incluindo a Web e o e-mail — simplesmente incapazes de funcionar sem ele. Desta maneira, perguntamos: como o DNS pode ser atacado? O DNS é um alvo esperando para ser atingido, pois causa dano à maioria das aplicações da Internet junto com ele?

O primeiro tipo de ataque que vem à mente é o ataque inundação na largura de banda DDoS (veja a Seção 1.6) contra servidores DNS. Por exemplo, um atacante pode tentar enviar para cada servidor DNS raiz uma inundação de pacotes, fazendo com que a maioria das consultas DNS legítimas nunca seja respondida. Tal ataque DDoS em larga escala contra servidores DNS raiz aconteceu em 21 de outubro de 2002. Nesse ataque, os atacantes se aproveitavam de um botnet para enviar centenas de mensagens ping para cada um dos servidores DNS raiz. (As mensagens ICMP são discutidas no Capítulo 4. Por enquanto, basta saber que os pacotes ICMP são tipos especiais de datagramas IP.) Felizmente, esse ataque em larga escala causou um dano mínimo, tendo um pequeno ou nenhum impacto sobre a experiência dos usuários com a Internet. Os atacantes obtiveram êxito ao direcionar centenas de pacotes aos servidores raiz. Mas muitos dos servidores DNS raiz foram protegidos por filtros de pacotes, configurados para sempre bloquear todas as mensagens ping ICMP encaminhadas aos servidores raiz. Desse modo, esses servidores protegidos foram poupadados e funcionaram normalmente. Além disso, a maioria dos servidores DNS locais oculta os endereços IP dos servidores de domínio de nível superior, permitindo que o processo de consulta ultrapasse frequentemente os servidores DNS raiz.

Um ataque DDoS potencialmente mais eficaz contra o DNS seria enviar uma inundação de consultas DNS aos servidores de domínio de alto nível, por exemplo, para todos os servidores de domínio que lidam com o domínio .com. Seria mais difícil filtrar as consultas DNS direcionadas aos servidores DNS; e os servidores de domínio de alto nível não são ultrapassados tão facilmente quanto os servidores raiz. Mas a gravidade de tal ataque poderia ser parcialmente amenizada pelo cache nos servidores DNS locais.

O DNS poderia ser atacado potencialmente de outras maneiras. Em um ataque de homem no meio, o atacante intercepta consultas do hospedeiro e retorna respostas falsas. No ataque de envenenamento, o atacante envia respostas falsas a um servidor DNS, fazendo com que o servidor armazene os registros falsos em sua cache. Ambos os ataques podem ser utilizados, por exemplo, para redirecionar um usuário da Web inocente ao site Web do atacante. Esses ataques, entretanto, são difíceis de implementar, uma vez que requerem a intercepção de pacotes ou o estrangulamento de servidores [Skoudis, 2006].

Outro ataque DNS importante não é um ataque ao serviço DNS por si mesmo, mas, em vez disso, se aproveitar da infraestrutura do DNS para lançar um ataque DDoS contra um hospedeiro-alvo (por exemplo, o servidor de mensagens de sua universidade). Nesse ataque, o atacante envia consultas DNS para muitos servidores DNS autoritativos, com cada consulta tendo o endereço-fonte falsificado do hospedeiro alvo. Os servidores DNS, então, enviam suas respostas diretamente para o hospedeiro-alvo. Se as consultas puderem ser realizadas de tal maneira que uma resposta seja muito maior (em bytes) do que uma consulta (denominada amplificação), então o atacante pode entupir o alvo sem ter que criar muito de seu próprio tráfego. Tais ataques de reflexão que exploram o DNS possuem um sucesso limitado até hoje [Mirkovic, 2005].

Em resumo, não houve um ataque que tenha interrompido o serviço DNS com sucesso. Houve ataques refletores bem-sucedidos; entretanto, eles podem ser (e estão sendo) abordados por uma configuração apropriada de servidores DNS.

Ao registrar o nome de domínio `networkutopia.com`, você também precisará informar os nomes e endereços IP dos seus servidores DNS com autoridade, primários e secundários. Suponha que os nomes e endereços IP sejam `dns1.networkutopia.com`, `dns2.networkutopia.com`, `212.212.212.1` e `212.212.212.2`. A entidade registradora ficará encarregada de providenciar a inserção dos registros Type NS e de um registro Type A nos servidores TLD do domínio `com` para cada um desses dois servidores de nomes com autoridade. Especificamente para o servidor primário com autoridade `networkutopia.com`, a autoridade registradora inseriria no sistema DNS os dois registros de recursos seguintes:

`(networkutopia.com, dns1.networkutopia.com, NS)`

`(dns1.networkutopia.com, 212.212.212.1, A)`

Não esqueça de providenciar também a inserção em seus servidores de nomes com autoridade do registro de recurso Type A para seu servidor Web `www.networkutopia.com` e o registro de recurso Type MX para seu servidor de correio `mail.networkutopia.com`. (Até há pouco tempo, o conteúdo de cada servidor DNS era configurado estaticamente, por exemplo, a partir de um arquivo de configuração criado por um gerenciador de sistema. Mais recentemente, foi acrescentada ao protocolo DNS uma opção UPDATE que permite que dados sejam dinamicamente acrescentados no banco de dados ou apagados deles por meio de mensagens DNS. O [RFC 2136] e o [RFC 3007] especificam atualizações dinâmicas do DNS.)

Quando todas essas etapas estiverem concluídas, o público em geral poderá visitar seu site Web e enviar e-mails aos empregados de sua empresa. Vamos concluir nossa discussão do DNS verificando que essa afirmação é verdadeira, o que também ajudará a solidificar aquilo que aprendemos sobre o DNS. Suponha que Alice, que está na Austrália, queira consultar a página Web `www.networkutopia.com`. Como discutimos anteriormente, seu provedor primeiramente enviará uma consulta DNS a seu servidor de nomes local, que então contatará um servidor TLD do domínio `com`. (O servidor de nomes local também terá de contatar um servidor de nomes raiz caso não tenha em seu cache o endereço de um servidor TLD `com`.) Esse servidor TLD contém os registros de recursos Type NS e Type A citados anteriormente, porque a entidade registradora já os tinha inserido em todos os servidores TLD `com`. O servidor TLD `com` envia uma resposta ao servidor de nomes local de Alice, contendo os dois registros de recursos. Então, o servidor de nomes local envia uma consulta DNS a `212.212.212.1`, solicitando o registro Type A correspondente a `www.networkutopia.com`. Este registro provê o endereço IP do servidor Web desejado, digamos, `212.212.71.4`, que o servidor local de nomes transmite para o provedor de Alice. Agora, o browser de Alice pode iniciar uma conexão TCP com o provedor `212.212.71.4` e enviar uma requisição HTTP pela conexão. Ufa! Acontecem muito mais coisas do que percebemos quando navegamos na Web!

2.6 Aplicações P2P

As aplicações descritas neste capítulo até agora — inclusive a Web, e-mail e DNS — todas empregam arquiteturas cliente-servidor com dependência significativa em servidores com infraestrutura que sempre permanecem ligados. Como consta na Seção 2.1.1, com uma arquitetura P2P, há dependência mínima (se houver) de servidores com infraestrutura que permanecem sempre ligados. Em vez disso, duplas de provedores intermitentemente conectados, chamados pares, comunicam-se diretamente entre si. Os pares não são de propriedade de um provedor de serviços, mas sim de desktops e laptops controlados por usuários.

Nesta seção, examinaremos três diferentes aplicações que são particularmente bem apropriadas a projetos P2P. A primeira é a distribuição de arquivos, em que a aplicação distribui um arquivo a partir de uma única fonte para um grande número de pares. A distribuição de arquivos é um bom local para iniciar a investigação de P2P, visto que expõe claramente a autoescalabilidade de arquiteturas P2P. Como exemplo específico para distribuição de arquivos, descreveremos o popular sistema BitTorrent. A segunda aplicação P2P que examinaremos é um banco de dados distribuído em uma grande comunidade de pares. Para essa aplicação, exploraremos o conceito de uma Distributed Hash Table (DHT). Por fim, para nossa terceira aplicação, examinaremos o Skype, uma aplicação de telefonia P2P da Internet de sucesso fenomenal.

2.6.1 Distribuição de arquivos P2P

Começaremos nossa investida em P2P considerando uma aplicação bastante natural, ou seja, a distribuição de um grande arquivo a partir de um único servidor a um grande número de hospedeiros (chamados pares). O arquivo pode ser uma nova versão do sistema operacional Linux, um patch de software para um sistema operacional ou aplicação existente, um arquivo de música MP3 ou um arquivo de vídeo MPEG. Em uma distribuição de arquivo cliente-servidor, o servidor deve enviar uma cópia do arquivo para cada um dos pares — colocando um enorme fardo sobre o servidor e consumindo uma grande quantidade de banda do servidor. Na distribuição de arquivos P2P, cada par pode redistribuir qualquer parte do arquivo que recebeu para outros pares, auxiliando, assim, o servidor no processo de distribuição. Atualmente (outono [Nos EUA] de 2009), o protocolo de distribuição de arquivos P2P mais popular é o BitTorrent [BitTorrent, 2009]. Originalmente desenvolvido por Bram Cohen (consulte a entrevista com Bram Cohen no final deste capítulo), há atualmente muitos diferentes clientes independentes de BitTorrent conforme o protocolo do BitTorrent, assim como há diversos clientes de navegadores Web conformes ao protocolo HTTP. Nesta subseção, examinaremos primeiro a autoescalabilidade de arquiteturas P2P no contexto de distribuição de arquivos. Então, descreveremos o BitTorrent em um certo nível de detalhes, destacando suas características mais importantes.

Escalabilidade de arquiteturas P2P

Para comparar arquiteturas cliente-servidor com arquiteturas P2P, e ilustrar a inerente autoescalabilidade de P2P, consideraremos um modelo quantitativo simples para a distribuição de um arquivo para um conjunto fixo de pares para ambos os tipos de arquitetura. Conforme demonstrado na Figura 2.24, o servidor e os pares são conectados por enlaces de acesso da Internet. A taxa de upload do enlace de acesso do servidor é denotada por u_s , e a taxa de upload do enlace de acesso do par i é denotada por u_i , e a taxa de download do enlace de acesso do par i é denotada por d_i . O tamanho do arquivo a ser distribuído (em bits) é denotado por F e o número de pares que querem obter uma cópia do arquivo, por N . O tempo de distribuição é o tempo necessário para que todos os N pares obtenham uma cópia do arquivo. Em nossa análise do tempo de distribuição abaixo, tanto para a arquitetura cliente-servidor como para a arquitetura P2P, fazemos a hipótese simplificada (e geralmente precisa [Akella, 2003]) de que o núcleo da Internet tem largura de banda abundante, o que implica que todos os

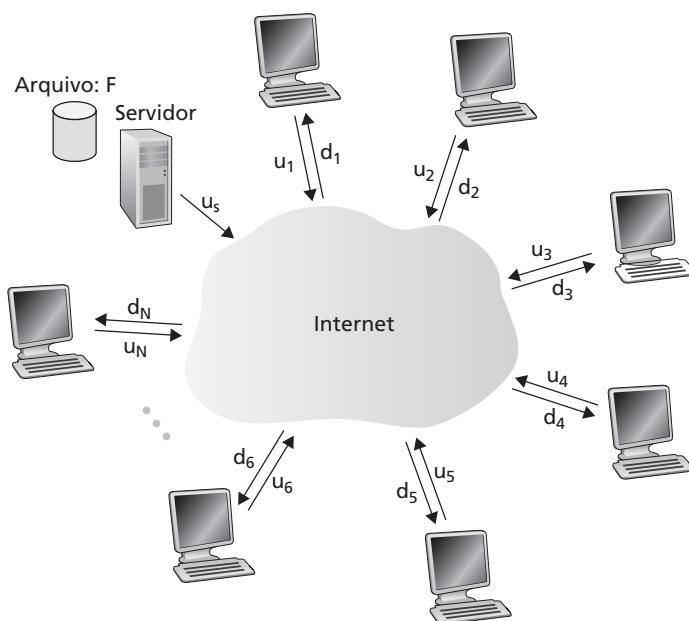


Figura 2.24 Um problema ilustrativo de distribuição de arquivo

gargalos encontram-se no acesso à rede. Suponha também que o servidor e os clientes não participam de nenhuma outra aplicação de rede, para que toda sua largura de banda de acesso de upload e download possa ser totalmente devotada à distribuição do arquivo.

Determinemos primeiro o tempo de distribuição para a arquitetura cliente-servidor, que denotaremos por D_{cs} . Na arquitetura cliente-servidor, nenhum dos pares auxilia na distribuição do arquivo. Fazemos as observações abaixo:

O servidor deve transmitir uma cópia do arquivo a cada um dos N pares. Assim, o servidor deve transmitir NF bits. Como a taxa de upload do servidor é de u_s , o tempo para distribuição do arquivo deve ser de pelo menos NF/u_s .

Deixemos que d_{\min} denote a taxa de download do par com menor taxa de download, ou seja, $d_{\min} = \min\{d_1, d_p, \dots, d_n\}$. O par com a menor taxa de download não pode obter todos os bits F do arquivo em menos de F/d_{\min} segundos. Assim, o tempo de distribuição mínimo é de pelo menos F/d_{\min} .

Reunindo essas observações, temos:

$$D_{cs} \geq \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{\min}} \right\}.$$

Isso proporciona um limite inferior para o tempo mínimo de distribuição para a arquitetura cliente-servidor. Nos problemas do final do capítulo, você deverá demonstrar que o servidor pode programar suas transmissões de forma que o limite inferior seja sempre alcançado. Portanto, consideraremos esse limite inferior fornecido anteriormente como o tempo real de distribuição, ou seja,

$$D_{cs} = \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{\min}} \right\}. \quad (2.1)$$

Vemos, a partir da Equação 2.1, que para N grande o suficiente, o tempo de distribuição cliente-servidor é dado por NF/u_s . Assim, o tempo de distribuição aumenta linearmente com o número de pares N . Portanto, por exemplo, se o número de pares de uma semana para a outra for multiplicado por mil, de mil para um milhão, o tempo necessário para distribuir o arquivo para todos os pares aumentará mil vezes.

Passemos agora para uma análise semelhante para a arquitetura P2P, em que cada par pode auxiliar o servidor na distribuição do arquivo. Em particular, quando um par recebe alguns dados do arquivo, ele pode usar sua própria capacidade de upload para redistribuir os dados a outros pares. Calcular o tempo de distribuição para a arquitetura P2P é, de certa forma, mais complicado do que para a arquitetura cliente-servidor, visto que o tempo de distribuição depende de como cada par distribui parcelas do arquivo aos outros pares. Não obstante, uma simples expressão para o tempo mínimo de distribuição pode ser obtida [Kumar, 2006]. Para essa finalidade, faremos as observações a seguir:

No início da distribuição, apenas o servidor tem o arquivo. Para levar esse arquivo à comunidade de pares, o servidor deve enviar cada bit do arquivo pelo menos uma vez para seu enlace de acesso. Assim, o tempo de distribuição mínimo é de pelo menos F/u_s . (Diferente do esquema cliente-servidor, um bit enviado uma vez pelo servidor pode não precisar ser enviado novamente, visto que os pares podem redistribuir entre si esse bit).

Assim como na arquitetura cliente-servidor, o par com a menor taxa de download não pode obter todos os bits F do arquivo em menos de F/d_{\min} segundos. Assim, o tempo mínimo de distribuição é de pelo menos F/d_{\min} .

Finalmente, observemos que a capacidade de upload total do sistema como um todo é igual à taxa de upload do servidor mais as taxas de upload de cada um dos pares individuais, ou seja, $u_{\text{total}} = u_s + u_1 + \dots + u_n$. O sistema deve entregar (fazer o upload de) F bits para cada um dos N pares, entregando assim um total de NF bits. Isso não pode ser feito em uma taxa mais rápida do que u_{total} . Assim, o tempo mínimo de distribuição é também de pelo menos $NF/(u_s + u_1 + \dots + u_n)$.

Juntando essas três observações, obtemos o tempo mínimo de distribuição para P2P, denotado por D_{P2P} .

$$D_{\text{P2P}} \geq \max \left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\} \quad (2.2)$$

A Equação 2.2 fornece um limite inferior para o tempo mínimo de distribuição para a arquitetura P2P. Ocorre que, se imaginarmos que cada par pode redistribuir um bit assim que o recebe, há um esquema de redistribuição que, de fato, alcança esse limite inferior [Kumar, 2006] (Provaremos um caso especial desse resultado na lição de casa). Na realidade, quando blocos do arquivo são redistribuídos, em vez de bits individuais, a Equação 2.2 serve como uma boa aproximação do tempo mínimo real de distribuição. Assim, peguemos o limite inferior fornecido pela Equação 2.2 como o tempo mínimo real de distribuição, que é:

$$D_{\text{P2P}} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\} \quad (2.3)$$

A Figura 2.25 compara o tempo mínimo de distribuição para as arquiteturas cliente-servidor e P2P, pressupondo que todos os pares têm a mesma taxa de upload u . Na Figura 2.25, temos definido que $F/u = 1$ hora, $u_s = 10u$ e $d_{\min} \geq u_s$. Assim, um par pode transmitir todo o arquivo em uma hora, sendo a taxa de transmissão do servidor 10 vezes a taxa de upload do par, e (para simplicidade) as taxas de download de par são definidas grandes o suficiente de forma a não ter efeito. Vemos na Figura 2.25 que, para a arquitetura cliente-servidor, o tempo de distribuição aumenta linearmente e sem limite, conforme aumenta o número de pares. No entanto, para a arquitetura P2P, o tempo mínimo de distribuição não é apenas menor do que o tempo de distribuição da arquitetura cliente-servidor; é também de menos do que uma hora para qualquer número de pares N . Assim, aplicações com a arquitetura P2P podem ter autoescalabilidade. Essa escalabilidade é uma consequência direta de pares sendo redistribuidores, bem como consumidores de bits.

BitTorrent

O BitTorrent é um protocolo P2P popular para distribuição de arquivos [BitTorrent, 2009]. No jargão do BitTorrent, a coleção de todos os pares que participam da distribuição de um determinado arquivo é chamada de *torrent*. Os pares em um torrent fazem o download de *blocos* de tamanho igual do arquivo entre si, com um

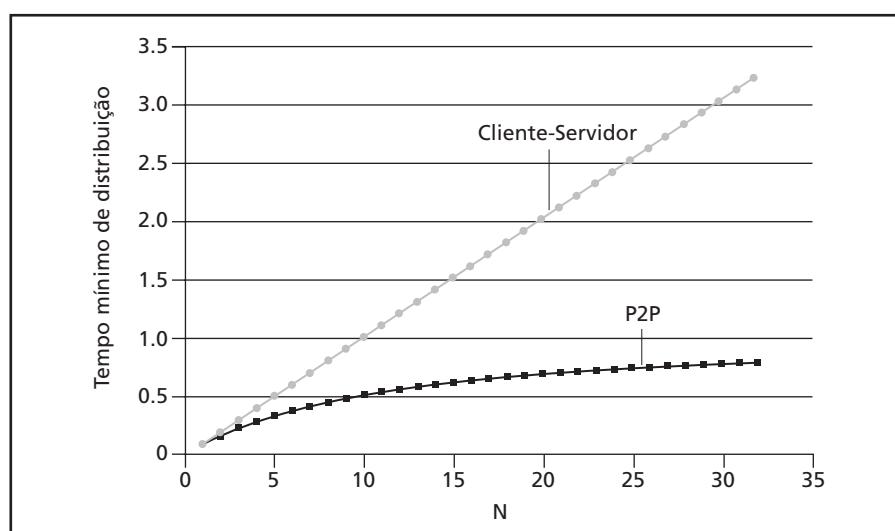


Figura 2.25 Tempo de distribuição para arquiteturas P2P e cliente-servidor

tamanho típico de bloco de 256 KBytes. Quando um par entra em um torrent, ele não tem nenhum bloco. Com o tempo, ele acumula mais blocos. Enquanto ele faz o download de blocos, faz também uploads de blocos para outros pares. Uma vez que um par adquire todo o arquivo, ele pode (de forma egoísta) sair do torrent ou (de forma altruísta) permanecer no torrent e continuar fazendo o upload de blocos a outros pares. Além disso, qualquer par pode sair do torrent a qualquer momento com apenas um subconjunto de blocos, e depois voltar.

Observemos agora, mais atentamente, como opera o BitTorrent. Como o BitTorrent é um protocolo e sistema complicado, descreveremos apenas seus mecanismos mais importantes, ignorando alguns detalhes; isso nos permitirá ver a floresta através das árvores. Cada torrent tem um nó de infraestrutura chamado rastreador. Quando um par chega em um torrent, ele se registra com o rastreador e periodicamente informa ao rastreador que ainda está no torrent. Dessa forma, o rastreador mantém um registro dos pares que participam do torrent. Um determinado torrent pode ter menos de dez ou mais de mil pares participando a qualquer momento.

Como demonstrado na Figura 2.26, quando um novo par, Alice, chega no torrent, o rastreador seleciona aleatoriamente um subconjunto de pares (para dados concretos, digamos que sejam 50) do conjunto de pares participantes, e envia os endereços IP desses 50 pares para Alice. Com a lista de pares, Alice tenta estabelecer conexões TCP simultâneas com todos os pares da lista. Chamaremos todos os pares com quem Alice consiga estabelecer uma conexão TCP de “pares vizinhos” (na Figura 2.26, Alice é representada com apenas três pares vizinhos. Normalmente, ela teria muito mais). Com o tempo, alguns desses pares podem sair e outros pares (fora dos 50 iniciais) podem tentar estabelecer conexões TCP com Alice. Portanto, os pares vizinhos de um par podem flutuar com o tempo.

A qualquer momento, cada par terá um subconjunto de blocos do arquivo, com pares diferentes com subconjuntos diferentes. Periodicamente, Alice pedirá a cada um de seus pares vizinhos (nas conexões TCP) a lista de quais blocos eles têm. Caso Alice tenha L vizinhos diferentes, ela obterá L listas de blocos. Com essa informação, Alice emitirá solicitações (novamente, nas conexões TCP) de blocos que ela não tem.

Portanto, a qualquer momento, Alice terá um subconjunto de blocos e saberá quais blocos seus vizinhos têm. Com essa informação, Alice terá duas decisões importantes a fazer. Primeiro, quais blocos ela deve solicitar primeiro de seus vizinhos, e segundo, a quais vizinhos ela deve enviar os blocos solicitados. Ao decidir quais

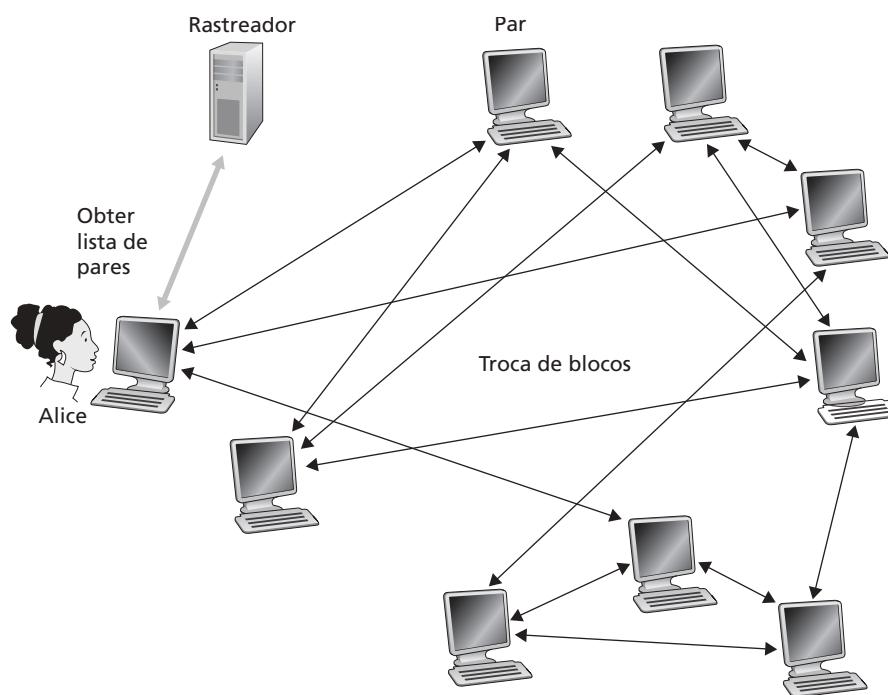


Figura 2.26 Distribuição de arquivos com o BitTorrent

blocos solicitar, Alice usa uma técnica chamada *rarest first* (o mais raro primeiro). A ideia é determinar, dentre os blocos que ela não tem, quais são os mais raros dentre seus vizinhos (ou seja, os blocos que têm o menor número de cópias repetidas em seus vizinhos) e então solicitar esses blocos mais raros primeiro. Dessa forma, os blocos mais raros são redistribuídos mais rapidamente, procurando (*grosso modo*) equalizar os números de cópias de cada bloco no torrent.

Para determinar a quais pedidos atender, o BitTorrent usa um algoritmo de troca inteligente. A ideia básica é Alice dar prioridade aos vizinhos que atualmente fornecem seus dados *com a maior taxa*. Especificamente, para cada um de seus vizinhos, Alice mede continuamente a taxa em que recebe bits e determina os quatro pares que lhe fornecem na melhor taxa. Então, ela reciprocamente envia blocos a esses mesmos quatro pares. A cada 10 segundos, ela recalcula as taxas e possivelmente modifica o conjunto de quatro pares. No jargão do BitTorrent, esses quatro pares são chamados de *unchoked* (não sufocado). É importante informar que, a cada 30 segundos, ela também escolhe um vizinho adicional aleatoriamente e envia blocos para ele. Chamaremos o vizinho escolhido aleatoriamente de Bob. No jargão de BitTorrent, Bob é chamado de *optimistically unchoked*. Como Alice envia dados a Bob, ela pode se tornar um dos quatro melhores transmissores para Bob, caso em que Bob começaria a enviar dados para Alice. Caso a taxa em que Bob envie dados a Alice seja alta o suficiente, Bob pode, em troca, tornar-se um dos quatro melhores transmissores para Alice. Em outras palavras, Alice aleatoriamente escolherá um novo parceiro de troca e a começará com ele. Caso os dois pares estejam satisfeitos com a troca, eles colocarão um ao outro nas suas listas de quatro melhores pares e continuarão a troca até que um dos pares encontre um parceiro melhor. O efeito é que pares capazes de fazer uploads em taxas compatíveis tendem a se encontrar. A seleção aleatória de vizinho também permite que novos pares obtenham blocos, de forma que possam ter algo para trocar. Todos os pares vizinhos, além desses cinco pares (quatro pares “top” e um em experiência) estão “sufocados”, ou seja, não recebem nenhum bloco de Alice. O BitTorrent tem diversos mecanismos interessantes não discutidos aqui, incluindo pedaços (miniblocos), pipelining (tubulação), primeira seleção aleatória, modo *endgame* (fim de jogo) e *anti-snubbing* (antirrejeição) [Cohen 2003].

O mecanismo de incentivo para troca descrito acima é normalmente chamado de *tit-for-tat* (olho por olho) [Chen 2003]. Demonstrou-se que esse esquema de incentivo pode ser burlado [Liogkas, 2006; Locher, 2006; Piatek, 2007]. Não obstante, o ecossistema do BitTorrent é muito bem-sucedido, com milhões de pares simultâneos compartilhando arquivos ativamente em centenas de milhares de torrents. Caso o BitTorrent tivesse sido projetado sem o tit-for-tat (ou uma variante), mas com o restante da mesma maneira, ele provavelmente nem existisse mais, visto que a maioria dos usuários são pessoas que apenas querem obter as coisas gratuitamente [Saroui, 2002].

Variantes interessantes do protocolo BitTorrent são propostas [Guo, 2005; Piatek, 2007]. Além disso, muitas das aplicações de transmissão em tempo real P2P, como PPLive e ppstream, foram inspirados pelo BitTorrent [Hei 2007].

2.6.2 Distributed Hash Tables (DHTs)

Um componente crítico de muitas aplicações P2P e outras aplicações distribuídas é um índice (ou seja, um banco de dados simples), que suporta operações de busca e atualização. Quando esse banco de dados é distribuído, os pares podem realizar caching de conteúdo e roteamento sofisticado de consultas entre si. Como a indexação e busca de informações são um componente crítico nesses sistemas, cobriremos uma técnica popular de indexação e busca, Distributed Hash Tables (DHTs).

Consideremos assim a construção de um banco de dados simples distribuído em um grande número (possivelmente milhões) de pares que suportam indexação e solicitação simples. As informações armazenadas em nosso banco de dados consistirão de duplas (chave, valor). Por exemplo, as chaves podem ser números de segurança social e os valores podem ser nomes humanos correspondentes; nesse caso, um exemplo de dupla chave-valor é (156-45-7081, Johnny Wu). Ou as duplas podem ser nomes de conteúdo (ex.: nomes de filmes, álbuns e software), e os valores podem ser endereços IP onde o conteúdo está armazenado; nesse caso, um exemplo de par chave-valor é (Led Zeppelin IV, 203.17.123.38). Pares consultam nossos bancos de dados fornecendo a chave: caso haja duplas (chave, valor) em seus bancos de dados que correspondam à chave, o banco de dados retorna

as duplas correspondentes ao par solicitante. Portanto, por exemplo, caso o banco de dados armazene números de segurança social e seus nomes humanos correspondentes, um par pode consultar um número de segurança social específico e o banco de dados retornará o nome do humano que possui aquele número de segurança social. Pares também devem ser capazes de inserir duplas (chave, valor) em nosso banco de dados.

A construção desse banco de dados é direta com uma arquitetura cliente-servidor na qual todos os pares (chave, valor) são armazenados em um servidor central. Essa abordagem centralizada também foi considerada em antigos sistemas P2P como o Napster. Mas o problema é significativamente mais desafiador e interessante em um sistema distribuído que consiste de milhões de pares conectados sem autoridade central. Em um sistema P2P, queremos distribuir as duplas (chave, valor) entre todos os pares, de forma que cada par tenha apenas um pequeno subconjunto da totalidade dos pares (chave, valor). Uma abordagem inocente para a construção desse banco de dados P2P é (1) espalhar aleatoriamente as duplas (chave, valor) entre os pares e (2) fazer com que cada par tenha uma lista dos endereços de IP de todos os pares participantes. Dessa forma, o par solicitante pode enviar uma solicitação a todos os outros pares, e os pares que tenham duplas (chave, valor) que correspondam à chave podem responder com as duplas correspondentes. Essa abordagem é completamente não escalável, logicamente, visto que cada par precisaria rastrear todos os outros pares (possivelmente milhões) e, pior, enviar cada solicitação a todos os pares.

Agora descreveremos uma abordagem elegante para um projeto de banco de dados P2P. Para isso, primeiro designaremos um identificador a cada par, em que cada identificador é um número inteiro na faixa $[0, 2^n - 1]$ de algum n fixo. Observe que cada identificador pode ser expresso por uma representação com n -bits. Vamos também exigir que cada chave seja um número inteiro na mesma faixa. O leitor astuto pode ter observado que as chaves de exemplo descritas anteriormente não são (números de segurança social e nomes de conteúdo) números inteiros. Para criar números inteiros a partir dessas chaves, precisaremos usar uma função hash que mapeie cada chave (por exemplo, número de segurança social) em um número inteiro na faixa $[0, 2^n - 1]$. Uma função de hash é uma função de muitos-para-um para a qual duas entradas diferentes podem ter a mesma saída (mesmo número inteiro), mas a probabilidade de terem a mesma saída é extremamente pequena (leitores não familiarizados com funções de hash podem querer consultar o Capítulo 7, que discute detalhadamente funções de hash). A função de hash é considerada publicamente disponível a todos os pares no sistema. Portanto, quando nos referirmos à “chave”, nos referimos ao hash da chave original. Portanto, por exemplo, caso a chave original seja “Led Zeppelin IV”, a chave será o número inteiro que corresponda ao hash de “Led Zeppelin IV”. Além disso, como estamos usando hashes de chaves, em vez das próprias chaves, nos referiremos ao banco de dados distribuído como **Distributed Hash Table (DHT)**.

Consideraremos agora o problema de armazenar as duplas (chave, valor) no DHT. A questão central aqui é definir uma regra para designar chaves a pares. Considerando que cada par tenha um identificador de número inteiro e cada chave também seja um número inteiro na mesma faixa, uma abordagem natural é designar cada dupla (chave, valor) ao par cujo identificador está *mais próximo* da chave. Para implementar esse esquema, precisaremos definir o que significa ‘mais próximo’, o que admite muitas convenções. Por conveniência, definiremos que o par mais próximo é o *sucessor imediato da chave*. Para visualizarmos, observaremos um exemplo específico. Suponha que $n = 4$, portanto, todos os identificadores de par e chave estarão na faixa de $[0, 15]$. Suponha ainda que haja oito pares no sistema com identificadores 1, 3, 4, 5, 8, 10, 12 e 15. Finalmente, suponha que queiramos armazenar o par chave-valor (11, Johnny Wu) em um dos oito pares. Mas em qual? Usando nossa convenção de mais próximo, como o par 12 é o sucessor imediato da chave 11, armazenaremos, portanto, a dupla (11, Johnny Wu) no par 12 [para concluir nossa definição de mais próximo, caso a chave seja exatamente igual a um dos identificadores do par, armazenaremos o par (chave-valor) em um par correspondente; e caso a chave seja maior do que todos os identificadores de par, usaremos uma convenção módulo- 2^n , que armazena o par (chave-valor) no par com o menor identificador].

Suponha agora que um par, Alice, queira inserir uma dupla (chave-valor) no DHT. Conceitualmente, é um processo objetivo: ela primeiro determina o par cujo identificador é o mais próximo da chave; então ela envia uma mensagem a esse par, instruindo-o a armazenar a dupla (chave, valor). Mas como Alice determina o par mais próximo da chave? Se Alice rastreasse todos os pares no sistema (IDs de par e endereços IP correspondentes), ela poderia

determinar localmente o par mais próximo. Mas essa abordagem requer que *cada* par rastreie *todos* os outros pares no DHT — o que é completamente impraticável para um sistema de grande escala com milhões de pares.

DHT circular

Para abordar esse problema, consideraremos agora organizar os pares em um círculo. Nessa disposição circular, cada par rastreia apenas seu sucessor imediato ($módulo\ 2^n$). Um exemplo desse círculo é exibido na Figura 2.27(a). Nesse exemplo, n é novamente 4 e há os mesmos oito pares do exemplo anterior. Cada par está ciente apenas de seu sucessor imediato; por exemplo, o par 5 sabe o endereço IP e o identificador do par 8, mas não sabe necessariamente nada sobre quaisquer outros pares no DHT. Essa disposição circular dos pares é um caso especial de uma rede sobreposta. Em uma rede sobreposta, os pares formam uma rede lógica abstrata que reside acima da rede de computadores “inferior” que consiste de enlaces físicos, roteadores e hospedeiros. Os enlaces em uma rede sobreposta não são físicos, mas enlaces virtuais entre duplas de pares. Na rede de sobreposição da Figura 2.27(a), há oito pares e oito enlaces sobrepostos; na sobreposição da Figura 2.27(b) há oito pares e 16 enlaces sobrepostos. Um único enlace de sobreposição normalmente usa muitas ligações físicas e roteadores físicos na rede inferior.

Usando a rede de sobreposição circular da Figura 2.27(a), suponha agora que o par 3 deseja determinar qual par no DHT é responsável pela chave 11 [para inserir ou para requisitar uma dupla (chave-valor)]. Usando a rede sobreposta circular, o par de origem (par 3) cria uma mensagem que pergunta “Quem é responsável pela chave 11?” e a envia a seu sucessor, o par 4. Sempre que um par recebe essa mensagem, como sabe o identificador de seu sucessor, pode determinar se é responsável (ou seja, mais próximo) pela chave em questão. Caso um par não seja responsável pela chave, ele simplesmente envia a mensagem a seu sucessor. Portanto, por exemplo, quando o par 4 recebe a mensagem perguntando sobre a chave 11, ele determina que não é responsável pela chave (porque seu sucessor está mais perto da dela), portanto, ele passa a mensagem a seu sucessor, ou seja, o par 5. Esse processo continua até que a mensagem chegue ao par 12, que determina que é o mais próximo da chave 11. A essa altura, o par 12 pode enviar uma mensagem de volta à origem, o par 3, indicando que é responsável pela chave 11.

O DHT circular oferece uma solução bastante elegante para reduzir a quantidade de informação sobreposta que cada par deve gerenciar. Em particular, cada par está ciente apenas de dois pares, seu sucessor imediato e seu predecessor imediato (por padrão, o par está ciente de seu predecessor, visto que este lhe envia mensagens). Porém, essa solução ainda introduz um novo problema. Embora cada par esteja ciente de dois pares vizinhos, para encontrar o nó responsável por uma chave (no pior das hipóteses), todos os N nós no DHT deverão encaminhar uma mensagem pelo círculo; $N/2$ mensagens são enviadas em média.

Assim, no projeto de um DHT, há uma troca entre o número de vizinhos que cada par tem de rastrear e o número de mensagens que o DHT precisa enviar para resolver uma única solicitação. Por um lado, se cada par

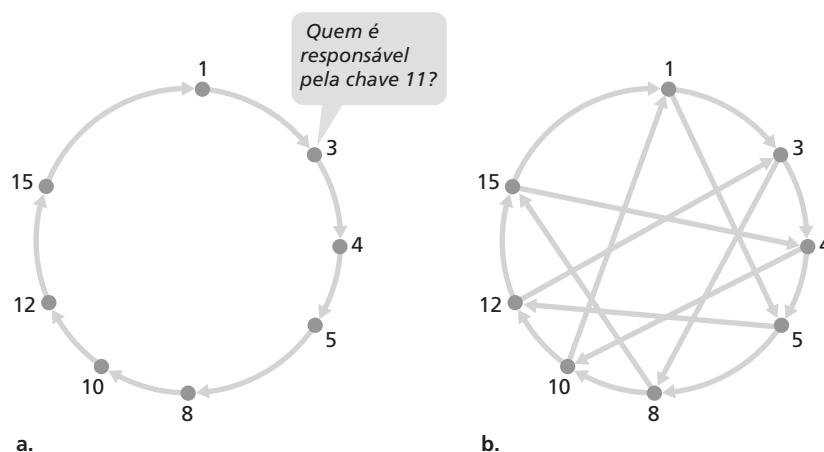


Figura 2.27 (a) Um DHT circular. O par 3 quer determinar quem é responsável pela chave 11. (b) Um DHT circular com atalhos

rastrear todos os outros pares (sobreposições de malha), apenas uma mensagem será enviada por solicitação, mas cada par deverá rastrear N pares. Por outro lado, com um DHT circular, cada par está ciente apenas de dois pares, mas $N/2$ mensagens são enviadas em média para cada solicitação. Felizmente, podemos refinar nossos projetos de DHTs de forma que o número de vizinhos por par, bem como o número de mensagens por solicitação seja mantido em um tamanho aceitável. Um desses refinamentos é usar a rede sobreposta circular como fundação, mas adicionar “atalhos” de forma que cada par não apenas rastreie seu sucessor imediato, mas também um número relativamente pequeno de pares espalhados pelo círculo. Um exemplo desse DHT circular com alguns atalhos é demonstrado na Figura 2.27(b). Atalhos são usados para expedir o roteamento das mensagens de solicitação. Especificamente, quando um par recebe uma mensagem que solicita uma chave, ele encaminha a mensagem ao vizinho (vizinho sucessor ou um dos vizinhos via atalho) que está mais perto da chave. Assim, na Figura 2.27(b), quando o par 4 recebe a mensagem solicitando a chave 11, ele determina que o par mais próximo (entre seus vizinhos) é seu vizinho no atalho 10 e então envia a mensagem diretamente ao par 10. Claramente, atalhos podem reduzir significativamente o número de mensagens usado para processar uma solicitação.

A próxima questão natural é “Quantos vizinhos no atalho cada par deve ter, e quais pares devem ser esses vizinhos de atalho?” Essa pergunta recebeu atenção significativa da comunidade de pesquisa [Stoica, 2001; Rowstron, 2001; Ratnasamy, 2001; Zhao, 2004; Maymounkov, 2002; Garces-Erce, 2003]. De forma importante, demonstrou-se que o DHT pode ser projetado de forma que tanto o número de vizinhos como o número de mensagens por solicitação seja da ordem de $\log N$, em que N é o número de pares. Esses projetos obtêm um compromisso satisfatório entre as soluções extremas de se usar topologias de sobreposição circular e de malha.

Peer churn

Em sistemas P2P, um par pode vir ou ir sem aviso. Assim, no projeto de um DHT, devemos nos preocupar em manter a sobreposição de DHT na presença desse peer churn. Para termos uma compreensão abrangente de como isso pode ser realizado, consideraremos mais uma vez o DHT circular da Figura 2.27(a). Para manejar o peer churn, exigiremos que cada par rastreie (ou seja, saiba o endereço IP de) seu primeiro e segundo sucessores; por exemplo, o par 4 agora rastreia tanto o par 5 como o par 8. Exigiremos também que cada par verifique periodicamente se seus dois sucessores estão vivos (por exemplo, enviando periodicamente mensagens de ping e pedindo respostas). Consideraremos agora como o DHT é mantido quando um par sai abruptamente. Por exemplo, suponha que o par 5 da Figura 2.27(a) saia abruptamente. Nesse caso, os dois pares precedentes ao par que saiu (4 e 3) saberão que o par saiu, pois não responde mais às mensagens de ping. Os pares 4 e 3 precisam, portanto, atualizar as informações do estado de seu sucessor. Consideraremos agora como o par 4 atualiza seu estado:

1. O par 4 substitui seu primeiro sucessor (par 5) por seu segundo sucessor (par 8).
2. O par 4, então, pergunta a seu novo primeiro sucessor (par 8) o identificador e o endereço IP de seu sucessor imediato (par 10). O par 4, então, torna o par 10 seu segundo sucessor.

Nos problemas, você deverá determinar como o par 3 atualiza suas informações de determinação de roteamento de sobreposição.

Tendo abordado brevemente o que deve ser feito quando um par sai, consideraremos agora o que acontece quando um par quer entrar no DHT. Digamos que um par com identificador 13 quer entrar no DHT, e quando entra, sabe apenas da existência do par 1 no DHT. O par 13 primeiro envia ao par 1 uma mensagem, perguntando “quem serão o predecessor e o sucessor do par 13?”. Essa mensagem é encaminhada através do DHT até alcançar o par 12, que percebe que será o predecessor do par 13, e que seu sucessor, o par 15, será o sucessor do par 13. Em seguida, o par 12 envia as informações de sucessor e predecessor ao par 13. O par 13, então, pode entrar no DHT, tornando o par 15 seu sucessor e notificando ao par 12 que deve mudar seu sucessor imediato para 13.

DHTs têm amplo uso na prática. Por exemplo, o BitTorrent usa o DHT Kademlia para criar um rastreador distribuído. No BitTorrent, a chave é o identificador do torrent e o valor é o endereço IP dos pares que atualmente participam dos torrents [Falkner, 2007; Neglia, 2007]; Dessa forma, solicitando ao DHT um identificador de torrent, um par de BitTorrent recém-chegado pode determinar o par responsável pelo identificador (ou seja, por determinar os pares no torrent). Após ter encontrado esse par, o par recém-chegado pode solicitar dele uma lista

de outros pares no torrent. DHTs são usados extensamente no sistema de compartilhamento de arquivos eMule para localizar conteúdos em pares [Liang, 2006].

2.6.3 Estudo de caso: telefonia por Internet P2P com Skype

O Skype é uma aplicação P2P imensamente popular, muitas vezes com sete ou oito milhões de usuários conectados simultaneamente. Além de fornecer serviço de telefonia PC para PC na Internet, o Skype oferece serviço de telefonia PC-para-telefone, telefone-para-PC e serviço de videoconferência PC-a-PC. Fundado pelos mesmos indivíduos que criaram o FastTrack e o Kazaa, o Skype foi adquirido pelo eBay em 2005 por US\$ 2,6 bilhões.

O Skype usa técnicas P2P de diversas maneiras inovadoras, ilustrando de uma boa maneira como o P2P pode ser usado em aplicações que vão além da distribuição de conteúdo e compartilhamento de arquivos. Assim como com programas de mensagens instantâneas, a telefonia PC-para-PC na Internet é inherentemente P2P, visto que, no núcleo do aplicativo, duplas de usuários (ou seja, pares) comunicam-se entre si em tempo real. Porém, o Skype também emprega técnicas P2P para duas outras funções importantes, que são localização de usuário e NAT traversal.

Os protocolos do Skype não são apenas proprietários, como todas as transmissões de pacotes do Skype (pacotes de controle e voz) são criptografadas. Não obstante, a partir do website do Skype e de diversos estudos de medição, os pesquisadores descobriram como o Skype geralmente funciona [Baset, 2006; Guha, 2006; Chen, 2006; Suh, 2006 Ren, 2006]. Assim como com o FastTrack, os nós no Skype são organizados em uma rede sobreposta hierárquica, com cada par classificado como superpar ou par comum. O Skype inclui um índice que mapeia os nomes de usuários do Skype a endereços IP atuais (e números de porta). Esse índice é distribuído entre os superpares. Quando Alice deseja telefonar para Bob, seu cliente Skype procura o índice distribuído para determinar o endereço IP atual de Bob. Como o protocolo do Skype é proprietário, atualmente não está claro como os mapeamentos de índice são organizados nos superpares, embora alguma forma de organização DHT seja muito possível.

Técnicas de P2P também são usadas em retransmissores do Skype, que são úteis para estabelecer chamadas entre hospedeiros em redes domésticas. Muitas configurações de redes domésticas fornecem acesso à Internet através de um roteador (tipicamente um roteador sem fio). Esses roteadores são, na verdade, mais do que roteadores, e normalmente incluem um elemento chamado Network Address Translator [Tradutor de Endereço da Internet] (NAT). Estudaremos NATs no Capítulo 4. Por enquanto, tudo do que precisamos saber é que um NAT impede que um hospedeiro de fora da rede doméstica inicie uma conexão com um hospedeiro dentro da rede doméstica. Caso *ambos* os chamadores do Skype tenham NATs, há um problema — nenhum deles pode aceitar uma chamada iniciada pelo outro, tornando aparentemente impossível uma chamada. O uso inteligente de superpares e retransmissores resolve muito bem esse problema. Suponha que quando Alice entra, ela recebe um superpar sem NAT. Alice pode iniciar uma sessão com seu superpar, visto que seu NAT apenas impede sessões iniciadas de fora de sua rede doméstica. Isso permite que Alice e seu superpar troquem mensagens de controle nessa sessão. O mesmo ocorre com Bob quando ele entra. Agora, quando Alice deseja telefonar para Bob, ela informa a seu superpar, que, por sua vez, informa ao superpar de Bob, que então informa a Bob sobre a chamada recebida de Alice. Caso Bob aceite a chamada, os dois superpares selecionam um terceiro superpar sem NAT — o nó de retransmissor — cujo trabalho será o de transmitir os dados entre Alice e Bob. Os superpares de Alice e de Bob, então, instruem Alice e Bob, respectivamente, a iniciarem uma sessão com o retransmissor. Alice, então, envia pacotes de voz ao retransmissor na conexão Alice-para-retransmissor (que foi iniciada por Alice), e o retransmissor transmite esses pacotes pela conexão retransmissor-para-Bob (que foi iniciada por Bob); pacotes de Bob para Alice fluem pelas mesmas duas conexões de retransmissor de forma inversa. E pronto, Bob e Alice têm uma conexão simultânea fim a fim mesmo que nenhum deles possa aceitar uma sessão originada fora de sua LAN. O uso de retransmissor ilustra o projeto cada vez mais sofisticado de sistemas P2P, em que pares realizam serviços de sistema central para outros (serviço de indexação e transmissão sendo dois exemplos) ao mesmo tempo em que usam o serviço de usuário final (por exemplo, download de arquivo, telefonia IP) fornecidos pelo sistema P2P.

O Skype é uma aplicação da Internet de amplo sucesso, que atinge literalmente dezenas de milhões de usuários. A incrivelmente rápida e ampla adoção de Skype, bem como de compartilhamento de arquivos

P2P, da Web e de programas de mensagens instantâneas antes dele, é um testamento da sabedoria do projeto de arquitetura geral da Internet, um projeto que não poderia ter previsto o conjunto rico e sempre em expansão de aplicações da Internet que seria desenvolvido nos próximos 30 anos. Os serviços de rede oferecidos a aplicações de Internet — transporte de datagrama sem conexão (UDP), transferência de datagrama orientado para conexão (TCP), a interface socket, endereçamento e nomes (DNS) entre outros — provaram ser suficientes para permitir que milhares de aplicações fossem desenvolvidas. Como todas essas aplicações foram colocadas sobre as quatro camadas inferiores existentes da pilha de protocolo de Internet, eles envolvem apenas o desenvolvimento de novos softwares cliente-servidor e peer-to-peer para uso nos sistemas finais. Isso, por sua vez, permitiu que essas aplicações fossem rapidamente empregadas e adotadas.

2.7 Programação e desenvolvimento de aplicações com TCP

Agora que já examinamos várias importantes aplicações de rede, vamos explorar como são escritos programas de aplicação de rede. Nesta seção escreveremos programas de aplicações que usam TCP; na seção seguinte, escreveremos programas que usam UDP.

Lembre-se de que na Seção 2.1 dissemos que muitas aplicações de rede consistem em um par de programas — um programa cliente e um programa servidor — que residem em dois sistemas finais diferentes. Quando esses programas são executados, criam-se um processo cliente e um processo servidor, que se comunicam entre si lendo de seus sockets e escrevendo através deles. Ao criar uma aplicação de rede, a tarefa principal do programador é escrever o código tanto para o programa cliente como para o programa servidor.

Há dois tipos de aplicações de rede. Um deles é uma implementação de um protocolo padrão definido, por exemplo, em um RFC. Para essa implementação, os programas, cliente e servidor, devem obedecer às regras ditadas pelo RFC. Por exemplo, o programa cliente poderia ser uma implementação do lado do cliente do protocolo FTP descrito na Seção 2.3 e definido explicitamente no RFC 959 e o programa servidor, uma implementação do protocolo de servidor FTP também descrito explicitamente no RFC 959. Se um programador escrever codificação para o programa cliente e um outro programador independente escrever uma codificação para o programa servidor e ambos seguirem cuidadosamente as regras do RFC, então os dois programas poderão interagir. Realmente, muitas das aplicações de rede de hoje envolvem comunicação entre programas cliente e servidor que foram criados por programadores diferentes — por exemplo, um browser Netscape que se comunica com um servidor Web Apache, ou um cliente FTP em um PC que carrega um arquivo em um servidor FTP UNIX. Quando um programa, cliente ou servidor, implementa um protocolo definido em um RFC, deve usar o número de porta associado com o protocolo. (Números de portas foram discutidos brevemente na Seção 2.1. Serão examinados mais detalhadamente no Capítulo 3.)

O outro tipo de aplicação cliente-servidor é uma aplicação *proprietária*. Nesse caso, o protocolo de camada de aplicação utilizado pelos programas cliente e servidor não obedecem necessariamente a nenhum RFC existente. Um único programador (ou equipe de desenvolvimento) cria ambos os programas cliente e servidor, e tem completo controle sobre o que entra no código. Mas, como a codificação não implementa um protocolo de domínio público, outros programadores independentes não poderão desenvolver programas que interajam com a aplicação. Ao desenvolver uma aplicação proprietária, o programador deve ter o cuidado de não usar um dos números de porta bem conhecidos, definidos em RFCs.

Nesta seção e na próxima, examinaremos as questões fundamentais do desenvolvimento de uma aplicação cliente-servidor proprietária. Durante a fase de desenvolvimento, uma das primeiras decisões que o programador deve tomar é se a aplicação rodará em TCP ou UDP. Lembre-se de que o TCP é orientado para conexão e provê um *canal confiável de cadeia de bytes*, pelo qual fluem dados entre dois sistemas finais. O UDP não é orientado para conexão e envia pacotes de dados independentes de um sistema final ao outro, sem nenhuma garantia de entrega.

Nesta seção, desenvolveremos uma aplicação cliente simples que roda em TCP; na seção seguinte, desenvolveremos uma aplicação cliente simples que roda em UDP. Apresentaremos essas aplicações TCP e UDP simples em Java. Poderíamos escrevê-las em linguagem C ou C++, mas optamos por Java por diversas razões. Em primeiro

lugar, o código das aplicações é mais elegante e mais limpo em Java. Com Java, há menos linhas de codificação e cada uma delas pode ser explicada a programadores iniciantes sem muita dificuldade. Mas não precisa ficar assustado se não estiver familiarizado com a linguagem Java. Você conseguirá acompanhar a codificação se tiver experiência de programação em outra linguagem.

Para leitores interessados em programação cliente-servidor em linguagem C, há várias boas referências à disposição [Donahoo, 2001; Stevens, 1997; Frost, 1994; Kurose, 1996].

2.7.1 Programação de aplicações com TCP

Comentamos na Seção 2.1 que processos que rodam em máquinas diferentes se comunicam uns com os outros enviando mensagens para sockets. Dissemos que cada processo é análogo a uma casa e que o socket do processo é análogo a uma porta. Como ilustrado na Figura 2.28, o socket é a porta entre o processo da aplicação e o TCP. O desenvolvedor da aplicação controla tudo que está no lado da camada de aplicação da porta; contudo, tem pouco controle do lado da camada de transporte. (No máximo, poderá fixar alguns parâmetros do TCP, tais como tamanho máximo do buffer e tamanho máximo de segmentos.)

Agora, vamos examinar mais de perto a interação dos programas cliente e servidor. O cliente tem a tarefa de iniciar contato com o servidor. Para que o servidor possa reagir ao contato inicial do cliente, tem de estar pronto, o que implica duas coisas. Em primeiro lugar, o programa servidor não pode estar inativo, isto é, tem de estar rodando como um processo antes de o cliente tentar iniciar contato. Em segundo lugar, o programa tem de ter alguma porta — mais precisamente, um socket — que acolha algum contato inicial de um processo cliente que esteja rodando em uma máquina qualquer. Recorrendo à nossa analogia casa/porta para processo/socket, às vezes, nos referiremos ao contato inicial do cliente como ‘bater à porta’.

Com o processo servidor em execução, o processo cliente pode iniciar uma conexão TCP com o servidor, o que é feito no programa cliente pela criação de um socket. Quando cria seu socket, o cliente especifica o endereço do processo servidor, a saber, o endereço IP do hospedeiro servidor e o número de porta do processo servidor. Com a criação do socket no programa cliente, o TCP no cliente inicia uma apresentação de três vias e estabelece uma conexão TCP com o servidor. A apresentação de três vias é completamente transparente para os programas cliente e servidor.

Durante a apresentação de três vias, o processo cliente bate no socket de entrada do processo servidor. Quando o servidor ‘ouve’ a batida, cria uma nova porta (mais precisamente, um novo socket) dedicada àquele cliente específico. No exemplo a seguir, a porta de entrada é um objeto ServerSocket que denominamos welcomeSocket. Quando um cliente bate nesse socket, o programa chama o método accept() do welcomeSocket,

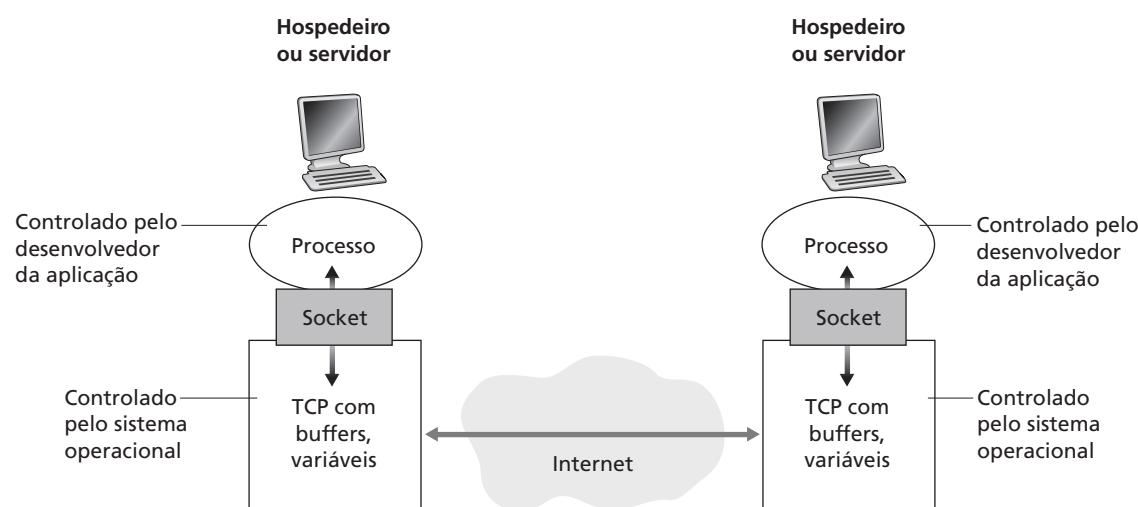


Figura 2.28 Processos que se comunicam através de sockets TCP

que cria um novo socket para o cliente. Ao final da fase de apresentação, existe uma conexão TCP entre o socket do cliente e o novo socket do servidor. Daqui em diante, vamos nos referir ao novo socket dedicado do servidor como **socket de conexão** do servidor.

Da perspectiva da aplicação, a conexão TCP é uma tubulação virtual direta entre o socket do cliente e o socket de conexão do servidor. O processo cliente pode enviar bytes para seu socket arbitrariamente; o TCP garante que o processo servidor receberá (através do socket de conexão) cada byte na ordem em que foram enviados. Assim, o TCP provê um **serviço confiável de corrente de bytes** entre os processos cliente e servidor. Além disso, exatamente como pessoas podem entrar e sair pela mesma porta, o processo cliente não somente envia bytes a seu socket, mas também os recebe dele; de modo semelhante, o processo servidor não só recebe bytes de seu socket de conexão, mas também os envia por ele. Isso é ilustrado na Figura 2.29. Como sockets desempenham um papel central em aplicações cliente-servidor, o desenvolvimento dessas aplicações também é denominado programação de sockets.

Antes de apresentarmos nosso exemplo de aplicação cliente-servidor, é útil discutirmos a noção de cadeia. Uma **cadeia** é uma sequência de caracteres que fluem para dentro ou para fora de um processo. Cada cadeia é, para o processo, uma cadeia de entrada ou uma cadeia de saída. Se a cadeia for de **entrada**, estará ligada a alguma fonte de entrada para o processo, tal como uma entrada-padrão (o teclado) ou um socket para o qual fluem dados vindos da Internet. Se a cadeia for de **saída**, estará ligada a alguma fonte de saída para o processo, tal como uma saída padrão (o monitor) ou um socket do qual fluem dados para a Internet.

2.7.2 Um exemplo de aplicação cliente-servidor em Java

Usaremos a seguinte aplicação cliente-servidor simples para demonstrar programação de sockets para TCP e UDP:

1. Um cliente lê uma linha a partir de sua **entrada padrão** (teclado) e a envia através de seu socket para o servidor.
2. O servidor lê uma linha a partir de seu socket de conexão.
3. O servidor converte a linha para letras maiúsculas.
4. O servidor envia a linha modificada ao cliente através de seu socket de conexão.
5. O cliente lê a linha modificada através de seu socket e apresenta a linha na sua **saída padrão** (monitor).

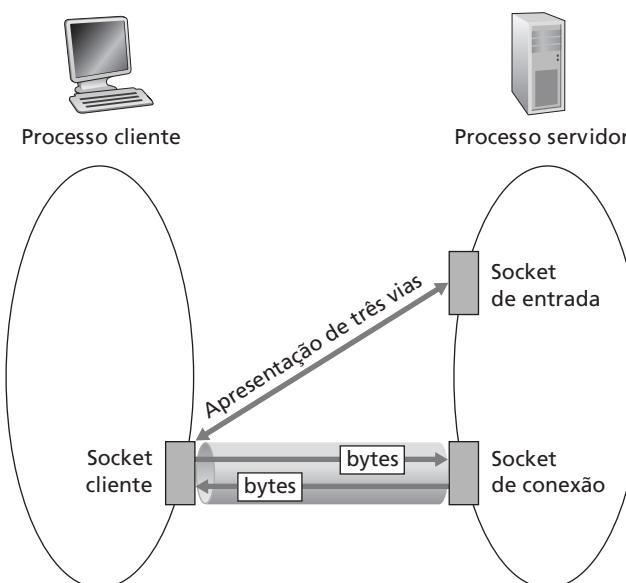


Figura 2.29 Socket cliente, socket de entrada e socket de conexão

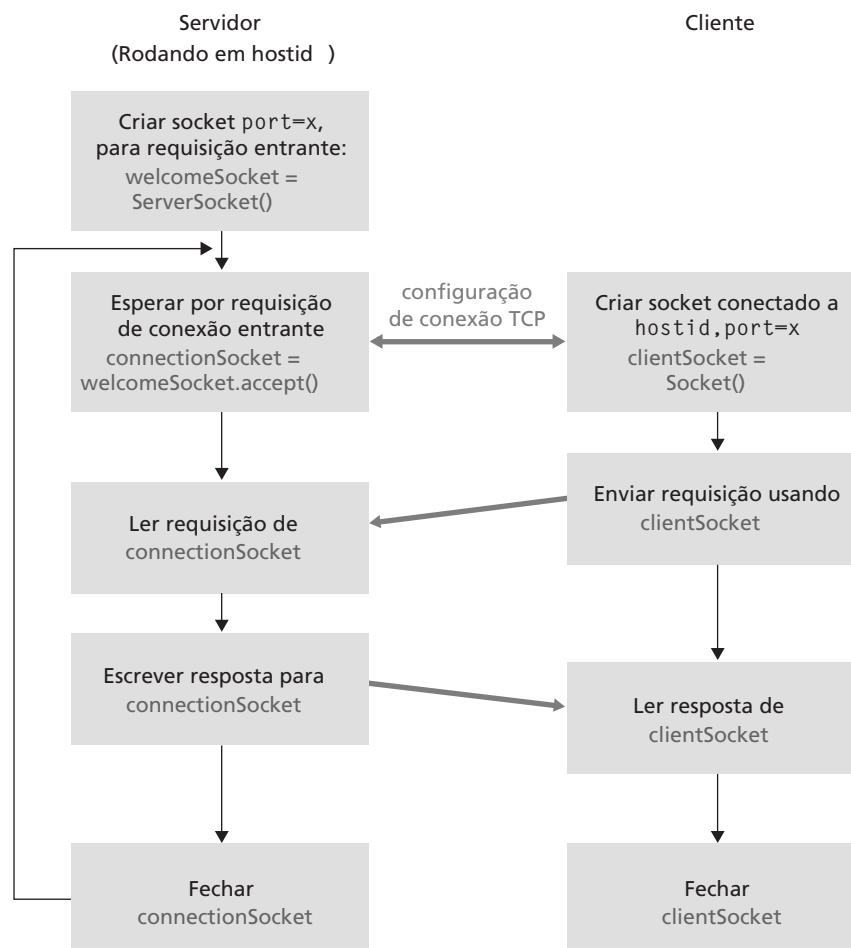


Figura 2.30 A aplicação cliente-servidor, usando serviços de transporte orientados à conexão

A Figura 2.30 ilustra a principal atividade do cliente e do servidor relacionada ao socket.

Em seguida, forneceremos o par de programas cliente-servidor para uma implementação da aplicação utilizando TCP. Apresentaremos também uma análise detalhada, linha a linha, após cada programa. O programa cliente será denominado `TCPClient.java` e o programa servidor, `TCPServer.java`. Para dar ênfase às questões fundamentais, forneceremos, intencionalmente, uma codificação objetiva, mas não tão à prova de fogo. Uma ‘boa codificação’ certamente teria algumas linhas auxiliares a mais.

Tão logo os dois programas estejam compilados em seus respectivos hospedeiros, o programa servidor será primeiramente executado no hospedeiro servidor, o que criará nele um processo servidor.

Como discutido anteriormente, o processo servidor espera para ser contatado por um processo cliente. Nesse exemplo de aplicação, quando o programa cliente é executado, um processo é criado no cliente, e esse processo imediatamente contata o servidor e estabelece uma conexão TCP com ele. O usuário no cliente pode então usar a aplicação para enviar uma linha e, em seguida, receber uma versão dessa linha em letras maiúsculas.

TCPClient.java

Eis a codificação para o lado cliente da aplicação:

```
import java.io.*;
import java.net.*;
```

```

class TCPClient {
    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser = new BufferedReader(
            new InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer = new DataOutputStream(
            clientSocket.getOutputStream());
        BufferedReader inFromServer =
            new BufferedReader(new InputStreamReader(
                clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " +
            modifiedSentence);
        clientSocket.close();
    }
}

```

O programa TCPClient cria três cadeias e um socket, como mostrado na Figura 2.31. O socket é denominado `clientSocket`. A cadeia `inFromUser` é uma cadeia de entrada para o programa ligada à entrada padrão (isto é, o teclado). Quando o usuário digita caracteres no teclado, eles fluem para dentro da cadeia `inFromUser`. A cadeia `inFromServer` é outra cadeia de entrada do programa ligada ao socket. Caracteres que chegam da rede fluem para dentro da cadeia `inFromServer`. Finalmente, a cadeia `outToServer` é uma cadeia de saída do programa que também é ligada ao socket. Caracteres que o cliente envia à rede fluem para dentro da cadeia `outToServer`.

Vamos agora examinar as várias linhas da codificação.

```

import java.io.*;
import java.net.*;

java.io e java.net são pacotes Java. O pacote java.io contém classes para cadeias de entrada e de saída. Em particular, contém as classes BufferedReader e DataOutputStream — classes que o programa usa para criar as três cadeias previamente ilustradas. O pacote java.net provê classes para suporte de rede. Em particular, contém as classes Socket e ServerSocket. O objeto clientSocket desse programa é derivado da classe Socket.

```

```

class TCPClient {
    public static void main(String argv[]) throws Exception
    {.....}
}

```

Até aqui, o que vimos é material padronizado que você vê no início da maioria das codificações Java. A terceira linha é o começo de um bloco de definição de classe. A palavra-chave `class` inicia a definição de classe para a classe denominada `TCPClient`. Uma classe contém variáveis e métodos, limitados pelas chaves `{ }` que iniciam e encerram o bloco de definição de classe. A classe `TCPClient` não tem nenhuma variável de classe e possui exatamente um método, que é o método `main()`. Métodos são semelhantes às funções ou aos procedimentos em linguagens como C; o método `main()` na linguagem Java é semelhante à função `main()` em C e C++. Quando o interpretador Java executa uma aplicação (ao ser chamado pela classe de controle da aplicação), ele começa chamando o método `main()` da classe. O método `main()` então chama todos os outros métodos exigidos para

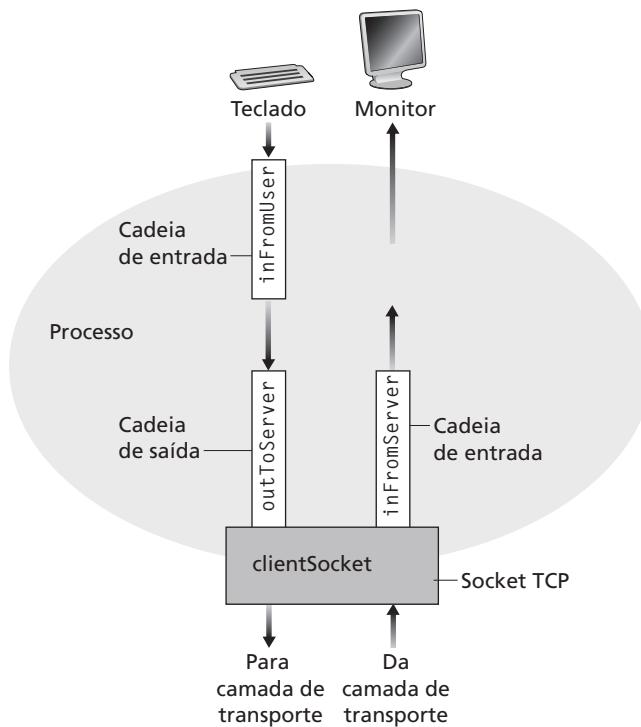


Figura 2.31 TCPClient tem três cadeias através das quais fluem caracteres

executar a aplicação. Para essa introdução à programação de portas em Java, você pode ignorar as palavras-chave `public`, `static`, `void`, `main` e `throws Exceptions` (embora deva incluí-las no código).

```
String sentence;
String modifiedSentence;
```

As duas linhas apresentadas acima declaram objetos do tipo `String` (cadeia). O objeto `sentence` é a cadeia digitada pelo usuário e enviada ao servidor. O objeto `modifiedSentence` é a cadeia obtida do servidor e enviada à saída padrão do usuário.

```
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
```

Essa linha cria o objeto de cadeia `inFromUser` do tipo `BufferedReader`. A cadeia de entrada é inicializada com `System.in`, que vincula a cadeia à entrada padrão. O comando permite que o cliente leia o texto de seu teclado.

```
Socket clientSocket = new Socket("hostname", 6789);
```

Essa linha cria o objeto `clientSocket` do tipo `Socket`. Ela também ativa a conexão TCP entre cliente e servidor. A cadeia `host-name` deve ser substituída pelo nome de hospedeiro do servidor (por exemplo, `apple.poly.edu`). Antes de a conexão TCP ser realmente iniciada, o cliente realiza um procedimento de consulta ao DNS do nome de hospedeiro para obter o endereço IP da máquina. O número 6789 é o número de porta. Você pode usar um número de porta diferente, mas precisa obrigatoriamente usar o mesmo do lado servidor da aplicação. Como discutimos anteriormente, o processo servidor é identificado pelo endereço IP do hospedeiro juntamente com o número de porta da aplicação.

```
DataOutputStream outToServer =
    new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer =
    new BufferedReader(new InputStreamReader(
        clientSocket.getInputStream()));
```

Essas linhas criam objetos de cadeia que são ligados ao socket. A cadeia `outToServer` fornece a saída do processo para o socket. A cadeia `inFromServer` fornece ao processo a entrada do socket (veja a Figura 2.31).

```
sentence = inFromUser.readLine();
```

Essa linha coloca uma linha digitada pelo usuário na cadeia `sentence`. A cadeia `sentence` continua a juntar caracteres até que o usuário termine a linha digitando ‘carriage return’. A linha passa da entrada padrão para dentro da cadeia `sentence` por meio da cadeia `inFromUser`.

```
outToServer.writeBytes(sentence + '\n');
```

Esta linha envia a cadeia `sentence` para dentro da corrente `outToServer` ampliada com um ‘carriage return’. A sentença ampliada flui pelo socket do cliente para dentro da conexão TCP. O cliente então espera para receber caracteres do servidor.

```
modifiedSentence = inFromServer.readLine();
```

Quando chegam do servidor, os caracteres fluem através da cadeia `inFromServer` e são colocados dentro da cadeia `modifiedSentence`. Continuam a se acumular em `modifiedSentence` até que a linha termine com um caractere de ‘carriage return’.

```
System.out.println("FROM SERVER " + modifiedSentence);
```

Essa linha envia para o monitor a cadeia `modifiedSentence` retornada pelo servidor.

```
clientSocket.close();
```

Essa última linha fecha o socket e, por conseguinte, a conexão TCP entre o cliente e o servidor. Ela faz com que o TCP no cliente envie uma mensagem para o TCP no servidor (veja a Seção 3.5).

TCPserver.java

Agora vamos examinar o programa servidor.

```
import java.io.*;
import java.net.*;
class TCPServer {
    public static void main(String argv[]) throws Exception
    {
        String clientSentence;
        String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket
            (6789);
        while(true) {
            Socket connectionSocket = welcomeSocket.
                accept();
            BufferedReader inFromClient =
                new BufferedReader(new InputStreamReader(
                    connectionSocket.getInputStream()));
            DataOutputStream outToClient =
                new DataOutputStream(
                    connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();
            capitalizedSentence =
                clientSentence.toUpperCase() + '\n';
            outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```



TCPServer tem muitas semelhanças com TCPClient. Vamos agora dar uma olhada nas linhas em TCPServer.java. Não comentaremos as linhas que são idênticas ou semelhantes aos comandos em TCPClient.java.

A primeira linha em TCPServer é substancialmente diferente da que vimos em TCPClient:

```
ServerSocket welcomeSocket = new ServerSocket(6789);
```

Essa linha cria o objeto welcomeSocket, que é do tipo ServerSocket. O welcomeSocket é uma espécie de porta que fica ‘ouvindo’, à espera que algum cliente bata. O welcomeSocket fica à escuta na porta número 6789. A linha seguinte é:

```
Socket connectionSocket = welcomeSocket.accept();
```

Essa linha cria um novo socket, denominado connectionSocket, quando algum cliente bate à porta welcomeSocket. O número de porta desse socket também é 6789. (Explicaremos por que ambos têm o mesmo número de porta no Capítulo 3). O TCP então estabelece uma conexão virtual direta entre clientSocket no cliente e connectionSocket no servidor. O cliente e o servidor podem então enviar bytes um para o outro pela conexão, e todos os bytes enviados chegam ao outro lado na ordem certa. Com connectionSocket estabelecido, o servidor pode continuar à escuta por outros clientes que requisitarão a aplicação usando welcomeSocket. (Essa versão do programa, na verdade, não fica à escuta por mais requisições de conexão, mas pode ser modificada com threads para fazer isso.) O programa então cria diversos objetos de cadeia, análogos aos objetos de cadeia em clientSocket. Considere agora:

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Esse comando é o coração da aplicação. Ele toma a linha enviada pelo cliente, passa todas as letras para maiúsculas e adiciona um ‘carriage return’. Ele usa o método toUpperCase(). Todos os outros comandos no programa são periféricos, usados para a comunicação com o cliente.

Para testar o par de programas, instale e compile TCPClient.java em um hospedeiro e TCPServer.java em outro hospedeiro. Não se esqueça de incluir o nome de hospedeiro do servidor adequado em TCPClient.java. Então, execute TCPServer.class, o programa servidor compilado, no servidor. Isso cria um processo no servidor que fica ocioso até ser contatado por algum cliente. Então, execute TCPClient.class, o programa cliente compilado, no cliente. Isso cria um processo no cliente e estabelece uma conexão TCP entre os processos cliente e servidor. Por fim, para usar a aplicação, digite uma sentença seguida de um ‘carriage return’.

Para desenvolver sua própria aplicação cliente-servidor, você pode começar modificando ligeiramente os programas. Por exemplo, em vez de converter todas as letras para maiúsculas, o servidor poderia contar o número de vezes que a letra ‘s’ aparece no texto e retornar esse número.

2.8 Programação de aplicações com UDP

Aprendemos na seção anterior que, quando dois processos se comunicam por TCP, é como se houvesse uma tubulação entre eles, que permanece ativa até que um dos dois processos a feche. Quando um dos processos quer enviar alguns bytes para o outro processo, simplesmente insere os bytes na tubulação. O processo de envio não tem de acrescentar um endereço de destino aos bytes porque a tubulação está ligada logicamente ao destino. Além disso, a tubulação provê um canal confiável de cadeia de bytes — a sequência de bytes recebida pelo processo receptor é exatamente a mesma que o remetente inseriu na tubulação.

O UDP também permite que dois (ou mais) processos que rodam em hospedeiros diferentes se comuniquem. Contudo, é diferente do TCP de muitas maneiras fundamentais. Primeiramente, é um serviço não orientado para conexão — não há uma fase inicial de apresentação, durante a qual é estabelecida uma tubulação entre os dois processos. Como o UDP não tem uma tubulação, quando um processo quer enviar um conjunto de bytes a outro, o processo remetente deve anexar o endereço do processo destinatário ao conjunto de bytes. E isso precisa ser feito para cada conjunto de bytes que o processo remetente enviar. Como analogia, considere um grupo de 20 pessoas que toma cinco táxis para um mesmo destino; quando cada um dos grupos entra em um carro, tem de informar separadamente ao motorista o endereço de destino. Assim, o UDP é semelhante a um serviço de táxi. O endereço de destino é uma tupla que consiste no endereço IP do hospedeiro destinatário e no número de porta do

processo destinatário. Vamos nos referir ao conjunto de bytes de informação, juntamente com o endereço IP do destinatário e o número de porta, como ‘pacote’. O UDP provê um modelo de serviço não confiável orientado para mensagem, no sentido de que faz o melhor esforço para entregar o conjunto de bytes ao destino. Esses conjuntos são bytes enviados em uma única operação na parte remetente, e serão entregues como um conjunto na parte destinatária; isso contrasta com o sentido da cadeia de bytes do TCP. O serviço UDP faz o melhor esforço já que o UDP não garante que o conjunto de bytes será realmente entregue. O serviço UDP contrasta acentuadamente (em vários aspectos) com o modelo de serviço confiável de cadeia de bytes do TCP.

Após ter criado um pacote, o processo remetente empurra-o para dentro da rede através de um socket. Continuando com nossa analogia do táxi, do outro lado do socket remetente há um táxi esperando pelo pacote. Esse táxi, então, leva o pacote até seu endereço de destino. Contudo, não garante que finalmente entregará o pacote no seu destino final, pois pode quebrar ou sofrer algum outro problema não previsto no caminho. Em outras palavras, o UDP *provê serviço não confiável de transporte a seus processos de comunicação* — não dá nenhuma garantia de que um datagrama ('pacote') alcançará seu destino final.

Nesta seção, ilustraremos programação de socket desenvolvendo novamente a mesma aplicação da seção anterior, mas, dessa vez, com UDP. Veremos que a codificação para UDP é diferente da codificação para TCP de muitas maneiras importantes. Em particular, (1) não há apresentação inicial entre os dois processos e, portanto, não há necessidade de um socket de entrada, (2) não há cadeias ligadas aos sockets, (3) os hospedeiros remetentes criam pacotes anexando o endereço IP do destinatário e o número de porta a cada conjunto de bytes que enviam e (4) o processo destinatário deve desmontar cada pacote recebido para obter os bytes de informação do pacote.

Lembre-se, mais uma vez, da nossa aplicação simples:

1. Um cliente lê uma linha a partir de sua entrada padrão (teclado) e a envia através de seu socket para o servidor.
2. O servidor lê uma linha a partir de seu socket.
3. O servidor converte a linha para letras maiúsculas.
4. O servidor envia a linha modificada através de seu socket ao cliente.
5. O cliente lê a linha modificada através de seu socket e apresenta a linha em sua saída padrão (monitor).

A Figura 2.32 destaca a principal atividade relacionada ao socket realizada pelo cliente e pelo servidor, que se comunicam por meio de um serviço de transporte (UDP) não orientado para conexão.

UDPClient.java

Eis a codificação para o lado cliente da aplicação:

```
import java.io.*;
import java.net.*;
class UDPCClient {
    public static void main(String args[]) throws Exception
    {
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader
                (System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress =
            InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket =
            new DatagramPacket(sendData, sendData.length,
                IPAddress, 54321);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
        clientSocket.receive(receivePacket);
        String reply = new String(receiveData);
        System.out.println("Echoed back to you: " + reply);
        clientSocket.close();
    }
}
```

```

        new DatagramPacket(sendData, sendData.length,
                           IPAddress, 9876);
    clientSocket.send(sendPacket);
    DatagramPacket receivePacket =
        new DatagramPacket(receiveData,
                           receiveData.length);
    clientSocket.receive(receivePacket);
    String modifiedSentence =
        new String(receivePacket.getData());
    System.out.println("FROM SERVER:" +
                       modifiedSentence);
    clientSocket.close();
}
}

```

O programa UDPClient.java constrói uma cadeia e um socket, como mostra a Figura 2.33. O socket é denominado `clientSocket` e é do tipo `DatagramSocket`. Note que o UDP usa, no cliente, um tipo de socket diferente do usado no TCP. Em particular, com UDP nosso cliente usa um `DatagramSocket`, enquanto com TCP ele usou um `Socket`. A cadeia `inFromUser` é uma cadeia de entrada para o programa; está ligada à entrada padrão, isto é, ao teclado. Tínhamos uma cadeia equivalente em nossa versão TCP do programa. Quando o usuário digita caracteres no teclado, esses caracteres fluem para dentro da cadeia `inFromUser`. Porém, ao contrário do TCP, não há cadeias (de entrada e de saída) ligadas ao socket. Em vez de alimentar bytes à cadeia ligada a um objeto `Socket`, o UDP transmite pacotes individuais por meio do objeto `DatagramSocket`.

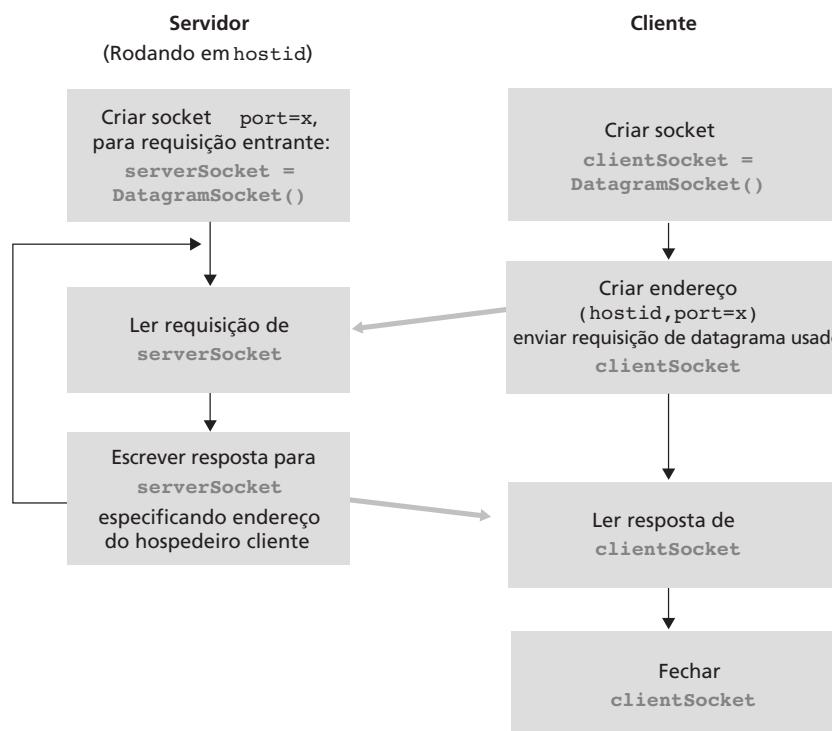


Figura 2.32 A aplicação cliente-servidor usando serviços de transporte não orientados para conexão

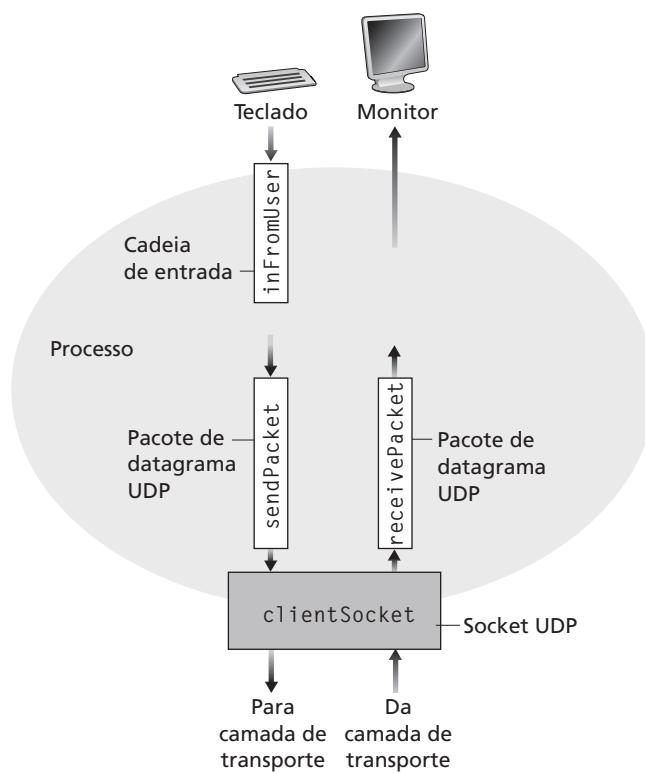


Figura 2.33 UDPClient tem uma corrente; o socket aceita pacotes do processo e entrega pacotes ao processo

Agora, vamos examinar as linhas da codificação de `TCPClient.java` que são significativamente diferentes.

```
DatagramSocket clientSocket = new DatagramSocket();
```

Essa linha cria o objeto `clientSocket` do tipo `DatagramSocket`. Ao contrário do `TCPClient.java`, ela não ativa uma conexão TCP. Em particular, o hospedeiro cliente não contata o hospedeiro servidor durante a execução desta linha. Por essa razão, o construtor `DatagramSocket()` não toma como argumento o nome do servidor ou o número da porta. Usando nossa analogia porta-tubulação, a execução da linha acima cria um socket para o processo cliente, mas não cria uma tubulação entre os dois processos.

```
InetAddress IPAddress = InetAddress.getByName("hostname");
```

Para enviar bytes a um processo destinatário, precisamos do endereço do processo. Parte desse endereço é o endereço IP do hospedeiro destinatário. A linha apresentada invoca uma consulta ao DNS que traduz o nome do destinatário (nesse exemplo, fornecido na codificação pelo programador) para um endereço IP. O DNS também foi chamado pela versão TCP do cliente, embora o tenha feito de maneira implícita, e não explícita. O método `getByName()` toma como argumento o nome de hospedeiro do servidor e retorna o endereço IP desse mesmo servidor. Coloca esse endereço no objeto `IPAddress` do tipo `InetAddress`.

```
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
```

Os vetores de bytes `sendData` e `receiveData` reterão os dados que o cliente envia e recebe, respectivamente.

```
sendData = sentence.getBytes();
```

Essa linha realiza, essencialmente, uma conversão de tipo. Pega a cadeia `sentence` e a renomeia como `sendData`, que é um vetor de bytes.

```
DatagramPacket sendPacket = new DatagramPacket(
    sendData, sendData.length, IPAddress, 9876);
```

Essa linha constrói o pacote, `sendPacket`, que o cliente enviará para a rede através do seu socket. Esse pacote inclui os dados contidos nele, `sendData`, o comprimento desses dados, o endereço IP do servidor e o número de porta da aplicação (escolhemos 9876). Note que `sendPacket` é do tipo `DatagramPacket`.

```
clientSocket.send(sendPacket);
```

Nessa linha, o método `send()` do objeto `clientSocket` toma o pacote recém-construído e passa-o para a rede através de `clientSocket`. Mais uma vez, note que o UDP envia a linha de caracteres de maneira muito diferente do TCP. O TCP simplesmente inseriu a linha de caracteres em uma cadeia que tinha uma conexão lógica direta com o servidor; o UDP cria um pacote que inclui o endereço do servidor. Após enviar esse pacote, o cliente espera para receber um pacote do servidor.

```
DatagramPacket receivePacket =  
    new DatagramPacket (receiveData, receiveData.length);
```

Nessa linha, enquanto espera pelo pacote do servidor, o cliente cria um lugar reservado para o pacote, `receivePacket`, um objeto do tipo `DatagramPacket`.

```
clientSocket.receive(receivePacket);
```

O cliente fica ocioso até receber um pacote; quando enfim o recebe, ele o coloca em `receivePacket`.

```
String modifiedSentence =  
    new String(receivePacket.getData());
```

Essa linha extrai os dados de `receivePacket` e realiza uma conversão de tipo, convertendo um vetor de bytes na cadeia `modifiedSentence`.

```
System.out.println("FROM SERVER:" + modifiedSentence);
```

Essa linha, que também está presente no `TCPClient`, apresenta a cadeia `modifiedSentence` no monitor do cliente.

```
clientSocket.close();
```

Essa última linha fecha o socket. Como o UDP não é orientado para conexão, esta linha não faz com que o cliente envie uma mensagem de camada de transporte ao servidor (ao contrário do `TCPClient`).

UDPServer.java

Vamos agora dar uma olhada no lado servidor da aplicação:

```
import java.io.*;  
import java.net.*;  
class UDPServer {  
    public static void main(String args[]) throws Exception  
    {  
        DatagramSocket serverSocket = new  
            DatagramSocket(9876);  
        byte[] receiveData = new byte[1024];  
        byte[] sendData = new byte[1024];  
        while (true)  
        {  
            DatagramPacket receivePacket =  
                new DatagramPacket(receiveData,  
                    receiveData.length);  
            serverSocket.receive(receivePacket);  
            String sentence = new String(  
                receivePacket.getData());  
            InetAddress IPAddress =  
                receivePacket.getAddress();
```

```

        int port = receivePacket.getPort();
        String capitalizedSentence =
            sentence.toUpperCase();
        sendData = capitalizedSentence.getBytes();
        DatagramPacket sendPacket =
            new DatagramPacket (sendData,
                sendData.length, IPAddress, port);
        serverSocket.send(sendPacket);
    }
}
}
}

```

O programa `UDPServer.java` constrói um socket, como mostra a Figura 2.34. O socket é denominado `serverSocket`. É um objeto do tipo `DatagramSocket`, como era o socket do lado cliente da aplicação. Mais uma vez, nenhuma cadeia está ligada ao socket.

Vamos agora examinar as linhas da codificação que são diferentes de `TCPServer.java`.

```
DatagramSocket serverSocket = new DatagramSocket (9876);
```

Essa linha constrói o `DatagramSocket` `serverSocket` na porta 9876. Todos os dados enviados e recebidos passarão através desse socket. Como o UDP não é orientado para conexão, não temos de criar um novo socket e continuar à escuta de novas requisições de conexão, como é feito no `TCPServer.java`. Se vários clientes acessarem essa aplicação, todos enviarão seus pacotes por esse único socket, `serverSocket`.

```

String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();

```

Essas três linhas desmontam o pacote que chega do cliente. A primeira delas extraí os dados do pacote e os coloca no `String sentence`; há uma linha análoga em `UDPClient`. A segunda linha extraí o endereço IP. A terceira linha extraí o número de porta do cliente, que é escolhido pelo cliente e é diferente do número de porta 9876 do servidor. (Discutiremos números de porta do cliente com mais detalhes no capítulo seguinte.) É necessário que o servidor obtenha o endereço (endereço IP e número de porta) do cliente para que possa enviar a sentença em letras maiúsculas de volta para ele.

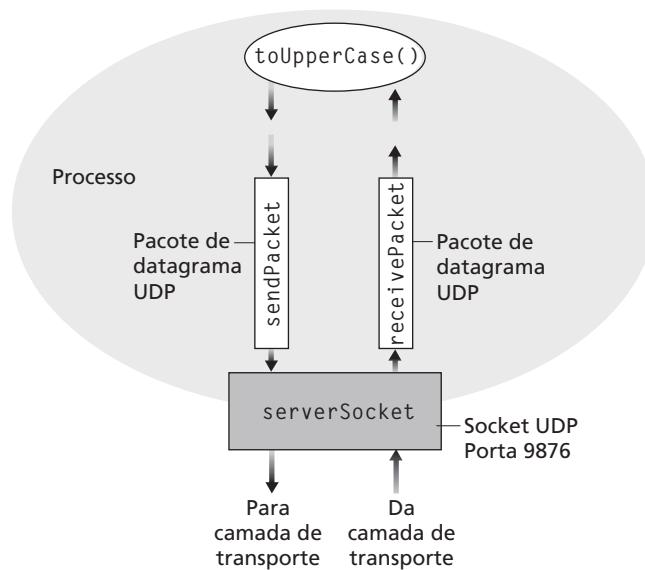


Figura 2.34 UDPServer não tem Cadeias; o socket aceita pacotes do processo e entrega pacotes ao processo



Isso conclui nossa análise sobre o par de programas UDP. Para testar a aplicação, instale e compile `UDPClient.java` em um hospedeiro e `UDPServer.java` em outro. (Não esqueça de incluir o nome apropriado do hospedeiro do servidor em `UDPClient.java`.) Em seguida, execute os dois programas em seus respectivos hospedeiros. Ao contrário do TCP, você pode executar primeiramente o lado cliente e depois o lado servidor. Isso acontece porque, quando você executa o programa cliente, o processo cliente não tenta iniciar uma conexão com o servidor. Assim que tiver executado os programas cliente e servidor, você poderá usar a aplicação digitando uma linha no cliente.

2.9 Resumo

Neste capítulo, estudamos os aspectos conceituais e os aspectos de implementação de aplicações de rede. Conhecemos a onipresente arquitetura cliente-servidor adotada por aplicações da Internet e examinamos sua utilização nos protocolos HTTP, FTP, SMTP, POP3 e DNS. Analisamos esses importantes protocolos de camada de aplicação e suas aplicações associadas (Web, transferência de arquivos, e-mail e DNS) com algum detalhe. Conhecemos também a arquitetura P2P, cada vez mais dominante, e examinamos sua utilização em muitas aplicações. Vimos como o API socket pode ser usado para construir aplicações de rede. Examinamos a utilização de portas para serviços de transporte fim a fim orientados para conexão (TCP) e não orientados para conexão (UDP) e também construímos um servidor Web simples usando sockets. A primeira etapa de nossa jornada de descida pela arquitetura das camadas da rede está concluída!

Logo no começo deste livro, na Seção 1.1, demos uma definição um tanto vaga e despojada de um protocolo. Dissemos que um protocolo é ‘o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento’. O material deste capítulo — em particular, o estudo detalhado dos protocolos HTTP, FTP, SMTP, POP3 e DNS — agregou considerável substância a essa definição. Protocolos são o conceito fundamental de redes. Nossa estudo sobre protocolos de aplicação nos deu agora a oportunidade de desenvolver uma noção mais intuitiva do que eles realmente são.

Na Seção 2.1, descrevemos os modelos de serviço que o TCP e o UDP oferecem às aplicações que os chamam. Examinamos esses modelos de serviço ainda mais de perto quando desenvolvemos, nas seções 2.7 e 2.8, aplicações simples que executam em TCP e UDP. Contudo, pouco dissemos sobre como o TCP e o UDP fornecem esses modelos de serviços. Por exemplo, sabemos que o TCP provê um serviço de dados confiável, mas ainda não mencionamos como ele o faz. No próximo capítulo, examinaremos cuidadosamente não apenas o que são protocolos de transporte, mas também o como e o porquê deles.

Agora que conhecemos a estrutura da aplicação da Internet e os protocolos de camada de aplicação, estamos prontos para continuar a descer a pilha de protocolos e examinar a camada de transporte no Capítulo 3.



Exercícios de fixação

Capítulo 2 Questões de revisão

Seção 2.1

1. Relacione cinco aplicações da Internet não proprietárias e os protocolos de camada de aplicação que elas usam.
2. Qual é a diferença entre arquitetura de rede e arquitetura de aplicação?
3. Para uma sessão de comunicação entre um par de processos, qual processo é o cliente e qual é o servidor?
4. Em uma aplicação de compartilhamento de arquivos P2P, você concorda com a afirmação: “não existe nenhuma noção de lados cliente e servidor de uma sessão de comunicação”? Por quê?
5. Que informação é usada por um processo que está rodando em um hospedeiro para identificar um processo que está rodando em outro hospedeiro?

6. Suponha que você queria fazer uma transação de um cliente remoto para um servidor da maneira mais rápida possível. Você usaria o UDP ou o TCP? Por quê?
7. Com referência à Figura 2.4, vemos que nenhuma das aplicações relacionadas nela requer ‘sem perda de dados’ e ‘temporização’. Você consegue imaginar uma aplicação que requeira ‘sem perda de dados’ e seja também altamente sensível ao atraso?
8. Relacione quatro classes de serviços que um protocolo de transporte pode prover. Para cada uma delas, indique se o UDP ou o TCP (ou ambos) fornece tal serviço.
9. Lembre-se de que o TCP pode ser aprimorado com o SSL para fornecer serviços de segurança processo a processo, incluindo a decodificação. O SSL opera na camada de transporte ou na camada de aplicação? Se o desenvolvedor da aplicação quer que o TCP seja aprimorado com o SSL, o que ele deve fazer?

Seções 2.2 a 2.5

10. O que significa protocolo de apresentação (handshaking protocol)?
11. Por que HTTP, FTP, SMTP, POP3 rodam sobre TCP e não sobre UDP?
12. Considere um site de comércio eletrônico que quer manter um registro de compras para cada um de seus clientes. Descreva como isso pode ser feito com cookies.
13. Descreva como o cache Web pode reduzir o atraso na recepção de um objeto desejado. O cache Web reduzirá o atraso para todos os objetos requisitados por um usuário ou somente para alguns objetos? Por quê?
14. Digite um comando Telnet em um servidor Web e envie uma mensagem de requisição com várias linhas. Inclua nessa mensagem a linha de cabeçalho `If-modified-since:` para forçar uma mensagem de resposta com a codificação de estado 304 `Not Modified`.
15. Por que se diz que o FTP envia informações de controle ‘fora da banda’?
16. Suponha que Alice envie uma mensagem a Bob por meio de uma conta de e-mail da Web (como o Hotmail), e que Bob acesse seu e-mail por seu servidor de correio usando POP3. Descreva como a mensagem vai do hospedeiro de Alice até o hospedeiro de Bob. Não se esqueça de relacionar a série de protocolos de camada de aplicação usados para movimentar a mensagem entre os dois hospedeiros.
17. Imprima o cabeçalho de uma mensagem de e-mail que acabou de receber. Quantas linhas de cabeçalho `Received:` há nela? Analise cada uma das linhas.

18. Da perspectiva de um usuário, qual é a diferença entre o modo ler-e-apagar e o modo ler-e-guardar no POP3?
19. É possível que o servidor Web e o servidor de correio de uma organização tenham exatamente o mesmo apelido para um nome de hospedeiro (por exemplo, `foo.com`)? Qual seria o tipo de RR que contém o nome de hospedeiro do servidor de correio?

Seção 2.6

20. No BitTorrent, suponha que Alice forneça blocos para Bob durante um intervalo de 30 segundos. Bob retornará, necessariamente, o favor e fornecerá blocos para Alice no mesmo intervalo? Por quê?
21. Considere um novo par, Alice, que entra no BitTorrent sem possuir nenhum bloco. Sem qualquer bloco, ela não pode se tornar uma das quatro melhores exportadoras de dados para qualquer um dos outros pares, visto que ela não possui nada para enviar. Então, como Alice obterá seu primeiro bloco?
22. O que é uma rede de sobreposição em um sistema de compartilhamento de arquivos P2P? Ela inclui roteadores? O que são as arestas da rede de sobreposição? Como a rede de sobreposição de inundação de consultas é criada e como é mantida?
23. De que modo a aplicação mensagem instantânea é um híbrido das arquiteturas cliente-servidor e P2P?
24. Considere um DHT com uma topologia da rede de sobreposição (ou seja, cada par rastreia todos os pares no sistema). Quais são as vantagens e desvantagens de um DHT circular (sem atalhos)?
25. O Skype utiliza técnicas P2P para duas funções importantes. Quais são elas?
26. Relacione quatro diferentes aplicações que são apropriadas naturalmente para arquiteturas P2P. (Dica: Distribuição de arquivo e mensagem instantânea são duas.)

Seções 2.7-2.8

27. O servidor UDP descrito na Seção 2.8 precisava de uma porta apenas, ao passo que o servidor TCP descrito na Seção 2.7 precisava de duas portas. Por quê? Se um servidor TCP tivesse de suportar n conexões simultâneas, cada uma de um hospedeiro cliente diferente, de quantas portas precisaria?
28. Para a aplicação cliente-servidor por TCP descrita na Seção 2.7, por que o programa servidor deve ser executado antes do programa cliente? Para a aplicação cliente-servidor por UDP descrita na Seção 2.8, por que o programa cliente pode ser executado antes do programa servidor?



Problemas

1. Falso ou verdadeiro?
 - a. Um usuário requisita uma página Web que consiste em texto e três imagens. Para essa página, o cliente enviará uma mensagem de requisição e receberá quatro mensagens de resposta.
 - b. Duas páginas Web distintas (por exemplo, www.mit.edu/research.html e www.mit.edu/students.html) podem ser enviadas pela mesma conexão persistente.
 - c. Com conexões não persistentes entre browser e servidor de origem, é possível que um único segmento TCP transporte duas mensagens distintas de requisição HTTP.
 - d. O cabeçalho Date: na mensagem de resposta HTTP indica a última vez que o objeto da resposta foi modificado.
 - e. As mensagens de resposta HTTP nunca possuem um corpo de mensagem vazio.
2. Leia o RFC 959 para FTP. Relacione todos os comandos de cliente que são suportados pelo RFC.
3. Considere um cliente HTTP que queira obter um documento Web em um dado URL. Inicialmente, o endereço IP do servidor HTTP é desconhecido. Nesse cenário, quais protocolos de transporte e de camada de aplicação são necessários, além do HTTP?
4. Considere a seguinte cadeia de caracteres ASCII capturada pelo Wireshark quando o browser enviou uma mensagem HTTP GET (ou seja, o conteúdo real de uma mensagem HTTP GET). Os caracteres <cr><lf> são retorno de carro e avanço de linha (ou seja, a cadeia de caractere em itálico <cr> no texto abaixo representa o caractere único retorno de carro que estava contido, naquele momento, no cabeçalho HTTP). Responda às seguintes questões, indicando onde está a resposta na mensagem HTTP GET abaixo.


```
GET /cs453/index.html HTTP/1.1
<cr><lf>Host: gaia.cs.umass.edu<cr><lf>User Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept: text/xml, application/xml, application/xhtml+xml, text/html; q=0.8, text/plain; q=0.8, image/png, */*; q=0.5 <cr><lf>Accept-Language: en-us, en; q=0.5<cr><lf>Accept-Encoding: zip, deflate<cr><lf>Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7 <cr><lf>Keep-Alive: 300<cr><lf>Connection: keep-alive
<cr><lf><cr><lf>
```
5.
 - a. Qual é a URL do documento requisitado pelo browser?
 - b. Qual versão do HTTP o browser está rodando?
 - c. O browser requisita uma conexão não persistente ou persistente?
 - d. Qual é o endereço IP do hospedeiro no qual o browser está rodando?
 - e. Que tipo de browser inicia essa mensagem? Por que é necessário o tipo de browser em uma mensagem de requisição HTTP?

O texto a seguir mostra a resposta enviada do servidor em reação à mensagem HTTP GET na questão acima. Responda às seguintes questões, indicando onde está a resposta na mensagem abaixo.

```
HTTP/1.1 200 OK<cr><lf>Date:Tue, 07 Mar 2008
12:39:45 GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>Keep-Alive: timeout=max=100<cr><lf>Connection: Keep-Alive<cr><lf>Content-Type: text/html; charset=ISO-8859-1<cr><lf><cr><lf><!doctype html public "-//w3c//dtd html 4.0 transitional//en"><lf><html><lf><head><lf><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"><lf><meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape]"><lf><title>CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>

```

 - a. O servidor foi capaz de encontrar o documento com sucesso ou não? A que horas foi apresentada a resposta do documento?
 - b. Quando o documento foi modificado pela última vez?
 - c. Quantos bytes existem no documento que está retornando?

- d.** Quais são os 5 primeiros bytes do documento que está retornando? O servidor aceitou uma conexão persistente?
- 6.** Obtenha a especificação HTTP/1.1 (RFC 2616). Responda às seguintes perguntas:
- Explique o mecanismo de sinalização que cliente e servidor utilizam para indicar que uma conexão persistente está sendo fechada. O cliente, o servidor, ou ambos, podem sinalizar o encerramento de uma conexão?
 - Que serviços de criptografia são providos pelo HTTP?
 - O cliente é capaz de abrir três ou mais conexões simultâneas com um determinado servidor?
 - Um servidor ou um cliente pode abrir uma conexão de transporte entre eles se um dos dois descobrir que a conexão ficou lenta por um tempo. É possível que um lado comece a encerrar a conexão enquanto o outro está transmitindo dados por meio dessa conexão? Explique.
- 7.** Suponha que você clique com seu browser Web sobre um ponteiro para obter uma página Web e que o endereço IP para o URL associado não esteja no cache de seu hospedeiro local. Portanto, será necessária uma consulta ao DNS para obter o endereço IP. Considere que n servidores DNS sejam visitados antes que seu hospedeiro receba o endereço IP do DNS; as visitas sucessivas incorrem em um RTT igual a RTT_1, \dots, RTT_n . Suponha ainda que a página Web associada ao ponteiro contenha exatamente um objeto que consiste em uma pequena quantidade de texto HTML. Seja RTT_0 o RTT entre o hospedeiro local e o servidor que contém o objeto. Admitindo que o tempo de transmissão do objeto seja zero, quanto tempo passará desde que o cliente clica o ponteiro até que receba o objeto?
- 8.** Com referência ao problema 7, suponha que o arquivo HTML referece três objetos muito pequenos no mesmo servidor. Desprezando tempos de transmissão, quanto tempo passa, usando-se:
- HTTP não persistente sem conexões TCP paralelas?
 - HTTP não persistente com o browser configurado para 5 conexões paralelas?
 - HTTP persistente?
- 9.** Considere a Figura 2.12, que mostra uma rede institucional conectada à Internet. Suponha que o tamanho médio do objeto seja 850 mil bits e que a taxa média de requisição dos browsers da instituição aos servidores de origem seja 1,6 requisição por segundo. Suponha também que a quantidade de tempo que leva desde o instante em que o roteador do lado da Internet do enlace de acesso transmite uma requisição HTTP até que receba a resposta seja 3 segundos em média (veja Seção 2.2.5). Modele o tempo total médio de resposta como a soma do atraso de acesso médio (isto é, o atraso entre o roteador da Internet e o roteador da instituição) e o tempo médio de atraso da Internet. Para a média de atraso de acesso, use $\Delta(1 - \Delta\beta)$, onde Δ é o tempo médio requerido para enviar um objeto pelo enlace de acesso e β é a taxa de chegada de objetos ao enlace de acesso.
- Determine o tempo total médio de resposta.
 - Agora, considere que um cache é instalado na LAN institucional e que a taxa de resposta local seja 0,4. Determine o tempo total de resposta.
- 10.** Considere um enlace curto de 10 metros através do qual um remetente pode transmitir a uma taxa de 150 bits/s em ambas as direções. Suponha que os pacotes com dados tenham 100 mil bits de comprimento, e os pacotes que contêm controle (por exemplo, ACK ou apresentação) tenham 200 bits de comprimento. Admita que N conexões paralelas recebam cada $1/N$ da largura de banda do enlace. Agora, considere o protocolo HTTP e suponha que cada objeto baixado tenha 100 Kbits de comprimento e que o objeto inicial baixado contenha 10 objetos referenciados do mesmo remetente. Os downloads paralelos por meio de instâncias paralelas de HTTP não persistente fazem sentido nesse caso? Agora considere o HTTP persistente. Você espera ganhos significativos sobre o caso não persistente? Justifique sua resposta.
- 11.** Considere o cenário apresentado na questão anterior. Agora suponha que o enlace é compartilhado por Bob e mais quatro usuários. Bob usa instâncias paralelas de HTTP não persistente, e os outros quatro usuários usam HTTP não persistente sem downloads paralelos.
- As conexões paralelas de Bob ajudam a acessar páginas Web mais rapidamente? Por quê? Por que não?
 - Se cinco usuários abrirem cinco instâncias paralelas de HTTP não persistente, então as conexões paralelas de Bob ainda seriam úteis? Por quê? Por que não?
- 12.** Escreva um programa TCP simples para um servidor que aceite linhas de entrada de um cliente e envie as linhas para a saída padrão do servidor. (Você pode fazer isso modificando o programa TCPServer.java no texto.) Compile e execute seu programa. Em qualquer outra máquina que contenha um browser Web, defina o servidor proxy no browser para a máquina que está executando seu programa servidor e também configure o número de porta adequadamente. Seu browser deverá agora enviar suas mensagens.

gens de requisição GET a seu servidor, e este deverá apresentar as mensagens em sua saída padrão. Use essa plataforma para determinar se seu browser gera mensagens GET condicionais para objetos que estão em caches locais.

13. Qual é a diferença entre MAIL FROM: em SMTP e FROM: na mensagem de correio?
14. Como o SMTP marca o final de um corpo de mensagem? E o HTTP? O HTTP pode usar o mesmo método que o SMTP para marcar o fim de um corpo de mensagem? Explique.
15. Leia o RFC 5321 para SMTP. O que significa MTA? Considere a seguinte mensagem spam recebida (modificada de um spam verdadeiro). Admitindo que o criador desse spam seja malicioso e que os hospedeiros sejam honestos, identifique o hospedeiro malicioso que criou essa mensagem spam.

```
From - Fri Nov 07 13:41:30 2008
Return-Path: tenniss@pp33head.com
Received: from barmail.cs.umass.edu
(barmail.cs.umass.edu [128.119.240.3])
by cs.umass.edu
(8.13.1/8.12.6) for hg@cs.umass.edu;
Fri, 7 Nov 2008
13:27:10 -0500
Received: from asusus-4b96 (localhost
[127.0.0.1]) by barmail.cs.umass.edu
(Spam Firewall) for
<hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:
07 -0500 (EST)
Received: from asusus-4b96 ([58.88.21.177]) by
barmail.cs.umass.edu for <hg@cs.umass.edu>; Fri,
07 Nov 2008 13:27:07 -0500 (EST)
Received: from [58.88.21.177] by
Inbnd55.exchangeddd.com; Sat, 8 Nov
2008 01:27:07 +0700
From: "Jonny"<tennis5@pp33head.com>
To: <hg@cs.umass.edu>
Subject: How to secure your savings
```

16. Leia o RFC do POP3 [RFC 1939]. Qual é a finalidade do comando UIDL do POP3?
17. Considere acessar seu e-mail com POP3.

- a. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-apagar. Conclua a seguinte transação:

```
C: list
S: 1 498
S: 2 912
S: .
```

```
C: retr 1
S: blah blah ...
S: .....blah
S: .
?
?
```

- b. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-guardar. Conclua a seguinte transação:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: .....blah
S: .
?
?
```

- c. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-guardar. Usando sua solução na parte (b), suponha que você recupere as mensagens 1 e 2, saia do POP e então, 5 minutos mais tarde, acesse novamente o POP para obter um novo e-mail. Imagine que nenhuma outra mensagem foi enviada nesse intervalo. Elabore um transcript dessa segunda sessão POP.

18. a. O que é um banco de dados whois?
b. Use vários bancos de dados whois da Internet para obter os nomes de dois servidores DNS. Cite quais bancos de dados whois você utilizou.
c. Use nslookup em seu hospedeiro local para enviar consultas DNS a três servidores de nomes: seu servidor DNS local e os dois servidores DNS que encontrou na parte (b). Tente consultar registros dos tipos A, NS e MX. Faça um resumo do que encontrou.
d. Use nslookup para encontrar um servidor Web que tenha vários endereços IP. O servidor Web de sua instituição (escola ou empresa) tem vários endereços IP?
e. Use o banco de dados whois ARIN para determinar a faixa de endereços IP usados por sua universidade.
f. Descreva como um invasor pode usar bancos de dados whois e a ferramenta nslookup para fazer o reconhecimento de uma instituição antes de lançar um ataque.
g. Discuta por que bancos de dados whois devem estar disponíveis publicamente.

- 19.** Neste problema, utilizamos a ferramenta funcional *dig* disponível em hospedeiros Unix e Linux para explorar a hierarquia dos servidores DNS. Lembre de que, na Figura 2.21, um servidor DNS de nível superior na hierarquia do DNS delega uma consulta DNS para um servidor DNS de nível inferior na hierarquia enviando de volta ao cliente DNS o nome daquele servidor DNS de nível inferior. Em primeiro lugar, leia a *man page* sobre a ferramenta *dig* e responda às seguintes questões:
- Iniciando com o servidor DNS raiz (de um dos servidores raiz [a-m].root-servers.net), construa uma sequência de consultas para o endereço IP para seu servidor Web de departamento utilizando o *dig*. Mostre a relação de nomes de servidores DNS na cadeia de delegação ao responder à sua consulta.
 - Repita o item “a” com vários sites da Internet populares, como google.com, yahoo.com ou amazon.com.
- 20.** Suponha que você consiga acessar os caches nos servidores DNS locais do seu departamento. Você é capaz de propor uma maneira de determinar, em linhas gerais, os servidores Web (fora de seu departamento) que são mais populares entre os usuários do seu departamento? Explique.
- 21.** Suponha que seu departamento possua um servidor DNS local para todos os computadores do departamento. Você é um usuário comum (ou seja, não é um administrador de rede/sistema). Você consegue encontrar um modo de determinar se um site da Internet externo foi muito provavelmente acessado de um computador do seu departamento alguns segundos atrás? Explique.
- 22.** Considere um arquivo de distribuição de $F = 15$ Gbits para N pares. O servidor possui um taxa de upload de $u_s = 30$ Mbps e cada par possui uma taxa de download de $d_i = 2$ Mbps e uma taxa de upload de u . Para $N = 10, 100$ e 1.000 e $u = 300$ Kbps, 700 Kbps e 2 Mbps, prepare um gráfico apresentando o tempo mínimo de distribuição para cada uma das combinações de N e u para o modo cliente-servidor e para o modo distribuição P2P.
- 23.** Considere distribuir um arquivo de F bits para N pares utilizando uma arquitetura cliente-servidor. Admita um modelo fluido no qual o servidor pode transmitir simultaneamente para diversos pares, a diferentes taxas, desde que a taxa combinada não ultrapasse u_s .
- Suponha que $u_s/N \leq d_{\min}$. Especifique um esquema de distribuição que possua o tempo de distribuição de NF/u_s .
 - Suponha que $u_s/N \geq d_{\min}$. Especifique um esquema de distribuição que possua o tempo de distribuição de F/d_{\min} .
 - Conclua que o tempo mínimo de distribuição é, geralmente, dado por $\max\{NF/u_s, F/d_{\min}\}$.
- 24.** Considere distribuir um arquivo de F bits para N pares utilizando uma arquitetura P2P. Admita um modelo fluido e que d_{\min} é muito grande, de modo que a largura de banda do download do par nunca é um gargalo.
- Suponha que $us \leq (u_s + u_1 + \dots + u_N)/N$. Especifique um esquema de distribuição que possua o tempo de distribuição de F/u_s .
 - Suponha que $u_s \geq (u_s + u_1 + \dots + u_N)/N$. Especifique um esquema de distribuição que possua o tempo de distribuição de $NF/(u_s + u_1 + \dots + u_N)$.
 - Conclua que o tempo mínimo de distribuição é, geralmente, dado por $\max\{F/u_s, NF/(u_s + u_1 + \dots + u_N)\}$.
- 25.** Considere uma rede de sobreposição com N pares ativos, sendo que cada dupla de pares possua uma conexão TCP. Além disso, suponha que as conexões TCP passem por um total de M roteadores. Quantos nós e arestas há na rede de sobreposição correspondente?
- 26.** Suponha que Bob tenha entrado no BitTorrent, mas ele não quer fazer o upload de nenhum dado para qualquer outro par (denominado carona).
- Bob alega que consegue receber uma cópia completa do arquivo compartilhado pelo grupo. A alegação de Bob é possível? Por quê?
 - Bob alega que ele pode “pegar carona” de um modo mais eficiente usando um conjunto de diversos computadores (com endereços IP distintos) no laboratório de informática de seu departamento. Como ele pode fazer isso?
- 27.** Neste problema, queremos descobrir a eficiência de um sistema de compartilhamento de arquivo P2P semelhante ao BitTorrent. Considere dois pares, Bob e Alice. Eles entram em um torrent com M pares no total (incluindo Bob e Alice) que estão compartilhando um arquivo que consiste em N blocos. Admita que em um tempo específico t , os blocos que um par possui são escolhidos aleatoriamente a partir de todos os N blocos, e nenhum par possui todos os N blocos. Responda às seguintes questões:
- Qual é a probabilidade de Bob ter todos os blocos de Alice, sabendo que os números de blocos que Bob e Alice têm são representados por n_b e n_a ?
 - Remova parte do condicionamento do item “a” para descobrir a probabilidade de Bob ter os mesmos blocos de Alice, sabendo que Alice possui n_a blocos.
 - Suponha que cada par no BitTorrent tenha 5 vizinhos. Qual é a probabilidade de Bob ter dados que sejam de interesse de pelo menos um dos cinco vizinhos?

28. No exemplo de DHT circular na Seção 2.6.2, suponha que o par 3 descobriu que o par 5 saiu. Como o par 3 atualiza informações sobre o estado de seu sucessor?
29. No exemplo de DHT circular na Seção 2.6.2, suponha que um novo par 6 queira entrar no DHT, sabendo, inicialmente, o endereço IP do par 15. Quais passos são tomados?
30. Considere um DHT circular com identificadores de nós e de chave na faixa [0, 63]. Suponha que haja oito pares com identificadores 0, 8, 16, 24, 32, 40, 48 e 56.
- Suponha que cada par possa ter um par no atalho. Para cada um dos oito atalhos, determine seu par no atalho para que o número de mensagens enviadas para qualquer consulta (iniciando em qualquer par) seja reduzido.
 - Repita o item “a”, mas permita que cada par tenha dois pares no atalho.
31. Como um número inteiro em $[0, 2^n - 1]$ pode ser expresso como um número binário de n bit em um DHT, cada chave pode ser expressa como $k = (k_0, k_1, \dots, k_{n-1})$, e cada identificador de par pode ser expresso como $p = (p_0, p_1, \dots, p_{n-1})$. Vamos, agora, definir a distância XOR entre a chave k e o par p como

$$d(k, p) = \sum_{j=0}^{n-1} |k_j - p_j| 2^j$$

Descreva como essa métrica pode ser usada para determinar duplas (chave, valor) para pares. (Para aprender mais sobre como construir um DHT eficiente usando essa métrica natural, consulte [Maymounkov, 2002], no qual o DHT Kademlia é descrito.)

32. Considere uma versão generalizada do esquema descrito no problema acima. Em vez de usar números binários, vamos tratar os identificadores de chave e de par como números de base b , sendo $b > 2$, e

então usamos a métrica do problema anterior para criar um DHT (com 2 substituído b). Compare esse DHT baseado nos números de base b com o DHT baseado nos números binários. Na pior das hipóteses, qual DHT gera mais mensagens por consulta? Por quê?

33. Como os DHTs são redes de sobreposição, eles não se adequam necessariamente bem à rede física de sobreposição no sentido de que dois pares vizinhos podem estar fisicamente muito distantes; por exemplo, um par poderia estar na Ásia e seu vizinho, na América do Norte. Se atribuirmos identificadores aleatória e uniformemente para pares recém-unidos, esse esquema de atribuição causaria essa incompatibilidade? Explique. E como tal incompatibilidade afetaria o desempenho do DHT?
34. Instale e compile os programas Java TCPClient e UDPClient em um hospedeiro e TCPServer e UDPServer em outro.
- Suponha que você execute TCPClient antes de executar TCPServer. O que acontece? Por quê?
 - Imagine que você execute UDPClient antes de UDPServer. O que acontece? Por quê?
 - O que acontece se você usar números de porta diferentes para os lados cliente e servidor?

35. Suponha que, em UDPClient.java, a linha
`DatagramSocket clientSocket = new DatagramSocket();`
 seja substituída por
`DatagramSocket clientSocket = new DatagramSocket(5432);`
 Será necessário mudar UDPServer.java? Quais são os números de porta para os sockets em UDPClient e UDPServer? Quais eram esses números antes dessa mudança?

Questões dissertativas

- Na sua opinião, por que as aplicações de compartilhamento de arquivos P2P são tão populares? Será porque distribuem música e vídeo gratuitamente (o que é legalmente discutível) ou porque seu número imenso de servidores atende eficientemente uma demanda maciça por megabytes? Ou será pelas duas razões?
- Leia o artigo “The Darknet and the Future of Content Distribution” de Biddle, England, Peinado e Willman [Biddle, 2003]. Você concorda com a opinião dos autores? Por quê? Por que não?
- Sites de comércio eletrônico e outros sites Web frequentemente têm bancos de dados “de apoio”. Como servidores HTTP se comunicam com esses bancos de dados?
- Como você pode configurar seu browser para cache local? Que opções de cache você tem?
- Você pode configurar seu browser para abrir várias conexões simultâneas com um site Web? Quais são as vantagens e as desvantagens de ter um grande número de conexões TCP simultâneas?

6. Vimos que sockets TCP da Internet tratam os dados que estão sendo enviados como uma cadeia de bytes, mas que sockets UDP reconhecem fronteiras de mensagens. Cite uma vantagem e uma desvantagem da API orientada para bytes em relação à API que reconhece e preserva explicitamente as fronteiras das mensagens definidas por aplicações.
7. O que é o servidor Web Apache? Quanto custa? Que funcionalidade tem atualmente?
8. Muitos clientes BitTorrent utilizam DHTs para criar um rastreador distribuído. Para esses DHTs, qual é a “chave” e qual é o “valor”?
9. Imagine que as organizações responsáveis pela padronização da Web decidam modificar a convenção de nomeação de modo que cada objeto seja nomeado e referenciado por um nome exclusivo que independa de localização (um URN). Discuta algumas questões que envolveriam tal modificação.
10. Há empresas distribuindo transmissões televisivas ao vivo por meio da Internet hoje? Se sim, essas empresas estão usando arquiteturas cliente-servidor e P2P?
11. As empresas, hoje, estão oferecendo um serviço de vídeo ao vivo através da Internet usando uma arquitetura P2P?
12. Como o Skype provê um serviço PC para telefone a vários países de destino?
13. Quais são os clientes mais populares do BitTorrent atualmente?

Tarefas de programação de sockets

Tarefa 1: servidor Web multithread

Ao final desta tarefa de programação, você terá desenvolvido, em Java, um servidor Web multithread, que seja capaz de atender várias requisições em paralelo. Você implementará a versão 1.0 do HTTP como definida no RFC 1945.

O HTTP/1.0 cria uma conexão TCP separada para cada par requisição/resposta. Cada uma dessas conexões será manipulada por um thread. Haverá também um thread principal, no qual o servidor ficará à escuta de clientes que quiserem estabelecer conexões. Para simplificar o trabalho de programação, desenvolveremos a codificação em dois estágios. No primeiro estágio, você escreverá um servidor multithread que simplesmente apresenta o conteúdo da mensagem de requisição HTTP que recebe. Depois que esse programa estiver executando normalmente, você adicionará a codificação necessária para gerar uma resposta apropriada.

Ao desenvolver a codificação, você poderá testar seu servidor com um browser Web. Mas lembre-se de que você não estará atendendo através da porta padrão 80, portanto, precisará especificar o número de porta dentro do URL que der a seu browser. Por exemplo, se o nome de seu hospedeiro for `host.someschool.edu`, seu servidor estiver à escuta na porta 6789 e você quiser obter o arquivo `index.html`, então deverá especificar o seguinte URL dentro do browser:

`http://host.someschool.edu:6789/index.html`

Quando seu servidor encontrar um erro, deverá enviar uma mensagem de resposta com uma fonte HTML adequada, de modo que a informação de erro

seja apresentada na janela do browser. Você pode encontrar mais detalhes sobre esta tarefa, assim como trechos importantes em código java no site <http://www.aw.com/kurose.br>.

Tarefa 2: cliente de correio

Nesta tarefa, você desenvolverá um agente de usuário de correio em Java com as seguintes características:

 Que provê uma interface gráfica para o remetente com campos para o servidor de correio local, para o endereço de e-mail do remetente, para o endereço de e-mail do destinatário, para o assunto da mensagem e para a própria mensagem.

 Que estabelece uma conexão TCP entre o cliente de correio e o servidor de correio local. Que envia comandos SMTP para o servidor de correio local. Que recebe e processa comandos SMTP do servidor de correio local.

Esta será a aparência de sua interface:



Você desenvolverá o agente de usuário de modo que ele envie uma mensagem de e-mail para no máximo um destinatário por vez. Além disso, o agente de usuário admitirá que a parte de domínio do endereço de e-mail do destinatário será o nome canônico do servidor SMTP do destinatário. (O agente de usuário não realizará uma busca no DNS para um registro MX; portanto, o remetente deverá fornecer o nome real do servidor de correio.)

Tarefa 3: UDP Pinger Lab

Neste laboratório, você implementará um cliente e um servidor Ping simples, em UDP. A funcionalidade que esses programas oferecem é similar à do programa Ping padronizado, disponível em sistemas operacionais modernos. O Ping padronizado funciona enviando à Internet uma mensagem ECHO do Protocolo de Mensagens de Controle da Internet (ICMP), que a máquina remota devolve ao remetente como se fosse um eco. Então, o remetente pode

determinar o tempo de viagem de ida e volta entre ele mesmo e o computador para o qual enviou o Ping.

O Java não provê nenhuma funcionalidade para enviar ou receber mensagens ICMP e é por isso que neste laboratório você implementará o Ping na camada de aplicação com sockets e mensagens UDP padronizados.

Tarefa 4: servidor Proxy Web

Neste laboratório, você desenvolverá um servidor proxy Web simples que também poderá fazer cache de páginas Web. Esse servidor aceitará uma mensagem GET de um browser, transmitirá essa mensagem ao servidor Web destinatário, receberá a mensagem de resposta HTTP do servidor destinatário e transmitirá a mensagem de resposta ao browser. Esse servidor proxy é muito simples: entende apenas requisições GET simples. Contudo, pode manipular todos os tipos de objetos; não somente páginas HTML, mas também imagens.

Wireshark Labs

Você encontrará detalhes completos, em inglês, sobre os Wireshark Labs no site www.aw.com/kurose_br.

Wireshark Lab: HTTP

Como já tivemos uma primeira experiência com o analisador de pacotes Wireshark no Lab 1, estamos prontos para usar o Wireshark para investigar protocolos em operação. Neste laboratório, exploraremos diversos aspectos do protocolo HTTP: a interação básica GET/resposta, formatos de mensagens HTTP, extração de grandes arquivos HTML, extração de arquivos HTML com URLs inseridos, conexões persistentes e não persistentes e autenticação e segurança do HTTP.

Wireshark Lab: DNS

Neste laboratório, examinaremos mais de perto o lado cliente do DNS, o protocolo que transforma nomes

de hospedeiros da Internet em endereços IP. Lembre-se de que, na Seção 2.5, dissemos que o papel desempenhado pelo cliente no DNS é relativamente simples — um cliente envia uma consulta a seu servidor de nomes local e recebe uma resposta. Mas há muita coisa que pode acontecer nos bastidores e que é invisível para os clientes DNS enquanto os servidores DNS hierárquicos se comunicam uns com os outros para resolver a consulta DNS do cliente de modo recursivo ou iterativo. Do ponto de vista do cliente DNS, todavia, o protocolo é bastante simples — é formulada uma consulta ao servidor de nomes local e é recebida uma resposta deste. Nesse laboratório observamos o DNS em ação.

Entrevista

Bram Cohen

Bram Cohen é Cientista-chefe e cofundador da BitTorrent, Inc. e criador do protocolo de distribuição de arquivo P2P BitTorrent. Bram é também o cofundador da CodeCon e coautor do Codeville. Antes de criar o BitTorrent, Bram trabalhou no MojoNation, que permitia que as pessoas separassem arquivos confidenciais em blocos codificados e distribuíssem essas partes para outros computadores que rodavam o software MojoNation. Esse conceito serviu de inspiração para Bram desenvolver o BitTorrent. Antes do MojoNation, Bram era um grande profissional da área de informática, trabalhando para várias empresas de Internet do meio ao fim dos anos 90. Ele cresceu em Nova York, se formou na Stuyvesant High School e frequentou a Universidade de Buffalo.



Como surgiu a ideia de desenvolver o BitTorrent?

Eu adquiri experiência criando redes (protocolos acima TCP/UDP), e implementar enxames parecia o problema irresoluto mais interessante da época, então resolvi trabalhar nisso.

O cálculo por trás do núcleo do BitTorrent é simples: Existe muita capacidade para upload lá fora. Muitas outras pessoas também fizeram essa mesma observação. Mas implementar algo que pudesse lidar com a logística envolvida é um outro problema.

Quais foram os aspectos mais desafiadores ao desenvolver o BitTorrent?

A parte principal foi acertar todo o projeto e a estrutura do protocolo. Assim que estivesse funcionando, sua implementação era “uma mera questão de programação”. Com relação à implementação, de longe a parte mais difícil foi implementar um sistema confiável. Ao lidar com pares desconhecidos, você tem de imaginar que, a qualquer momento, qualquer um deles pode fazer qualquer coisa e ter um tipo de resposta estabelecida para todos os casos extremos. Eu tinha de ficar reescrevendo uma longa seção sobre o BitTorrent quando o estava criando pela primeira vez porque apareceram novos problemas e o projeto geral se tornou mais claro.

Inicialmente, como as pessoas descobriram o BitTorrent?

Normalmente as pessoas descobrem o BitTorrent fazendo o seu download. Elas queriam um certo tipo de conteúdo, que era encontrado somente usando o BitTorrent, então o baixavam. Um editor resolveu usar o BitTorrent porque simplesmente não tinha a largura de banda para distribuir seu conteúdo de outra maneira.

Comente sua opinião sobre as ações legais da RIAA e da MPAA contra as pessoas que utilizam programas de compartilhamento de arquivo, como o BitTorrent, para dividir filmes e música? Você já foi processado por desenvolver tecnologias que distribuem ilegalmente material protegido por direitos autorais?

A violação dos direitos autorais é ilegal, mas a tecnologia, não. Nunca fui processado, pois nunca me envolvi em nenhuma violação dos direitos autorais. Se você tem interesse em produzir tecnologia, precisa se agarrar a ela.

Você acha que, em um futuro próximo, outros sistemas de distribuição de arquivo podem substituir o BitTorrent? Por exemplo, a Microsoft pode incluir seu próprio protocolo de distribuição de arquivo em um lançamento futuro de um sistema operacional?

Pode haver protocolos no futuro, mas é improvável que os princípios fundamentais de como agrupar dados, esclarecidos no protocolo BitTorrent, mudem. A maneira mais provável de ocorrer uma substituição é se houver uma mudança em toda a estrutura da Internet em razão das proporções entre algumas das constantes fundamentais que mudam radicalmente à medida que a velocidade aumenta. Mas projeções para os próximos anos apenas reforçam ainda mais o modelo atual.

De forma geral, onde você vê a Internet liderando? Na sua opinião, quais são, ou serão, os desafios técnicos mais importantes? Você visualiza alguma nova “aplicação inovadora” que está por vir?

A Internet e os computadores em geral estão se tornando cada vez mais onipresentes. O iPod nano parece uma lembrancinha de festa, pois inevitavelmente um dia ele o será, em razão da baixa dos preços. O desafio técnico atual mais interessante é reunir o maior número de dados possível de todos os dispositivos conectados e disponibilizar esses dados de uma forma útil e acessível. Por exemplo, quase todos os aparelhos portáteis poderiam conter um GPS, e cada objeto que você possui, incluindo roupas, brinquedos, eletrodomésticos e mobília, poderiam informá-lo onde se encontram ao perdê-los e fazer um resumo completo sobre seu histórico atual, incluindo manutenção necessária, utilidade futura esperada, detecção de maus tratos etc. Você não só poderia obter informações sobre suas propriedades como também, digamos, a vida útil de um produto específico poderia ser determinada de forma precisa, e a coordenação com outras pessoas se tornaria mais fácil, além do simples, mas surpreendente, aperfeiçoamento de as pessoas poderem se encontrar facilmente quando possuem um telefone celular.

Profissionalmente, alguém lhe serviu de inspiração? Em que sentidos?

Nenhuma parábola me vem à cabeça, mas os mitos do Vale do Silício foram algo que segui à risca.



Você tem algum conselho para os alunos ingressantes na área de rede/Internet?

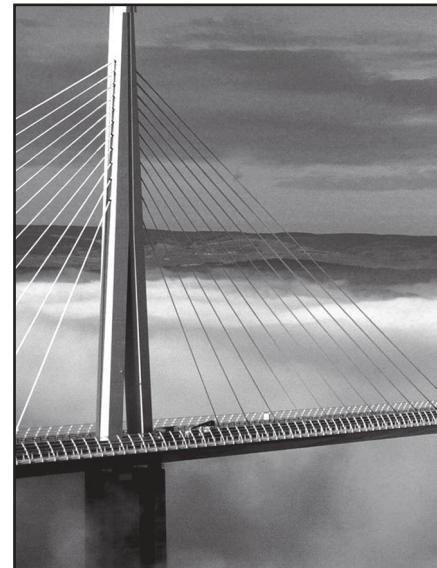
Encontre algo que não esteja em alta no momento, mas que você ache que possam dar origem a coisas empolgantes e que seja, particularmente, interessante para você e comece a trabalhar nisso. Tente também obter experiência profissional na área em que deseja trabalhar. Experiências do mundo real ensinam você o que é importante no mundo real, e isso é algo que é sempre bem distorcido quando visto somente de dentro do ambiente acadêmico.





Capítulo 3

Camada de transporte



Posicionada entre as camadas de aplicação e de rede, a camada de transporte é uma peça central da arquitetura de rede em camadas. Ela desempenha o papel fundamental de fornecer serviços de comunicação diretamente aos processos de aplicação que rodam em hospedeiros diferentes. A abordagem pedagógica que adotamos neste capítulo é alternar entre discussões de princípios de camada de transporte e o modo como esses princípios são implementados em protocolos existentes; como de costume, daremos particular ênfase aos protocolos da Internet, em especial aos protocolos de camada de transporte TCP e UDP.

Começaremos discutindo a relação entre as camadas de transporte e de rede, preparando o cenário para o exame da primeira função importante da camada de transporte — ampliar o serviço de entrega da camada de rede entre dois sistemas finais para um serviço de entrega entre dois processos de camada de aplicação que rodam nos sistemas finais. Ilustraremos essa função quando abordarmos o UDP, o protocolo de transporte não orientado para conexão da Internet.

Então retornaremos aos princípios e enfrentaremos um dos problemas mais fundamentais de redes de computadores — como duas entidades podem se comunicar de maneira confiável por um meio que pode perder e corromper dados. Mediante uma série de cenários cada vez mais complicados (e realistas!), construiremos um conjunto de técnicas que os protocolos de transporte utilizam para resolver esse problema. Então, mostraremos como esses princípios estão incorporados no TCP, o protocolo de transporte orientado para conexão da Internet.

Em seguida, passaremos para um segundo problema fundamentalmente importante em redes — o controle da taxa de transmissão de entidades de camada de transporte para evitar ou se recuperar de congestionamentos dentro da rede. Consideraremos as causas e consequências do congestionamento, bem como técnicas de controle de congestionamento comumente usadas. Após adquirir um sólido entendimento das questões que estão por trás do controle de congestionamento, estudaremos como o TCP o aborda.

3.1 Introdução e serviços de camada de transporte

Nos dois capítulos anteriores, citamos o papel da camada de transporte e os serviços que ela fornece. Vamos revisar rapidamente o que já aprendemos sobre a camada de transporte.

Um protocolo de camada de transporte fornece **comunicação lógica** entre processos de aplicação que rodam em hospedeiros diferentes. *Comunicação lógica* nesse contexto significa que, do ponto de vista de uma aplicação, tudo se passa como se os hospedeiros que rodam os processos estivessem conectados diretamente; na

verdade, esses hospedeiros poderão estar em lados opostos do planeta, conectados por numerosos roteadores e uma ampla variedade de tipos de enlace. Processos de aplicação usam a comunicação lógica provida pela camada de transporte para enviar mensagens entre si, livres da preocupação dos detalhes da infraestrutura física utilizada para transportar essas mensagens. A Figura 3.1 ilustra a noção de comunicação lógica.

Nela, vemos que protocolos de camada de transporte são implementados nos sistemas finais, mas não em roteadores de rede. No lado remetente, a camada de transporte converte as mensagens que recebe de um

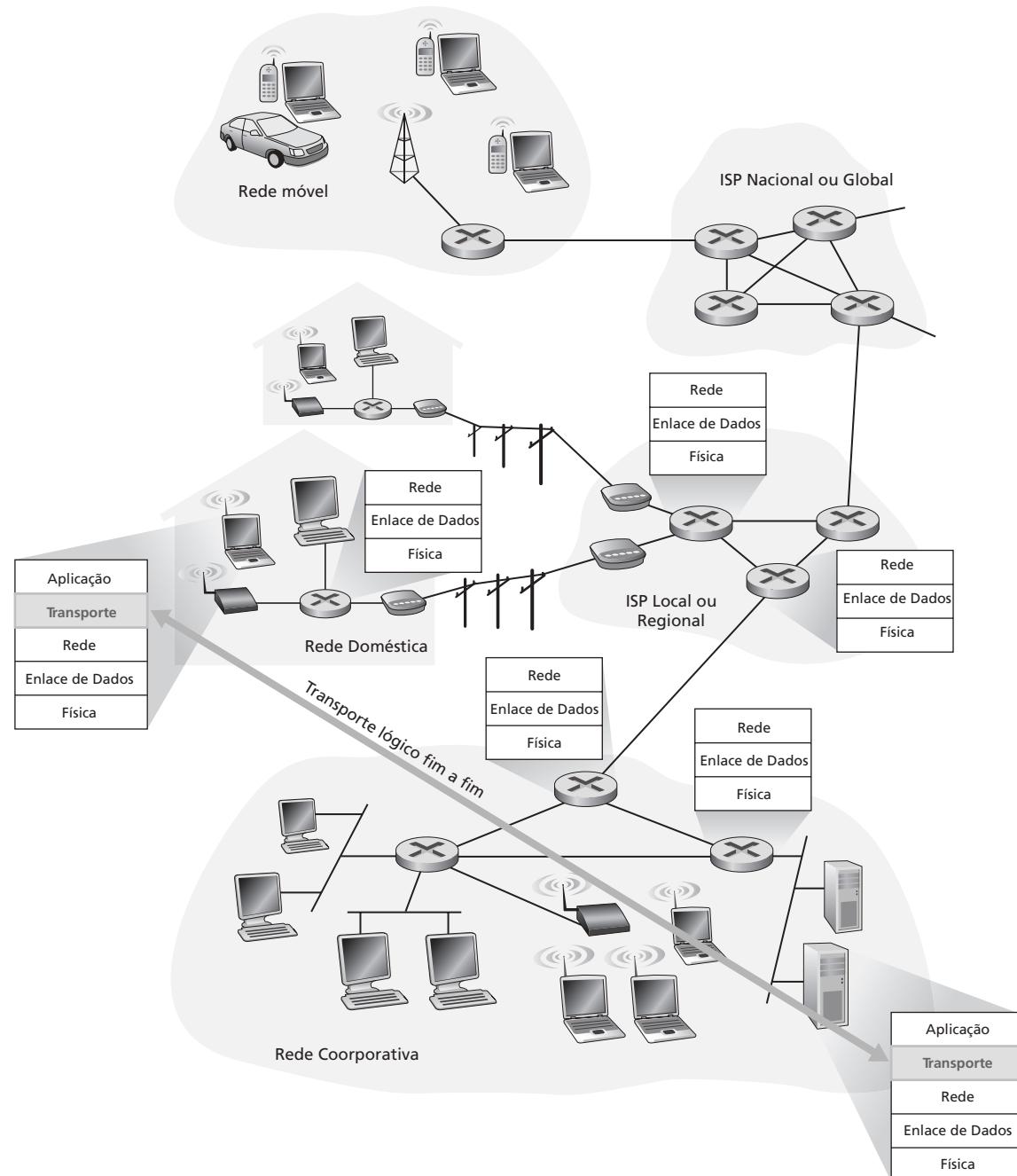


Figura 3.1 A camada de transporte fornece comunicação lógica, e não física, entre processos de aplicações. Pode haver mais de um protocolo de camada de transporte disponível para aplicações de rede. Por exemplo, a Internet tem dois protocolos — TCP e UDP. Cada um deles provê um conjunto diferente de serviços de camada de transporte à aplicação que o está chamando

processo de aplicação remetente em pacotes de camada de transporte, denominados **segmentos** de camada de transporte na terminologia da Internet. Isso é (possivelmente) feito fragmentando-se as mensagens da aplicação em pedaços menores e adicionando-se um cabeçalho de camada de transporte a cada pedaço para criar o segmento de camada de transporte. A camada de transporte, então, passa o segmento para a camada de rede no sistema final remetente, onde ele é encapsulado em um pacote de camada de rede (um datagrama) e enviado ao destinatário. É importante notar que roteadores de rede agem somente nos campos de camada de rede do datagrama; isto é, não examinam os campos do segmento de camada de transporte encapsulado com o datagrama. No lado destinatário, a camada de rede extraí do datagrama o segmento de camada de transporte e passa-o para a camada de transporte. Em seguida, essa camada processa o segmento recebido, disponibilizando os dados para a aplicação destinatária.

3.1.1 Relação entre as camadas de transporte e de rede

Lembre-se de que a camada de transporte se situa logo acima da camada de rede na pilha de protocolos. Enquanto um protocolo de camada de transporte fornece comunicação lógica entre *processos* que rodam em hospedeiros diferentes, um protocolo de camada de rede fornece comunicação lógica entre *hospedeiros*. Essa distinção é sutil, mas importante. Vamos examiná-la com o auxílio de uma analogia com moradias.

Considere duas casas, uma na Costa Leste e outra na Costa Oeste dos Estados Unidos e que, em cada uma delas, há uma dúzia de crianças. As crianças da Costa Leste são primas das crianças da Costa Oeste e todas adoram escrever cartas umas para as outras — cada criança escreve a cada primo uma vez por semana e cada carta é entregue pelo serviço de correio tradicional dentro de um envelope separado. Assim, cada moradia envia 144 cartas por semana para a outra. (Essas crianças economizariam muito dinheiro se tivessem e-mail!) Em cada moradia há uma criança responsável pela coleta e distribuição da correspondência — Ann, na casa da Costa Oeste, e Bill, na da Costa Leste. Toda semana, Ann coleta a correspondência de seus irmãos e irmãs e a coloca no correio. Quando as cartas chegam à casa da Costa Oeste, também é Ann quem tem a tarefa de distribuir a correspondência a seus irmãos e irmãs. Bill realiza o mesmo trabalho na casa da Costa Leste.

Nesse exemplo, o serviço postal provê uma comunicação lógica entre as duas casas — ele movimenta a correspondência de uma casa para outra, e não de uma pessoa para outra. Por outro lado, Ann e Bill proveem comunicação lógica entre os primos — eles coletam e entregam a correspondência de seus irmãos e irmãs. Note que, da perspectiva dos primos, Ann e Bill são o serviço postal, embora sejam apenas uma parte do sistema (a parte do sistema final) do processo de entrega fim a fim. Esse exemplo das moradias é uma analogia interessante para explicar como a camada de transporte se relaciona com a camada de rede:

mensagens de aplicação = cartas em envelopes

processos = primos

hospedeiros (também denominados sistemas finais) = casas

protocolo de camada de transporte = Ann e Bill

protocolo de camada de rede = serviço postal (incluindo os carteiros)

Continuando com essa analogia, observe que Ann e Bill fazem todo o seu trabalho dentro de suas respectivas casas; eles não estão envolvidos, por exemplo, com a classificação da correspondência em nenhuma central intermediária dos correios ou com o transporte da correspondência de uma central a outra. De maneira semelhante, protocolos de camada de transporte moram nos sistemas finais, onde movimentam mensagens de processos de aplicação para a borda da rede (isto é, para a camada de rede) e vice-versa, mas não interferem no modo como as mensagens são movimentadas dentro do núcleo da rede. Na verdade, como ilustrado na Figura 3.1, roteadores intermediários não reconhecem nenhuma informação que a camada de transporte possa ter anexado às mensagens da aplicação nem agem sobre ela.

Prosseguindo com nossa saga familiar, suponha agora que, quando Ann e Bill saem de férias, outro par de primos — digamos, Susan e Harvey — substitua-os e encarregue-se da coleta interna da correspondência e de sua entrega. Infelizmente para as duas famílias, eles não desempenham essa tarefa do mesmo modo que Ann e Bill.

Por serem crianças mais novas, Susan e Harvey recolhem e entregam a correspondência com menos frequência e, ocasionalmente, perdem cartas (que às vezes acabam mastigadas pelo cão da família). Assim, o par de primos Susan e Harvey não provê o mesmo conjunto de serviços (isto é, o mesmo modelo de serviço) oferecido por Ann e Bill. De uma maneira análoga, uma rede de computadores pode disponibilizar vários protocolos de transporte, em que cada um oferece um modelo de serviço diferente às aplicações.

Os possíveis serviços que Ann e Bill podem fornecer são claramente limitados pelos possíveis serviços que os correios fornecem. Por exemplo, se o serviço postal não estipula um prazo máximo para entregar a correspondência entre as duas casas (digamos, três dias), então não há nenhuma possibilidade de Ann e Bill definirem um atraso máximo para a entrega da correspondência entre qualquer par de primos. De maneira semelhante, os serviços que um protocolo de transporte pode fornecer são frequentemente limitados pelo modelo de serviço do protocolo subjacente da camada de rede. Se o protocolo de camada de rede não puder dar garantias contra atraso ou garantias de largura de banda para segmentos de camada de transporte enviados entre hospedeiros, então o protocolo de camada de transporte não poderá dar essas mesmas garantias para mensagens de aplicação enviadas entre processos.

No entanto, certos serviços *podem* ser oferecidos por um protocolo de transporte mesmo quando o protocolo de rede subjacente não oferece o serviço correspondente na camada de rede. Por exemplo, como veremos neste capítulo, um protocolo de transporte pode oferecer serviço confiável de transferência de dados a uma aplicação mesmo quando o protocolo subjacente da rede não é confiável, isto é, mesmo quando o protocolo de rede perde, embaralha ou duplica pacotes. Como outro exemplo (que exploraremos no Capítulo 8, quando discutirmos segurança de rede), um protocolo de transporte pode usar criptografia para garantir que as mensagens da aplicação não sejam lidas por intrusos mesmo quando a camada de rede não puder garantir o sigilo de segmentos de camada de transporte.

3.1.2 Visão geral da camada de transporte na Internet

Lembre-se de que a Internet — e, de maneira mais geral, a rede TCP/IP — disponibiliza dois protocolos de transporte distintos para a camada de aplicação. Um deles é o **UDP** (User Datagram Protocol — Protocolo de Datagrama de Usuário), que provê à aplicação solicitante um serviço não confiável, não orientado para conexão. O segundo desses protocolos é o **TCP** (Transmission Control Protocol — Protocolo de Controle de Transmissão), que provê à aplicação solicitante um serviço confiável, orientado para conexão. Ao projetar uma aplicação de rede, o criador da aplicação deve especificar um desses dois protocolos de transporte. Como vimos nas seções 2.7 e 2.8, o desenvolvedor da aplicação escolhe entre o UDP e o TCP ao criar sockets.

Para simplificar a terminologia, quando no contexto da Internet, faremos alusão ao pacote de camada de transporte como um *segmento*. Devemos mencionar, contudo, que a literatura da Internet (por exemplo, os RFCs) também se refere ao pacote de camada de transporte com/para TCP como um segmento, mas muitas vezes se refere ao pacote com/para UDP como um *datagrama*. Porém, essa mesma literatura também usa o termo *datagrama* para o pacote de camada de rede! Como este é um livro de introdução a redes de computadores, acreditamos que será menos confuso se nos referirmos a ambos os pacotes TCP e UDP como segmentos; reservaremos o termo *datagrama* para o pacote de camada de rede.

Antes de continuarmos com nossa breve apresentação do UDP e do TCP, é útil dizer algumas palavras sobre a camada de rede da Internet. (A camada de rede é examinada detalhadamente no Capítulo 4.) O protocolo de camada de rede da Internet tem um nome — IP, que quer dizer *Internet Protocol*. O IP provê comunicação lógica entre hospedeiros. O modelo de serviço do IP é um **serviço de entrega de melhor esforço**, o que significa que o IP faz o ‘melhor esforço’ para levar segmentos entre hospedeiros comunicantes, *mas não dá nenhuma garantia*. Em especial, o IP não garante a entrega de segmentos, a entrega ordenada de segmentos e tampouco a integridade dos dados nos segmentos. Por essas razões, ele é denominado um **serviço não confiável**. Mencionamos também neste livro que cada hospedeiro tem, no mínimo, um endereço de camada de rede, denominado endereço IP. Examinaremos endereçamento IP detalhadamente no Capítulo 4. Para este capítulo, precisamos apenas ter em mente que *cada hospedeiro tem um endereço IP*.

Agora que abordamos ligeiramente o modelo de serviço IP, vamos resumir os modelos de serviço providos por UDP e TCP. A responsabilidade fundamental do UDP e do TCP é ampliar o serviço de entrega IP entre dois sistemas finais para um serviço de entrega entre dois processos que rodam nos sistemas finais. A ampliação da entrega hospedeiro a hospedeiro para entrega processo a processo é denominada **multiplexação/demultiplexação de camada de transporte**. Discutiremos multiplexação e demultiplexação de camada de transporte na próxima seção. O UDP e o TCP também fornecem verificação de integridade ao incluir campos de detecção de erros nos cabeçalhos de seus segmentos. Esses dois serviços mínimos de camada de transporte — entrega de dados processo a processo e verificação de erros — são os únicos que o UDP fornece! Em especial, como o IP, o UDP é um serviço não confiável — ele não garante que os dados enviados por um processo cheguem (quando chegam!) intactos ao processo destinatário. O UDP será discutido detalhadamente na Seção 3.3.

O TCP, por outro lado, oferece vários serviços adicionais às aplicações. Primeiramente, e mais importante, ele provê **transferência confiável de dados**. Usando controle de fluxo, números de sequência, reconhecimentos e temporizadores (técnicas que exploraremos detalhadamente neste capítulo), o protocolo assegura que os dados sejam entregues do processo remetente ao processo destinatário corretamente e em ordem. Assim, o TCP converte o serviço não confiável do IP entre sistemas finais em um serviço confiável de transporte de dados entre processos. Ele também provê **controle de congestionamento**. O controle de congestionamento não é tanto um serviço fornecido à aplicação solicitante; é mais um serviço dirigido à Internet como um todo — para o bem geral. Em termos genéricos, o controle de congestionamento do TCP evita que qualquer outra conexão TCP aborreto os enlaces e comutadores entre hospedeiros comunicantes com uma quantidade excessiva de tráfego. Em princípio, o TCP permite que conexões TCP trafegando por um enlace de rede congestionado compartilhem em pé de igualdade a largura de banda daquele enlace. Isso é feito pela regulagem da taxa com a qual o lado remetente do TCP pode enviar tráfego para dentro da rede. O tráfego UDP, por outro lado, não é regulado. Uma aplicação que usa transporte UDP pode enviar tráfego à taxa que quiser, pelo tempo que quiser.

Um protocolo que fornece transferência confiável de dados e controle de congestionamento é, necessariamente, complexo. Precisaremos de várias seções para detalhar os princípios da transferência confiável de dados e do controle de congestionamento, bem como de seções adicionais para explicar o protocolo TCP. Esses tópicos são analisados nas seções 3.4 a 3.8. A abordagem escolhida neste capítulo é alternar entre princípios básicos e o protocolo TCP. Por exemplo, discutiremos primeiramente a transferência confiável de dados em âmbito geral e, em seguida, como o TCP fornece especificamente a transferência confiável de dados. De maneira semelhante, discutiremos inicialmente o controle de congestionamento em âmbito geral e, em seguida, como o TCP realiza o controle de congestionamento. Mas, antes de chegarmos a essa parte boa, vamos examinar, primeiramente, multiplexação/demultiplexação na camada de transporte.

3.2 Multiplexação e demultiplexação

Nesta seção, discutiremos multiplexação e demultiplexação na camada de transporte, isto é, a ampliação do serviço de entrega hospedeiro a hospedeiro provido pela camada de rede para um serviço de entrega processo a processo para aplicações que rodam nesses hospedeiros. Para manter a discussão em nível concreto, vamos examinar esse serviço básico da camada de transporte no contexto da Internet. Enfatizamos, contudo, que o serviço de multiplexação/demultiplexação é necessário para todas as redes de computadores.

No hospedeiro de destino, a camada de transporte recebe segmentos da camada de rede logo abaixo dela e tem a responsabilidade de entregar os dados desses segmentos ao processo de aplicação apropriado que roda no hospedeiro. Vamos examinar um exemplo. Suponha que você esteja sentado à frente de seu computador, descarregando páginas Web enquanto roda uma sessão FTP e duas sessões Telnet. Por conseguinte, você tem quatro processos de aplicação de rede em execução — dois processos Telnet, um processo FTP e um processo HTTP. Quando a camada de transporte em seu computador receber dados da camada de rede abaixo dela, precisará direcionar os dados recebidos a um desses quatro processos. Vamos ver agora como isso é feito.

Em primeiro lugar, lembre-se de que dissemos, nas seções 2.7 e 2.8, que um processo (como parte de uma aplicação de rede) pode ter um ou mais **sockets**, portas pelas quais dados passam da rede para o processo e do

processo para a rede. Assim, como mostra a Figura 3.2, a camada de transporte do hospedeiro destinatário na verdade não entrega dados diretamente a um processo, mas a um socket intermediário. Como, a qualquer dado instante, pode haver mais de um socket no hospedeiro destinatário, cada um tem um identificador exclusivo. O formato do identificador depende de o socket ser UDP ou TCP, como discutiremos em breve.

Agora, vamos considerar como um hospedeiro destinatário direciona à porta correta um segmento de camada de transporte que chega. Cada segmento de camada de transporte tem um conjunto de campos para essa finalidade. Na extremidade receptora, a camada de transporte examina esses campos para identificar a porta receptora e direcionar o segmento a esse socket. A tarefa de entregar os dados contidos em um segmento da camada de transporte à porta correta é denominada **demultiplexação**. O trabalho de reunir, no hospedeiro de origem, porções de dados provenientes de diferentes portas, encapsular cada porção de dados com informações de cabeçalho (que mais tarde serão usadas na demultiplexação) para criar segmentos, e passar esses segmentos para a camada de rede é denominada **multiplexação**. Note que a camada de transporte do hospedeiro que está no meio da Figura 3.2 tem de demultiplexar segmentos que chegam da camada de rede abaixo para os processos P1 ou P2 acima; isso é feito direcionando à porta correspondente do processo os dados contidos no segmento que está chegando. A camada de transporte desse hospedeiro também tem de juntar dados de saída dessas portas, formar segmentos de camada de transporte e passá-los à camada de rede. Embora tenhamos apresentado multiplexação e demultiplexação no contexto dos protocolos de transporte da Internet, é importante entender que essas operações estarão presentes sempre que um único protocolo em uma camada (na de transporte ou em qualquer outra) for usado por vários protocolos na camada mais alta seguinte.

Para ilustrar o serviço de demultiplexação, lembre-se da metáfora das moradias apresentada na seção anterior. Cada uma das crianças é identificada por seu nome próprio. Quando Bill recebe uma grande quantidade de correspondência do carteiro, realiza uma operação de demultiplexação ao examinar para quem as cartas estão endereçadas e, em seguida, entregar a correspondência a seus irmãos e irmãs. Ann realiza uma operação de multiplexação quando coleta as cartas de seus irmãos e irmãs e entrega a correspondência na agência do correio.

Agora que entendemos os papéis da multiplexação e da demultiplexação na camada de transporte, vamos examinar como isso é feito em um hospedeiro. Sabemos, pela nossa discussão anterior, que multiplexação na camada de rede requer (1) que as portas tenham identificadores exclusivos e (2) que cada segmento tenha campos especiais que indiquem a porta para a qual o segmento deve ser entregue. Esses campos especiais, ilustrados na Figura 3.3, são o **campo de número de porta da fonte** e o **campo de número de porta do destino**. (Os segmentos UDP e TCP têm outros campos também, que serão examinados nas seções subsequentes deste capítulo.) Cada número de porta é um número de 16 bits na faixa de 0 a 65535. Os números de porta entre 0 e 1023 são denominados **números de porta bem conhecidos**; eles são restritos, o que significa que estão reservados para utilização por protocolos de aplicação bem conhecidos, como HTTP (que usa a porta número 80) e FTP.

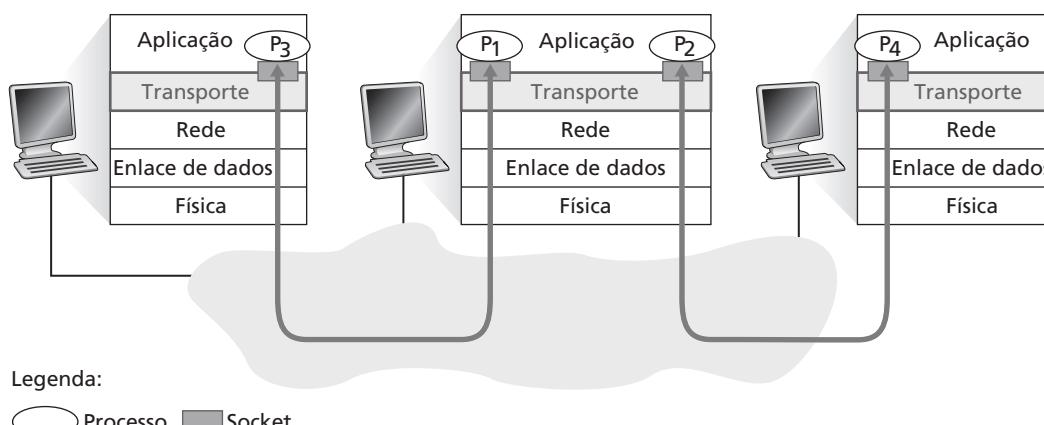


Figura 3.2 Multiplexação e demultiplexação na camada de transporte

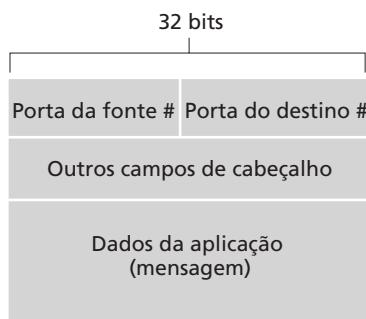


Figura 3.3 Campos de número de porta da fonte e do destino em um segmento de camada de transporte

(que usa a porta número 21). A lista dos números de porta bem conhecidos é apresentada no RFC 1700 e atualizada em <http://www.iana.org> [RFC 3232]. Quando desenvolvemos uma nova aplicação (como as desenvolvidas nas seções 2.7 a 2.8), devemos atribuir a ela um número de porta.

Agora já deve estar claro como a camada de transporte *poderia* implementar o serviço de demultiplexação: cada porta do hospedeiro pode receber um número designado; quando um segmento chega ao hospedeiro, a camada de transporte examina seu número de porta de destino e o direciona à porta correspondente. Então, os dados do segmento passam através da porta e entram no processo ligado a ela. Como veremos, é assim que o UDP faz demultiplexação. Todavia, veremos também que multiplexação/demultiplexação em TCP é ainda mais sutil.

Multiplexação e demultiplexação não orientadas para conexão

Lembre-se, na Seção 2.8, de que um programa em Java que roda em um hospedeiro pode criar uma porta UDP com a linha

```
 DatagramSocket mySocket = new DatagramSocket();
```

Quando uma porta UDP é criada dessa maneira, a camada de transporte automaticamente designa um número de porta ao socket. Em especial, a camada de transporte designa um número de porta na faixa de 1024 a 65535 que não esteja sendo usado naquele instante por qualquer outra porta UDP no hospedeiro. Alternativamente, um programa Java poderia criar uma porta com a linha

```
 DatagramSocket mySocket = new DatagramSocket(19157);
```

Nesse caso, a aplicação designa um número de porta específico — a saber, 19157 — à porta UDP. Se o desenvolvedor responsável por escrever o código da aplicação estivesse implementando o lado servidor de um ‘protocolo bem conhecido’, ele teria de designar o número de porta bem conhecido correspondente. O lado cliente da aplicação normalmente permite que a camada de transporte designe o número de porta automaticamente — e transparentemente — ao passo que o lado servidor da aplicação designa um número de porta específico.

Agora que os sockets UDP já têm seus números de porta designados, podemos descrever multiplexação/demultiplexação UDP com precisão. Suponha que um processo no hospedeiro A, cujo número de porta UDP é 19157, queira enviar uma porção de dados de aplicação a um processo cujo número de porta UDP seja 46428 no hospedeiro B. A camada de transporte no hospedeiro A cria um segmento de camada de transporte que inclui os dados de aplicação, o número da porta da fonte (19157), o número da porta de destino (46428) e mais outros dois valores (que serão discutidos mais adiante, mas que não são importantes para a discussão em curso). Então, a camada de transporte passa o segmento resultante para a camada de rede. Essa camada encapsula o segmento em um datagrama IP e faz uma tentativa de melhor esforço para entregar o segmento ao hospedeiro destinatário. Se o segmento chegar à máquina de destino B, a camada de destino no hospedeiro destinatário examinará o número da porta de destino no segmento (46428) e o entregará a seu socket identificado pelo número 46428. Note que a máquina B poderia estar rodando vários processos, cada um com sua própria porta UDP e número de porta

associado. À medida que segmentos UDP chegassem da rede, a máquina B direcionaria (demultiplexaria) cada segmento à porta apropriada examinando o número de porta de destino do segmento.

É importante notar que um socket UDP é totalmente identificado por uma tupla com dois elementos, consistindo em um endereço IP de destino e um número de porta de destino. Consequentemente, se dois segmentos UDP tiverem endereços IP de fonte e/ou números de porta de fonte diferentes, porém o mesmo endereço IP de destino e o mesmo número de porta de destino, eles serão direcionados ao mesmo processo de destino por meio do mesmo socket de destino.

É possível que agora você esteja imaginando qual é a finalidade do número de porta da fonte. Como mostra a Figura 3.4, no segmento A-B, o número de porta da fonte serve como parte de um “endereço de retorno” — quando B quer enviar um segmento de volta para A, a porta de destino no segmento B-A tomará seu valor do valor da porta de fonte do segmento A-B. (O endereço de retorno completo é o endereço IP e o número de porta de fonte de A). Como exemplo, lembre-se do programa servidor UDP que estudamos na Seção 2.8. Em `UDPServer.java`, o servidor usa um método para extrair o número de porta da fonte do segmento que recebe do cliente; então envia um novo segmento ao cliente, com o número de porta que extraiu servindo como o número de porta de destino desse novo segmento.

Multiplexação e demultiplexação orientadas para conexão

Para entender demultiplexação TCP, temos de examinar de perto sockets TCP e estabelecimento de conexão TCP. Há uma diferença sutil entre um socket UDP e um socket TCP: o socket TCP é identificado por uma tupla de quatro elementos: (endereço IP da fonte, número de porta da fonte, endereço IP de destino, número de porta do destino). Assim, quando um segmento TCP que vem da rede chega a um hospedeiro, este usa todos os quatro valores para direcionar (demultiplexar) o segmento para o socket apropriado. Em especial, e ao contrário do UDP, dois segmentos TCP chegando com endereços IP de fonte ou números de porta de fonte diferentes serão direcionados para dois sockets diferentes (com exceção de um TCP que esteja carregando a requisição de estabelecimento de conexão original). Para perceber melhor, vamos considerar novamente o exemplo de programação cliente-servidor TCP apresentado na Seção 2.7:

A aplicação servidor TCP tem um socket de entrada que espera requisições de estabelecimento de conexão vindas de clientes TCP (ver Figura 2.29) na porta número 6789.

O cliente TCP gera um segmento de estabelecimento de conexão com a linha

```
Socket clientSocket = new Socket("serverHostName", 6789);
```

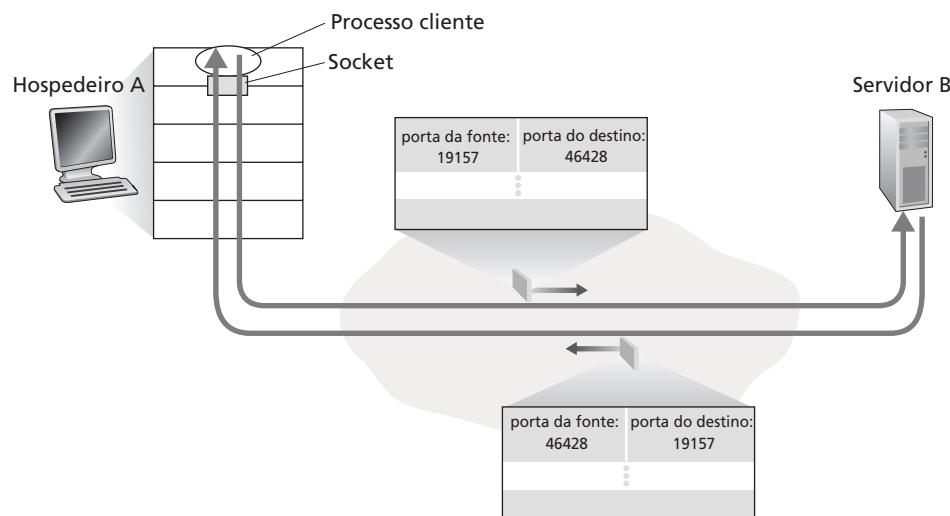


Figura 3.4 Campos de número de porta da fonte e do destino em um segmento de camada de transporte

Uma requisição de estabelecimento de conexão nada mais é do que um segmento TCP com número de porta de destino 6789 e um bit especial de estabelecimento de conexão marcado no cabeçalho TCP (que será discutido na Seção 3.5). O segmento inclui também um número de porta de fonte que foi escolhido pelo cliente. A linha acima cria ainda um socket TCP para o processo cliente, através do qual dados podem entrar e sair do processo cliente.

Quando o sistema operacional do computador que está rodando o processo servidor recebe o segmento de requisição de conexão que está chegando e cuja porta de destino é 6789, ele localiza o processo servidor que está esperando para aceitar uma conexão na porta número 6789. Então, o processo servidor cria um novo socket:

```
Socket connectionSocket = welcomeSocket.accept();
```

A camada de transporte no servidor também nota os quatro valores seguintes no segmento de requisição de conexão: (1) o número de porta da fonte no segmento, (2) o endereço IP do hospedeiro da fonte, (3) o número de porta do destino no segmento e (4) seu próprio endereço IP. O socket de conexão recém-criado é identificado por esses quatro valores; todos os segmentos subsequentes que chegarem, cuja porta da fonte, endereço IP da fonte, porta de destino e endereço IP de destino combinarem com esses quatro valores, serão demultiplexados para essa porta. Com a conexão TCP agora ativa, o cliente e o servidor podem enviar dados um para o outro.

O hospedeiro servidor pode suportar vários sockets TCP simultâneos, sendo cada um ligado a um processo e identificado por sua própria tupla de quatro elementos. Quando um segmento TCP chega ao hospedeiro, todos os quatro campos (endereço IP da fonte, porta da fonte, endereço IP de destino, porta de destino) são usados para direcionar (demultiplexar) o segmento para o socket apropriado.

A situação é ilustrada na Figura 3.5, na qual o hospedeiro C inicia duas sessões HTTP para o servidor B, e o hospedeiro A inicia uma sessão HTTP para o servidor B.

Os hospedeiros A e C e o servidor B possuem, cada um, seu próprio endereço IP exclusivo: A, C e B, respectivamente. O hospedeiro C atribui dois números diferentes (26145 e 7532) de porta da fonte às suas duas conexões HTTP. Como o hospedeiro A está escolhendo números de porta independentemente de C, ele poderia

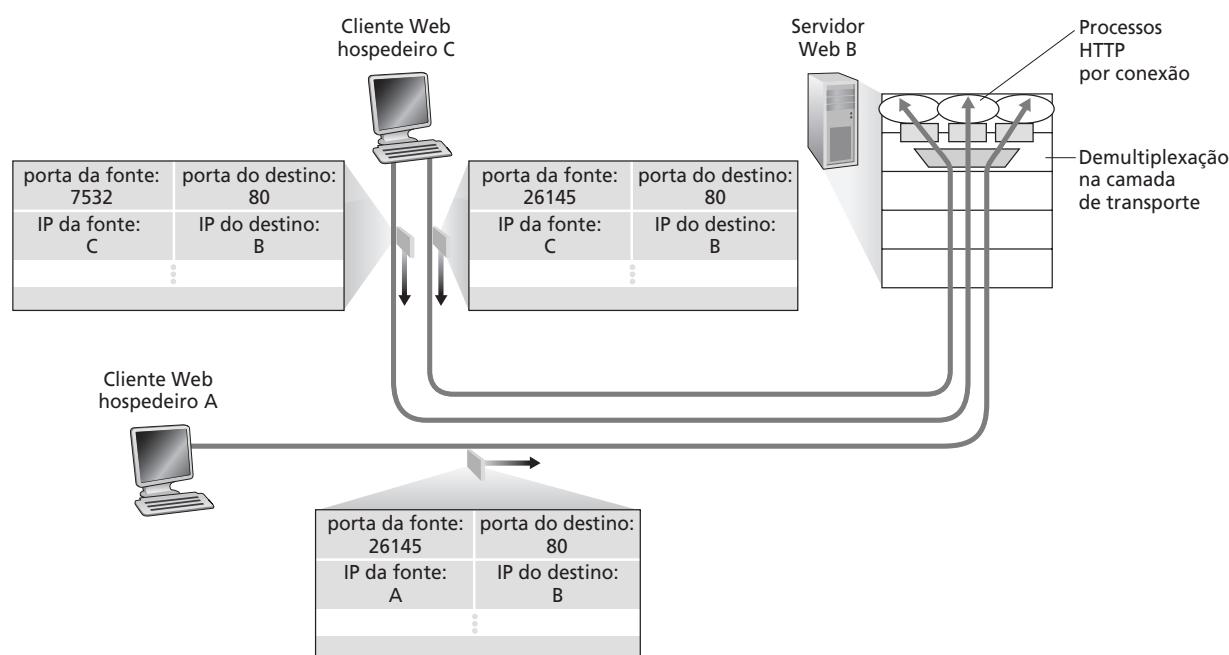


Figura 3.5 Dois clientes que usam o mesmo número de porta de destino (80) para se comunicar com a mesma aplicação de servidor Web

também atribuir um número de porta da fonte 26145 à sua conexão HTTP. Apesar disso, o servidor B ainda será capaz de demultiplexar corretamente as duas conexões que têm o mesmo número de porta de fonte, já que elas têm endereços IP de fonte diferentes.

Servidores Web e TCP

Antes de encerrar essa discussão, é instrutivo falar um pouco mais sobre servidores Web e como eles usam números de porta. Considere um hospedeiro rodando um servidor Web, tal como um Apache, na porta 80. Quando clientes (por exemplo, browsers) enviam segmentos ao servidor, *todos* os segmentos terão a porta de destino 80. Em especial, os segmentos que estabelecem a conexão inicial e os segmentos que carregam as mensagens de requisição HTTP, ambos terão a porta de destino 80. Como acabamos de descrever, o servidor distingue os segmentos dos diferentes clientes pelos endereços IP e pelos números de porta da fonte.

Como mostra a Figura 3.5, cada um desses processos tem seu próprio socket de conexão através do qual chegam requisições HTTP e são enviadas respostas HTTP. Mencionamos, contudo, que nem sempre existe uma correspondência unívoca entre sockets de conexão e processos. Na verdade, os servidores Web de alto desempenho atuais muitas vezes utilizam somente um processo, mas criam uma nova thread com um novo socket de conexão para cada nova conexão cliente. (Uma thread pode ser considerada um subprocesso leve.) Se você fez a primeira tarefa de programação do Capítulo 2, construiu um servidor Web que faz exatamente isso. Para um servidor desses, a qualquer dado instante podem haver muitos sockets de conexão (com identificadores diferentes) ligados ao mesmo processo.

Se o cliente e o servidor estiverem usando HTTP persistente, então, durante toda a conexão persistente, eles trocarão mensagens HTTP através do mesmo socket do servidor. Todavia, se o cliente e o servidor usarem HTTP



Segurança em foco

Varredura de Porta

Vimos que um processo servidor espera pacientemente em uma porta aberta para contato por um cliente remoto. Algumas portas são reservadas para aplicações familiares (por exemplo, Web, FTP, DNS e os serviços SMTP); outras portas são utilizadas por convenção por aplicações populares (por exemplo, o Microsoft SQL Server 2000 ouve as solicitações na porta 1434 do UDP). Desta forma, se determinarmos que uma porta está aberta no hospedeiro, talvez possamos mapear essa porta para uma aplicação específica sendo executada no hospedeiro. Isso é muito útil para administradores de sistema, que frequentemente têm interesse em saber quais aplicações de rede estão sendo executadas nos hospedeiros em suas redes. Porém, os atacantes, a fim de “examinarem o local”, também querem saber quais portas estão abertas nos hospedeiros direcionados. Se o hospedeiro estiver sendo executado em uma aplicação com uma falha de segurança conhecida (por exemplo, um SQL Server ouvindo em uma porta 1434 estava sujeito a esgotamento de buffer, permitindo que um usuário remoto execute um código arbitrário no hospedeiro vulnerável, uma falha explorada pelo worm Slammer [CERT, 2003-04]), então esse hospedeiro está pronto para o ataque.

Determinar quais aplicações estão ouvindo em quais portas é relativamente uma tarefa fácil. De fato há um número de programas de domínio público, denominados varredores de porta, que fazem exatamente isso. Talvez o mais utilizado seja o nmap, disponível gratuitamente em <http://insecure.org/nmap> e está incluso na maioria das distribuições Linux. Para o TCP, o nmap varre portas sequencialmente, procurando por portas que aceitam conexões TCP. Para o UDP, o nmap novamente varre portas sequencialmente, procurando por portas UDP que respondem aos segmentos UDP transmitidos. Em ambos os casos, o nmap retorna uma lista de portas abertas, fechadas ou inalcançáveis. Um hospedeiro executando o nmap pode tentar varrer qualquer hospedeiro direcionado em qualquer lugar da Internet. Voltaremos a falar sobre o nmap na Seção 3.5.6, ao discutirmos gerenciamento da conexão TCP.

não persistente, então uma nova conexão TCP é criada e encerrada para cada requisição/resposta e, portanto, um novo socket é criado e mais tarde encerrado para cada requisição/resposta. Essa criação e encerramento frequentes de sockets podem causar sério impacto sobre o desempenho de um servidor Web movimentado (embora o sistema operacional possa usar vários estratagemas para atenuar o problema). Aconselhamos o leitor interessado em questões de sistema operacional referentes a HTTP persistente e não persistente a consultar [Nielsen, 1997; Nahum, 2002].

Agora que já discutimos multiplexação e demultiplexação na camada de transporte, passemos à discussão de um dos protocolos da Internet, o UDP. Na próxima seção, veremos que a contribuição desse protocolo não é muito mais do que um serviço de multiplexação/demultiplexação.

3.3 Transporte não orientado para conexão: UDP

Nesta seção, examinaremos o UDP mais de perto, como ele funciona e o que ele faz. Aconselhamos o leitor a rever o material apresentado na Seção 2.1, que inclui uma visão geral do modelo de serviço UDP, e o da Seção 2.8, que discute a programação de portas por UDP.

Para motivar nossa discussão sobre UDP, suponha que você esteja interessado em projetar um protocolo de transporte simples, bem básico. Como você faria isso? De início, você deve considerar a utilização de um protocolo de transporte vazio. Em especial, do lado do remetente, considere pegar as mensagens do processo de aplicação e passá-las diretamente para a camada de rede; do lado do destinatário, considere pegar as mensagens que chegam da camada de rede e passá-las diretamente ao processo da aplicação. Mas, como aprendemos na seção anterior, o que teremos de fazer é praticamente nada. No mínimo, a camada de transporte tem de fornecer um serviço de multiplexação/demultiplexação para passar os dados da camada de rede ao processo de aplicação correto.

O UDP, definido no [RFC 768], faz apenas quase tão pouco quanto um protocolo de transporte pode fazer. À parte sua função de multiplexação/demultiplexação e de alguma verificação de erros, ele nada adiciona ao IP. Na verdade, se o criador da aplicação escolher o UDP, em vez do TCP, a aplicação estará ‘falando’ quase diretamente com o IP. O UDP pega as mensagens do processo de aplicação, anexa os campos de número de porta da fonte e do destino para o serviço de multiplexação/demultiplexação, adiciona dois outros pequenos campos e passa o segmento resultante à camada de rede, que encapsula o segmento dentro de um datagrama IP e, em seguida, faz uma tentativa de melhor esforço para entregar o segmento ao hospedeiro receptor. Se o segmento chegar ao hospedeiro receptor, o UDP usará o número de porta de destino para entregar os dados do segmento ao processo de aplicação correto. Note que, com o UDP, não há apresentação entre as entidades remetente e destinatária da camada de transporte antes de enviar um segmento. Por essa razão, dizemos que o UDP é não orientado para conexão.

O DNS é um exemplo de protocolo de camada de aplicação que usa o UDP. Quando a aplicação DNS em um hospedeiro quer fazer uma consulta, constrói uma mensagem de consulta DNS e passa a mensagem para o UDP. Sem realizar nenhuma apresentação com a entidade UDP que está funcionando no sistema final de destino, o UDP do lado do hospedeiro adiciona campos de cabeçalho à mensagem e passa o segmento resultante à camada de rede, que encapsula o segmento UDP em um datagrama e o envia a um servidor de nomes. A aplicação DNS no hospedeiro requisitante então espera por uma resposta à sua consulta. Se não receber uma resposta (possivelmente porque a rede subjacente perdeu a consulta ou a resposta), ela tentará enviar a consulta a outro servidor de nomes ou informará à aplicação consultante que não pode obter uma resposta.

É possível que agora você esteja imaginando por que um criador de aplicação escolheria construir uma aplicação sobre UDP, em vez de sobre TCP. O TCP não é sempre preferível ao UDP, já que fornece serviço confiável de transferência de dados e o UDP não? A resposta é ‘não’, pois muitas aplicações se adaptam melhor ao UDP pelas seguintes razões:

Melhor controle no nível da aplicação sobre quais dados são enviados e quando. Com UDP, tão logo um processo de aplicação passe dados ao UDP, o protocolo empacotará esses dados dentro de um segmento UDP e os passará imediatamente à camada de rede. O TCP, por outro lado, tem um mecanismo de controle de congestionamento que limita o remetente TCP da camada de transporte quando um ou mais



enlaces entre os hospedeiros da fonte e do destinatário ficam excessivamente congestionados. O TCP também continuará a reenviar um segmento até que o hospedeiro destinatário reconheça a recepção desse segmento, independentemente do tempo que a entrega confiável levar. Visto que aplicações de tempo real requerem uma taxa mínima de envio, não querem atrasar excessivamente a transmissão de segmentos e podem tolerar uma certa perda de dados, o modelo de serviço do TCP não é particularmente compatível com essas necessidades das aplicações. Como discutiremos mais adiante, essas aplicações podem usar UDP e implementar, como parte da aplicação, qualquer funcionalidade adicional necessária além do serviço de entrega de segmentos simples e básico do UDP.

Não há estabelecimento de conexão. Como discutiremos mais adiante, o TCP usa uma apresentação de três vias antes de começar a transferir dados. O UDP simplesmente envia mensagens sem nenhuma preliminar formal e, assim, não introduz nenhum atraso para estabelecer uma conexão. Provavelmente esta é a principal razão pela qual o DNS roda sobre UDP, e não sobre TCP — o DNS seria muito mais lento se rodasse em TCP. O HTTP usa o TCP, e não o UDP, porque a confiabilidade é fundamental para páginas Web com texto. Mas, como discutimos brevemente na Seção 2.2, o atraso de estabelecimento de uma conexão TCP é uma contribuição importante aos atrasos associados à recepção de documentos Web.

Não há estados de conexão. O TCP mantém o estado de conexão nos sistemas finais. Esse estado inclui buffers de envio e recebimento, parâmetros de controle de congestionamento e parâmetros numéricos de sequência e de reconhecimento. Veremos na Seção 3.5 que essa informação de estado é necessária para implementar o serviço de transferência confiável de dados do TCP e para prover controle de congestionamento. O UDP, por sua vez, não mantém o estado de conexão e não monitora nenhum desses parâmetros. Por essa razão, um servidor dedicado a uma aplicação específica pode suportar um número muito maior de clientes ativos quando a aplicação roda sobre UDP e não sobre TCP.

Pequena sobrecarga de cabeçalho de pacote. O segmento TCP tem 20 bytes de sobrecarga de cabeçalho além dos dados para cada segmento, enquanto o UDP tem somente 8 bytes de sobrecarga.

A Figura 3.6 relaciona aplicações populares da Internet e os protocolos de transporte que elas usam. Como era de esperar, o e-mail, o acesso a terminal remoto, a Web e a transferência de arquivos rodam sobre TCP — todas essas aplicações necessitam do serviço confiável de transferência de dados do TCP. Não obstante, muitas aplicações importantes executam sobre UDP, e não sobre TCP. O UDP é usado para atualização das tabelas de roteamento com o protocolo RIP (*routing information protocol* — protocolo de informação de roteamento) (veja Seção 4.6.1). Como as atualizações RIP são enviadas periodicamente (em geral, a cada 5 minutos), atualizações perdidas serão substituídas por atualizações mais recentes, tornando inútil a recuperação das atualizações perdidas. O UDP também é usado para levar dados de gerenciamento de rede (SNMP; veja o Capítulo 9). Nesse caso, o UDP é preferível ao TCP, já que aplicações de gerenciamento de rede frequentemente devem funcionar quando a rede está em estado sobrecarregado — exatamente quando é difícil conseguir transferência confiável de dados com congestionamento controlado. E também, como mencionamos anteriormente, o DNS roda sobre UDP, evitando, desse modo, atrasos de estabelecimento de conexões TCP.

Como mostra a Figura 3.6, hoje o UDP e o TCP também é comumente usado para aplicações de multimídia, como telefone por Internet, videoconferência em tempo real e recepção de áudio e vídeo armazenados. Examinaremos essas aplicações mais de perto no Capítulo 7. No momento, mencionamos apenas que todas essas aplicações podem tolerar uma pequena quantidade de perda de pacotes, de modo que a transferência confiável de dados não é absolutamente crítica para o sucesso da aplicação. Além disso, aplicações em tempo real, como telefone por Internet e videoconferência, reagem muito mal ao controle de congestionamento do TCP. Por essas razões, os desenvolvedores de aplicações de multimídia muitas vezes optam por rodar suas aplicações sobre UDP em vez de sobre TCP. Entretanto, o TCP está sendo utilizado cada vez mais para transporte de mídia. Por exemplo, [Sripanidkulchai, 2004] descobriu que aproximadamente 75% do fluxo em tempo real e gravado utilizaram TCP. Quando as taxas de perda de pacote são baixas, junto com algumas empresas que bloqueiam o tráfego UDP por razões de segurança (veja Capítulo 8), o TCP se torna um protocolo cada vez mais atrativo para o transporte de mídia.

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivo	FTP	TCP
Servidor remoto de arquivo	NFS	tipicamente UDP
Recepção de multimídia	tipicamente proprietária	UDP ou TCP
Telefonia por Internet	tipicamente proprietária	UDP ou TCP
Gerenciamento de rede	SNMP	tipicamente UDP
Protocolo de roteamento	RIP	tipicamente UDP
Tradução de nome	DNS	tipicamente UDP

Figura 3.6 Aplicações populares da Internet e seus protocolos de transporte subjacentes

Embora atualmente seja comum rodar aplicações de multimídia sobre UDP, isso é controvertido. Como já mencionamos, o UDP não tem controle de congestionamento. Mas esse controle é necessário para evitar que a rede entre em um estado no qual pouquíssimo trabalho útil é realizado. Se todos começassem a enviar vídeo com alta taxa de bits sem usar nenhum controle de congestionamento, haveria tamanho transbordamento de pacotes nos roteadores que poucos pacotes UDP conseguiram atravessar com sucesso o caminho da fonte ao destino. Além do mais, as altas taxas de perda induzidas pelos remetentes UDP sem controle fariam com que os remetentes TCP (que, como veremos mais adiante, *reduzem suas taxas de envio em face de congestionamento*) reduzissem drasticamente suas taxas. Assim, a falta de controle de congestionamento no UDP pode resultar em altas taxas de perda entre um remetente e um destinatário UDP e no acúmulo de sessões TCP — um problema potencialmente sério [Floyd, 1999]. Muitos pesquisadores propuseram novos mecanismos para forçar todas as fontes, inclusive as fontes UDP, a realizar um controle de congestionamento adaptativo [Mahdavi, 1997; Floyd, 2000; Kohler, 2006: RFC 4340].

Antes de discutirmos a estrutura do segmento UDP, mencionaremos que é possível que uma aplicação tenha transferência confiável de dados usando UDP. Isso pode ser feito se a confiabilidade for embutida na própria aplicação (por exemplo, adicionando mecanismos de reconhecimento e de retransmissão, tais como os que estudaremos na próxima seção). Mas esta é uma tarefa não trivial que manteria o desenvolvedor ocupado com a depuração por um longo tempo. Não obstante, embutir confiabilidade diretamente na aplicação permite que ela tire proveito de ambas as alternativas. Em outras palavras, os processos de aplicação podem se comunicar de maneira confiável sem ter de se sujeitar às limitações da taxa de transmissão impostas pelo mecanismo de controle de congestionamento do TCP.

3.3.1 Estrutura do segmento UDP

A estrutura do segmento UDP, mostrada na Figura 3.7, é definida no RFC 768. Os dados da aplicação ocupam o campo de dados do segmento UDP. Por exemplo, para o DNS, o campo de dados contém uma mensagem de consulta ou uma mensagem de resposta. Para uma aplicação de recepção de áudio, parcelas de áudio preenchem o campo de dados. O cabeçalho UDP tem apenas quatro campos, cada um consistindo em 2 bytes. Como já discutido na seção anterior, os números de porta permitem que o hospedeiro destinatário passe os dados da aplicação ao processo correto que está funcionando na máquina destinatária (isto é, realize a função de demultiplexação). A soma de verificação é usada pelo hospedeiro receptor para verificar se foram introduzidos erros no segmento. Na verdade, a soma de verificação também é calculada para alguns dos campos no cabeçalho IP, além do segmento UDP. Mas ignoramos esse detalhe para podermos enxergar a floresta por entre as árvores. Discutiremos o cálculo

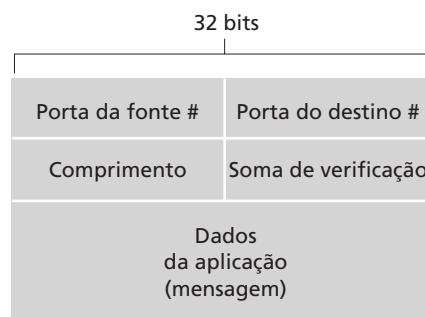


Figura 3.7 Estrutura do segmento UDP

da soma de verificação mais adiante. Os princípios básicos da detecção de erros estão descritos na Seção 5.2. O campo de comprimento especifica o comprimento do segmento UDP, incluindo o cabeçalho, em bytes.

3.3.2 Soma de verificação UDP

A soma de verificação UDP serve para detectar erros. Em outras palavras, é usada para determinar se bits dentro do segmento UDP foram alterados (por exemplo, por ruído nos enlaces ou enquanto armazenados em um roteador) durante sua movimentação da fonte até o destino. O UDP no lado remetente realiza o complemento de 1 da soma de todas as palavras de 16 bits do segmento levando em conta o “vai um” em toda a soma. Esse resultado é colocado no campo de soma de verificação no segmento UDP. Damos aqui um exemplo simples do cálculo da soma de verificação. Se quiser saber detalhes sobre a implementação eficiente do algoritmo de cálculo e sobre o desempenho com dados reais, consulte o RFC 1071 e [Stone, 1998 e 2000], respectivamente. Como exemplo, suponha que tenhamos as seguintes três palavras de 16 bits:

```
0110011001100000
0101010101010101
1000111100001100
```

A soma das duas primeiras dessas palavras de 16 bits é:

```
0110011001100000
0101010101010101
1011101110110101
```

Adicionando a terceira palavra à soma acima, temos:

```
1011101110110101
1000111100001100
0100101011000010
```

Note que essa última adição teve “vai um” no bit mais significativo que foi somado ao bit menos significativo. O complemento de 1 é obtido pela conversão de todos os 0 em 1 e de todos os 1 em 0. Desse modo, o complemento de 1 da soma 0100101011000010 é 1011010100111101, que passa a ser a soma de verificação. No destinatário, todas as quatro palavras de 16 bits são somadas, inclusive a soma de verificação. Se nenhum erro for introduzido no pacote, a soma no destinatário será, obviamente, 1111111111111111. Se um dos bits for um zero, saberemos então que houve introdução de erro no pacote.

Provavelmente você está imaginando por que o UDP fornece uma soma de verificação em primeiro lugar, visto que muitos protocolos de camada de enlace (entre os quais, o popular protocolo Ethernet) também for-

necem verificação de erros. A razão é que não há garantia de que todos os enlaces entre a origem e o destino forneçam verificação de erros — um dos enlaces pode usar um protocolo de camada de enlace que não forneça verificação de erros. Além disso, mesmo que os segmentos sejam corretamente transmitidos por um enlace, é possível haver introdução de erros de bits quando um segmento é armazenado na memória de um roteador. Dado que não são garantidas nem a confiabilidade enlace a enlace, nem a detecção de erro na memória, o UDP deve prover detecção de erro *fim a fim* na camada de transporte se quisermos que o serviço de transferência de dados *fim a fim* forneça detecção de erro. Esse é um exemplo do famoso **princípio fim a fim** do projeto de sistemas [Saltzer, 1984]. Esse princípio afirma que, uma vez que é dado como certo que funcionalidades (detecção de erro, neste caso) devem ser implementadas *fim a fim*, “funções colocadas nos níveis mais baixos podem ser redundantes ou de pouco valor em comparação com o custo de fornecê-las no nível mais alto”.

Como se pretende que o IP rode sobre qualquer protocolo de camada 2, é útil que a camada de transporte forneça verificação de erros como medida de segurança. Embora o UDP forneça verificação de erros, ele nada faz para recuperar um erro. Algumas implementações do UDP simplesmente descartam o segmento danificado; outras passam o segmento errado à aplicação acompanhado de um aviso.

Isso encerra nossa discussão sobre o UDP. Logo veremos que o TCP oferece transferência confiável de dados às suas aplicações, bem como outros serviços que o UDP não oferece. Naturalmente, o TCP também é mais complexo do que o UDP. Contudo, antes de discutirmos o TCP, primeiramente devemos examinar os princípios subjacentes da transferência confiável de dados.

3.4 Princípios da transferência confiável de dados

Nesta seção, consideramos o problema conceitual da transferência confiável de dados. Isso é apropriado, já que o problema de implementar transferência confiável de dados ocorre não somente na camada de transporte, mas também na camada de enlace e na camada de aplicação. Assim, o problema geral é de importância central para o trabalho em rede. Na verdade, se tivéssemos de fazer uma lista dos dez maiores problemas fundamentalmente importantes para o trabalho em rede, o da transferência confiável de dados seria o candidato número um da lista. Na seção seguinte, examinaremos o TCP e mostraremos, em especial, que ele utiliza muitos dos princípios que descreveremos aqui.

A Figura 3.8 ilustra a estrutura para nosso estudo de transferência confiável de dados. A abstração do serviço fornecido às entidades das camadas superiores é a de um canal confiável através do qual dados podem ser transferidos. Com um canal confiável, nenhum dos dados transferidos é corrompido (trocado de 0 para 1 ou vice-versa) nem perdido, e todos são entregues na ordem em que foram enviados. Este é exatamente o modelo de serviço oferecido pelo TCP às aplicações de Internet que recorrem a ele.

É responsabilidade de um **protocolo de transferência confiável de dados** implementar essa abstração de serviço. A tarefa é dificultada pelo fato de que a camada *abaixo* do protocolo de transferência confiável de dados pode ser não confiável. Por exemplo, o TCP é um protocolo confiável de transferência de dados que é implementado sobre uma camada de rede *fim a fim* não confiável (IP). De modo mais geral, a camada *abaixo* das duas extremidades que se comunicam confiavelmente pode consistir em um único enlace físico (como é o caso, por exemplo, de um protocolo de transferência de dados na camada de enlace) ou em uma rede global interligada (como é o caso de um protocolo de camada de transporte). Para nossa finalidade, contudo, podemos considerar essa camada mais baixa simplesmente como um canal ponto a ponto não confiável.

Nesta seção, desenvolveremos gradualmente os lados remetente e destinatário de um protocolo confiável de transferência de dados, considerando modelos progressivamente mais complexos do canal subjacente. A Figura 3.8(b) ilustra as interfaces de nosso protocolo de transferência de dados. O lado remetente do protocolo de transferência de dados será invocado de cima, por uma chamada a `rdt_send()`. Ele passará os dados a serem entregues à camada superior no lado destinatário. (Aqui, `rdt` representa o protocolo de transferência confiável de dados [*reliable data transfer*] e `_send` indica que o lado remetente do `rdt` está sendo chamado. O primeiro

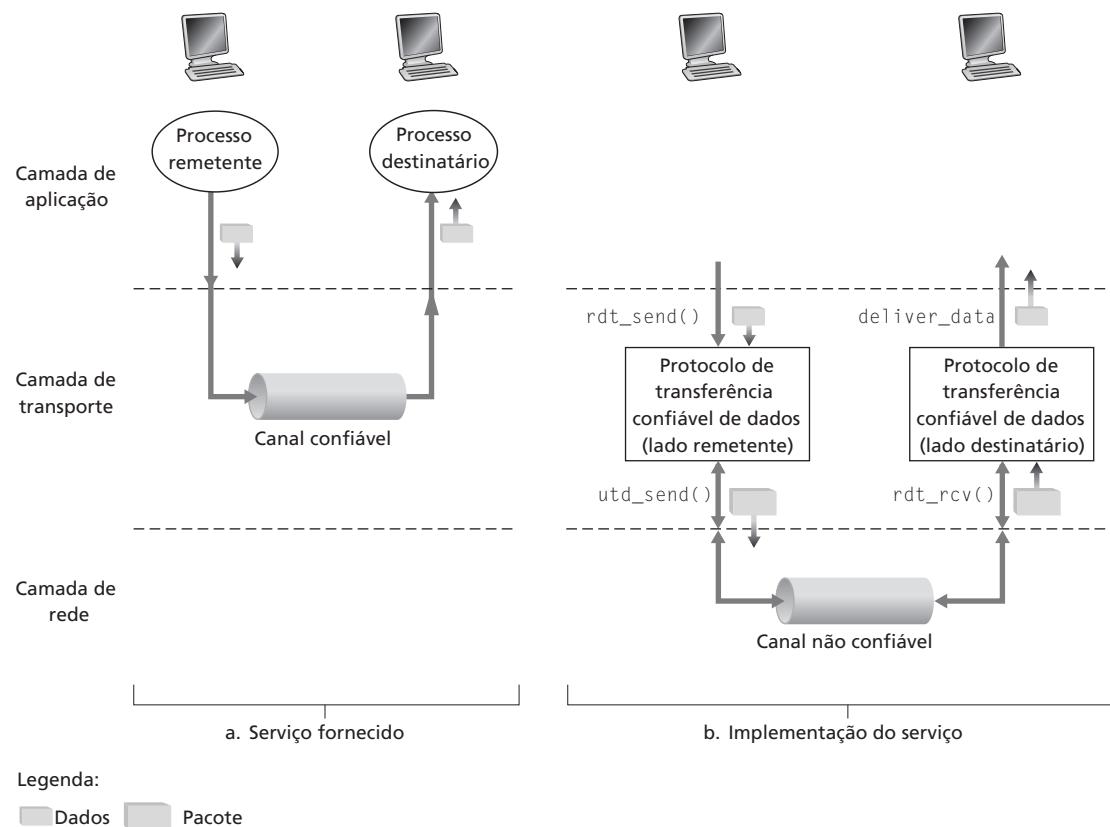


Figura 3.8 Transferência confiável de dados: modelo do serviço e implementação do serviço

passo no desenvolvimento de qualquer protocolo é dar-lhe um bom nome!) Do lado destinatário, `rdt_rcv()` será chamado quando um pacote chegar do lado destinatário do canal. Quando o protocolo `rdt` quiser entregar dados à camada superior, ele o fará chamando `deliver_data()`. No que se segue, usamos a terminologia ‘pacote’ em vez de ‘segmento’ de camada de transporte. Como a teoria desenvolvida nesta seção se aplica a redes de computadores em geral, e não somente à camada de transporte da Internet, o termo genérico ‘pacote’ talvez seja mais apropriado aqui.

Nesta seção, consideramos apenas o caso de **transferência unidirecional de dados**, isto é, transferência de dados do lado remetente ao lado destinatário. O caso de **transferência bidirecional** confiável de dados (isto é, *full-duplex*) não é conceitualmente mais difícil, mas é bem mais tedioso de explicar. Embora consideremos apenas a transferência unidirecional de dados, é importante notar que, apesar disso, os lados remetente e destinatário do nosso protocolo terão de transmitir pacotes em ambas as direções, como mostra a Figura 3.8. Logo veremos que, além de trocar pacotes contendo os dados a transferir, os lados remetente e destinatário do `rdt` também precisarão trocar pacotes de controle entre si. Ambos os lados de envio e destino do `rdt` enviam pacotes para o outro lado por meio de uma chamada a `utd_send()` (em que `utd` representa transferência não confiável de dados — *unreliable data transfer*).

3.4.1 Construindo um protocolo de transferência confiável de dados

Vamos percorrer agora uma série de protocolos que vão se tornando cada vez mais complexos, até chegar a um protocolo de transferência confiável de dados impecável.

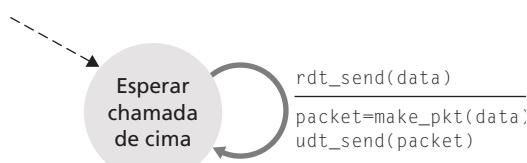
Transferência confiável de dados sobre um canal perfeitamente confiável: rdt1.0

Vamos considerar primeiramente o caso mais simples, em que o canal subjacente é completamente confiável. O protocolo em si, que denominaremos `rdt1.0`, é trivial. As definições de **máquina de estado finito** (*finite-state machine* — FSM) para o remetente e o destinatário `rdt1.0` são apresentadas na Figura 3.9. A FSM da Figura 3.9(a) define a operação do remetente, enquanto a FSM da Figura 3.9(b) define a operação do destinatário. É importante notar que há FSM *separadas* para o remetente e o destinatário. Ambas as FSM da Figura 3.9 têm apenas um estado. As setas na descrição da FSM indicam a transição do protocolo de um estado para outro. (Uma vez que cada FSM da Figura 3.9 tem apenas um estado, uma transição é, necessariamente, de um dado estado para ele mesmo; examinaremos diagramas de estados mais complicados em breve.) O evento que causou a transição é mostrado acima da linha horizontal que a rotula, e as ações realizadas quando ocorre o evento são mostradas abaixo da linha horizontal. Quando nenhuma ação é realizada em um evento, ou quando não ocorre nenhum evento e uma ação é realizada, usaremos o símbolo Λ , acima ou abaixo da linha horizontal, para indicar explicitamente a falta de uma ação ou de um evento, respectivamente. O estado inicial da FSM é indicado pela seta tracejada. Embora as FSMs da Figura 3.9 tenham somente um estado, as outras que veremos em breve têm vários estados, portanto, será importante identificar o estado inicial de cada FSM.

O lado remetente do `rdt` simplesmente aceita dados da camada superior pelo evento `rdt_send(data)`, cria um pacote que contém os dados (pela ação `make_pkt(data)`) e envia-o para dentro do canal. Na prática, o evento `rdt_send(data)` resultaria de uma chamada de procedimento (por exemplo, para `rdt_send()`) pela aplicação da camada superior.

Do lado destinatário, `rdt` recebe um pacote do canal subjacente pelo evento `rdt_rcv(packet)`, extraí os dados do pacote (pela ação `extract(packet, data)`) e os passa para a camada superior (pela ação `deliver_data(data)`). Na prática, o evento `rdt_rcv(packet)` resultaria de uma chamada de procedimento (por exemplo, para `rdt_rcv()`) do protocolo da camada inferior.

Nesse protocolo simples, não há diferença entre a unidade de dados e um pacote. E, também, todo o fluxo de pacotes corre do remetente para o destinatário; com um canal perfeitamente confiável, não há necessidade de o lado destinatário fornecer qualquer informação ao remetente, já que nada pode dar errado! Note que também admitimos que o destinatário está capacitado a receber dados seja qual for a velocidade em que o remetente os envie. Assim, não há necessidade de pedir para o remetente desacelerar!



a. `rdt1.0`: lado remetente



b. `rdt1.0`: lado destinatário

Figura 3.9 `rdt1.0` — Um protocolo para um canal completamente confiável

Transferência confiável de dados por um canal com erros de bits: rdt2.0

Um modelo mais realista de canal subjacente é um canal em que os bits de um pacote podem ser corrompidos. Esses erros de bits ocorrem normalmente nos componentes físicos de uma rede enquanto o pacote é transmitido, propagado ou armazenado. Continuaremos a admitir, por enquanto, que todos os pacotes transmitidos sejam recebidos (embora seus bits possam estar corrompidos) na ordem em que foram enviados.

Antes de desenvolver um protocolo para se comunicar de maneira confiável com esse canal, considere primeiramente como as pessoas enfrentariam uma situação como essa. Considere como você ditaria uma mensagem longa pelo telefone. Em um cenário típico, quem estivesse anotando a mensagem diria ‘o.k.’ após cada sentença que ouvisse, entendesse e anotasse. Se a pessoa ouvisse uma mensagem truncada, pediria que você a repetisse. Esse protocolo de ditado de mensagem usa **reconhecimentos positivos** (‘o.k.’) e **reconhecimentos negativos** (‘Repita, por favor’). Essas mensagens de controle permitem que o destinatário faça o remetente saber o que foi recebido corretamente e o que foi recebido com erro e, portanto, exige repetição. Em um arranjo de rede de computadores, protocolos de transferência confiável de dados baseados nesse tipo de retransmissão são conhecidos como **protocolos ARQ (Automatic Repeat reQuest — solicitação automática de repetição)**.

Essencialmente, são exigidas três capacitações adicionais dos protocolos ARQ para manipular a presença de erros de bits:

Detecção de erros. Primeiramente, é preciso um mecanismo que permita ao destinatário detectar quando ocorrem erros. Lembre-se de que dissemos na seção anterior que o UDP usa o campo de soma de verificação da Internet exatamente para essa finalidade. No Capítulo 5, examinaremos, com mais detalhes, técnicas de detecção e de correção de erros. Essas técnicas permitem que o destinatário detecte e possivelmente corrija erros de bits de pacotes. Por enquanto, basta saber que essas técnicas exigem que bits extras (além dos bits dos dados originais a serem transferidos) sejam enviados do remetente ao destinatário. Esses bits são colocados no campo de soma de verificação do pacote de dados do protocolo rdt2.0.

Realimentação do destinatário. Uma vez que remetente e destinatário normalmente estejam rodando em sistemas finais diferentes, possivelmente separados por milhares de quilômetros, o único modo de o remetente saber qual é a visão de mundo do destinatário (neste caso, se um pacote foi recebido corretamente ou não) é o destinatário fornecer realimentação explícita ao remetente. As respostas de reconhecimento positivo (ACK) ou negativo (NAK) no cenário do ditado da mensagem são exemplos dessa realimentação. Nossa protocolo rdt2.0 devolverá, dessa mesma maneira, pacotes ACK e NAK do destinatário ao remetente. Em princípio, esses pacotes precisam apenas ter o comprimento de um bit; por exemplo, um valor 0 poderia indicar um NAK e um valor 1 poderia indicar um ACK.

Retransmissão. Um pacote que é recebido com erro no destinatário será retransmitido pelo remetente.

A Figura 3.10 mostra a representação por FSM do rdt2.0, um protocolo de transferência de dados que emprega detecção de erros, reconhecimentos positivos e reconhecimentos negativos.

O lado remetente do rdt2.0 tem dois estados. No estado mais à esquerda, o protocolo do lado remetente está esperando que os dados sejam passados pela camada superior. Quando o evento `rdt_send(data)` ocorrer, o remetente criará um pacote (`sndpkt`) contendo os dados a serem enviados, juntamente com uma soma de verificação do pacote (por exemplo, como discutimos na Seção 3.3.2 para o caso de um segmento UDP) e, então, enviará o pacote pela operação `udt_send(sndpkt)`. No estado mais à direita, o protocolo remetente está esperando por um pacote ACK ou NAK da parte do destinatário. Se um pacote ACK for recebido (a notação `rdt_rcv(rcvpkt) && isACK(rcvpkt)` na Figura 3.10 corresponde a esse evento), o remetente saberá que o pacote transmitido mais recentemente foi recebido corretamente. Assim, o protocolo volta ao estado de espera por dados vindos da camada superior. Se for recebido um NAK, o protocolo retransmitirá o último pacote e esperará por um ACK ou NAK a ser devolvido pelo destinatário em resposta ao pacote de dados retransmitido. É importante notar que, quando o destinatário está no estado de espera por ACK ou NAK, não pode receber mais dados da camada superior; isto é, o evento `rdt_send()` não pode ocorrer; isso somente acontecerá após o remetente receber um ACK e sair desse estado. Assim, o remetente não enviará novos dados até ter certeza de que o

destinatário recebeu corretamente o pacote em questão. Devido a esse comportamento, protocolos como o rdt2.0 são conhecidos como protocolos pare e espere (*stop-and-wait*).

A FSM do lado destinatário para o rdt2.0 tem um único estado. Quando o pacote chega, o destinatário responde com um ACK ou um NAK, dependendo de o pacote recebido estar ou não corrompido. Na Figura 3.10, a notação `rdt_rcv(rcvpkt) && corrupt(rcvpkt)` corresponde ao evento em que um pacote é recebido e existe um erro.

Pode parecer que o protocolo rdt2.0 funciona, mas infelizmente ele tem um defeito fatal. Em especial, ainda não tratamos da possibilidade de o pacote ACK ou NAK estar corrompido! (Antes de continuar, é bom você começar a pensar em como esse problema pode ser resolvido.) Lamentavelmente, nossa pequena omissão não é tão inofensiva quanto possa parecer. No mínimo, precisaremos adicionar aos pacotes ACK/NAK bits de soma de verificação para detectar esses erros. A questão mais difícil é como o protocolo deve se recuperar de erros em pacotes ACK ou NAK. Nesse caso, a dificuldade é que, se um ACK ou um NAK estiverem corrompidos, o remetente não terá como saber se o destinatário recebeu ou não corretamente a última parcela de dados transmitidos.

Considere três possibilidades para manipular ACKs ou NAKs corrompidos:

Para a primeira possibilidade, imagine o que um ser humano faria no cenário do ditado da mensagem. Se quem estiver ditando não entender o ‘o.k.’ ou o ‘Repita, por favor’ do destinatário, provavelmente perguntará: “O que foi que você disse?” (introduzindo assim um novo tipo de pacote remetente–destinatário em nosso protocolo). O locutor então repetiria a resposta. Mas e se a frase “O que foi que

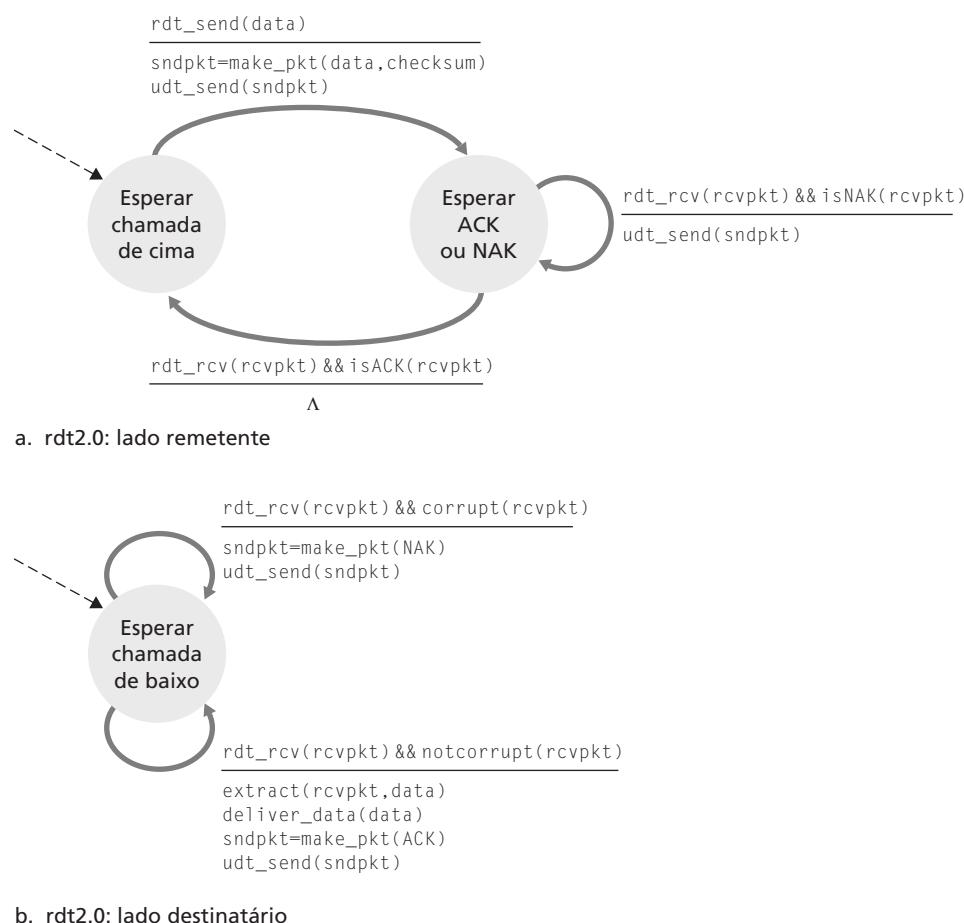


Figura 3.10 rdt2.0 — Um protocolo para um canal com erros de bits

você disse?" estivesse corrompida? O destinatário, sem ter nenhuma noção se a sentença corrompida era parte do ditado ou um pedido para repetir a última resposta, provavelmente responderia: "O que foi que você disse?" E então, é claro, essa resposta também poderia estar truncada. É óbvio que estamos entrando em um caminho difícil.

Uma segunda alternativa é adicionar um número suficiente de bits de soma de verificação para permitir que o remetente não somente detecte, mas também se recupere de erros de bits. Isso resolve o problema imediato para um canal que pode corromper pacotes, mas não perde-los.

Uma terceira abordagem é o remetente simplesmente reenviar o pacote de dados corrente quando receber um pacote ACK ou NAK truncado. Esse método, no entanto, introduz pacotes duplicados no canal remetente-destinatário. A dificuldade fundamental com pacotes duplicados é que o destinatário não sabe se o último ACK ou NAK que enviou foi recebido corretamente no remetente. Assim, ele não pode saber *a priori* se um pacote que chega contém novos dados ou se é uma retransmissão!

Uma solução simples para esse novo problema (e que é adotada em quase todos os protocolos de transferência de dados existentes, inclusive o TCP) é adicionar um novo campo ao pacote de dados e fazer com que o remetente numere seus pacotes de dados colocando um número de sequência nesse campo. O destinatário então teria apenas de verificar esse número de sequência para determinar se o pacote recebido é ou não uma retransmissão. Para esse caso simples de protocolo pare e espere, um número de sequência de um bit é suficiente, já que permitirá que o destinatário saiba se o remetente está reenviando o pacote previamente transmitido (o número de sequência do pacote recebido é o mesmo do pacote recebido mais recentemente) ou um novo pacote (o número de sequência muda, indo "para a frente" em progressão aritmética de módulo 2). Como estamos admitindo que este é um canal que não perde pacotes, os pacotes ACK e NAK em si não precisam indicar o número de sequência do pacote que estão reconhecendo. O remetente sabe que um pacote ACK ou NAK recebido (truncado ou não) foi gerado em resposta ao seu pacote de dados transmitidos mais recentemente.

As figuras 3.11 e 3.12 mostram a descrição da FSM para o rdt2.1, nossa versão corrigida do rdt2.0. Cada um dos rdt2.1 remetente e destinatário da FSM agora tem um número duas vezes maior de estados do que antes. Isso acontece porque o estado do protocolo deve agora refletir se o pacote que está sendo correntemente

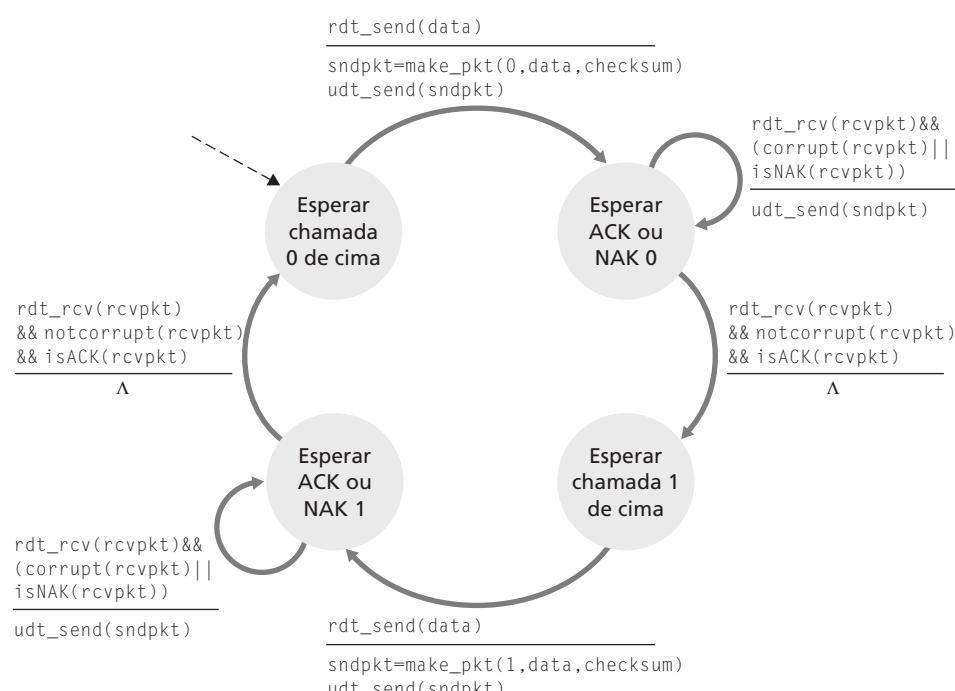


Figura 3.11 rdt2.1 remetente

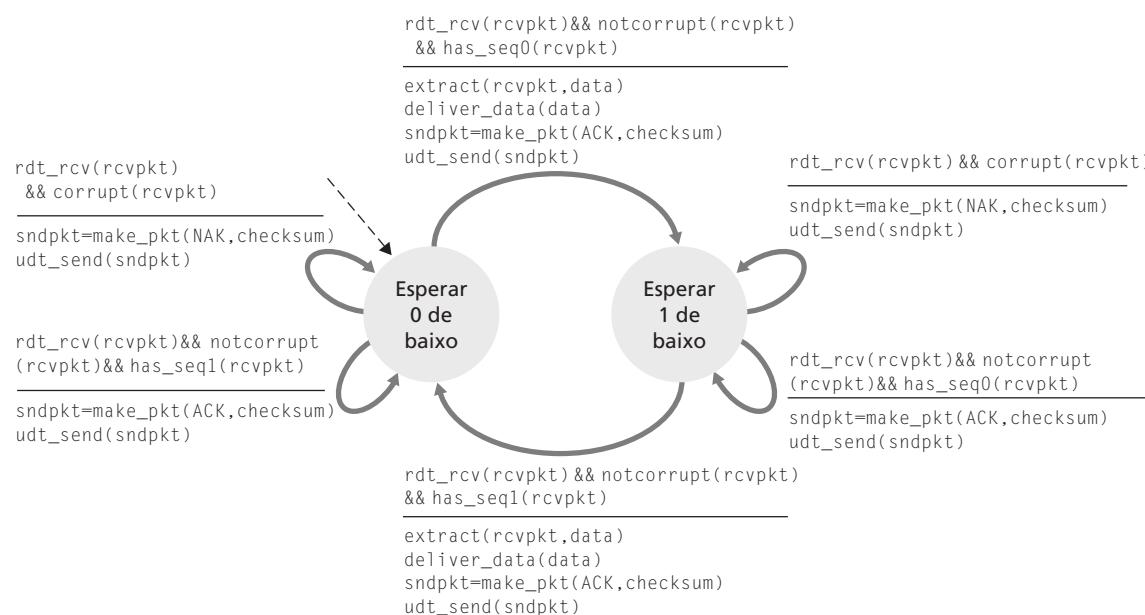


Figura 3.12 rdt2.1 destinatário

enviado (pelo remetente) ou aguardado (no destinatário) deveria ter um número de sequência 0 ou 1. Note que as ações nos estados em que um pacote numerado com 0 está sendo enviado ou aguardado são imagens especulares daquelas que devem funcionar quando estiver sendo enviado ou aguardado um pacote numerado com 1; as únicas diferenças têm a ver com a manipulação do número de sequência.

O protocolo rdt2.1 usa tanto o reconhecimento positivo como o negativo do remetente ao destinatário. Quando um pacote fora de ordem é recebido, o destinatário envia um reconhecimento positivo para o pacote que recebeu; quando um pacote corrompido é recebido, ele envia um reconhecimento negativo. Podemos conseguir o mesmo efeito de um pacote NAK se, em vez de enviarmos um NAK, enviarmos um ACK em seu lugar para o último pacote corretamente recebido. Um remetente que recebe dois ACKs para o mesmo pacote (isto é, **ACKs duplicados**) sabe que o destinatário não recebeu corretamente o pacote seguinte àquele para o qual estão sendo dados dois ACKs.

Nosso protocolo de transferência confiável de dados sem NAK para um canal com erros de bits é o rdt2.2, mostrado nas figuras 3.13 e 3.14. Uma modificação útil entre rdt2.1 e rdt2.2 é que o destinatário agora deve incluir o número de sequência do pacote que está sendo reconhecido por uma mensagem ACK (o que é feito incluindo o argumento ACK, 0 ou ACK, 1 em make_pkt() na FSM destinatária) e o remetente agora deve verificar o número de sequência do pacote que está sendo reconhecido por uma mensagem ACK recebida (o que é feito incluindo o argumento 0 ou 1 em isACK() na FSM remetente).

Transferência confiável de dados por um canal com perda e com erros de bits: rdt3.0

Suponha agora que, além de corromper bits, o canal subjacente possa perder pacotes, um acontecimento que não é incomum nas redes de computadores de hoje (incluindo a Internet). Duas preocupações adicionais devem agora ser tratadas pelo protocolo: como detectar perda de pacote e o que fazer quando isso ocorre. A utilização de soma de verificação, números de sequência, pacotes ACK e retransmissões — as técnicas já desenvolvidas em rdt2.2 — nos permitirão atender a última preocupação. Lidar com a primeira preocupação, por sua vez, exigirá a adição de um novo mecanismo de protocolo.

Há muitas abordagens possíveis para lidar com a perda de pacote (e diversas delas serão estudadas nos exercícios ao final do capítulo). Aqui, atribuiremos ao remetente o encargo de detectar e se recuperar das perdas de pacote. Suponha que o remetente transmita um pacote de dados e que esse pacote, ou o ACK do seu destinatário, seja perdido. Em qualquer um dos casos, nenhuma resposta chegará ao remetente vinda do destinatário. Se o remetente

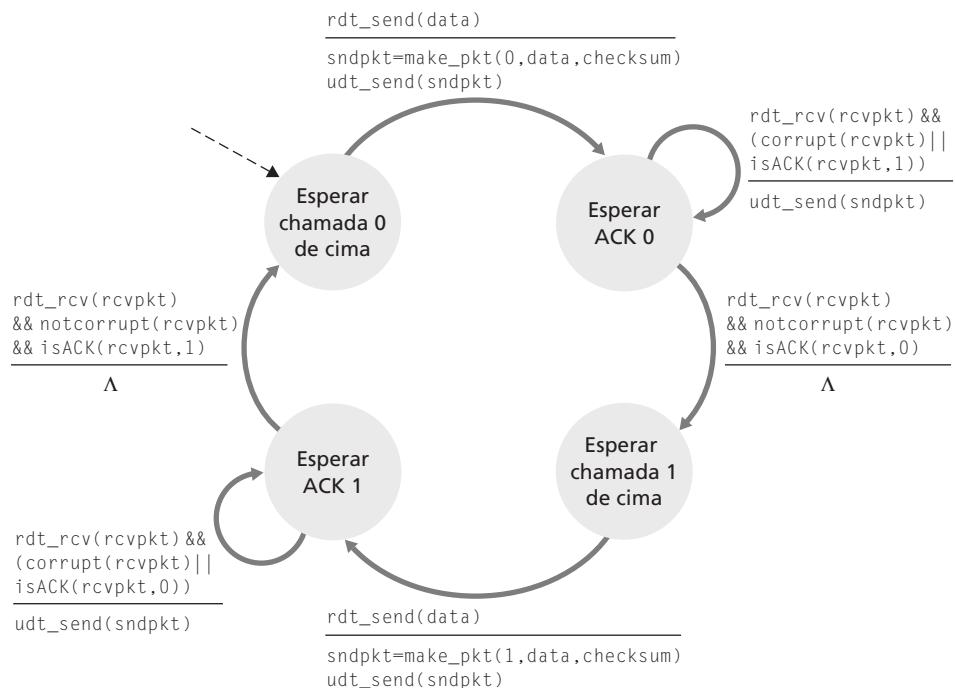


Figura 3.13 rdt2.2 remetente

estiver disposto a esperar o tempo suficiente para ter *certeza* de que o pacote foi perdido, ele poderá simplesmente retransmitir o pacote de dados. É preciso que você se convença de que esse protocolo funciona mesmo.

Mas quanto tempo o remetente precisa esperar para ter certeza de que algo foi perdido? É claro que deve aguardar no mínimo o tempo de um atraso de ida e volta entre ele e o destinatário (o que pode incluir buffers em roteadores ou equipamentos intermediários) e mais o tempo que for necessário para processar um pacote no destinatário. Em muitas redes, o atraso máximo para esses piores casos é muito difícil até de estimar, quanto mais

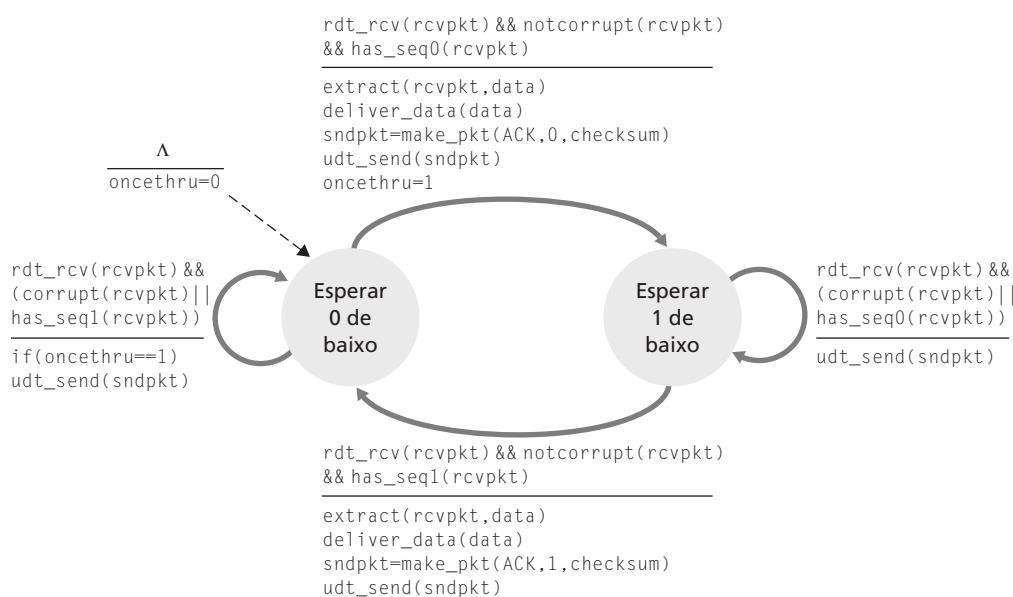


Figura 3.14 rdt2.2 destinatário

saber com certeza. Além disso, o ideal seria que o protocolo se recuperasse da perda de pacotes logo que possível; esperar pelo atraso do pior dos casos pode significar um longo tempo até que a recuperação do erro seja iniciada. Assim, a abordagem adotada na prática é a seguinte: o remetente faz uma escolha ponderada de um valor de tempo dentro do qual seria provável, mas não garantido, que a perda tivesse ocorrido. Se não for recebido um ACK nesse período, o pacote é retransmitido. Note que, se um pacote sofrer um atraso particularmente longo, o remetente poderá retransmiti-lo mesmo que, nem o pacote de dados, nem o seu ACK tenham sido perdidos. Isso introduz a possibilidade de **pacotes de dados duplicados** no canal remetente-destinatário. Felizmente, o protocolo rdt2.2 já dispõe de funcionalidade suficiente (isto é, números de sequência) para tratar dos casos de pacotes duplicados.

Do ponto de vista do remetente, a retransmissão é uma panaceia. O remetente não sabe se um pacote de dados foi perdido, se um ACK foi perdido ou se o pacote ou o ACK simplesmente estavam muito atrasados. Em todos os casos, a ação é a mesma: retransmitir. Para implementar um mecanismo de retransmissão com base no tempo, é necessário um **temporizador de contagem regressiva** que interrompa o processo remetente após ter decorrido um dado tempo. Assim, será preciso que o remetente possa (1) acionar o temporizador todas as vezes que um pacote for enviado (quer seja a primeira vez, quer seja uma retransmissão), (2) responder a uma interrupção feita pelo temporizador (realizando as ações necessárias) e (3) parar o temporizador.

A Figura 3.15 mostra a FSM remetente para o rdt3.0, um protocolo que transfere confiavelmente dados por um canal que pode corromper ou perder pacotes; nos “Exercícios de fixação” pediremos a você que projete a FSM destinatária para rdt3.0. A Figura 3.16 mostra como o protocolo funciona sem pacotes perdidos ou atrasados e como manipula pacotes de dados perdidos. Nessa figura, a passagem do tempo ocorre do topo do diagrama para baixo. Note que o instante de recebimento de um pacote é necessariamente posterior ao instante de envio de um pacote, como resultado de atrasos de transmissão e de propagação. Nas figuras 3.16(b-d), os colchetes do lado remetente indicam os instantes em que o temporizador foi acionado e, mais tarde, os instantes em que ele parou. Vários dos aspectos mais sutis desse protocolo são examinados nos exercícios ao final deste capítulo. Como os números de sequência se alternam entre 0 e 1, o protocolo rdt3.0 às vezes é conhecido como **protocolo bit alternante**.

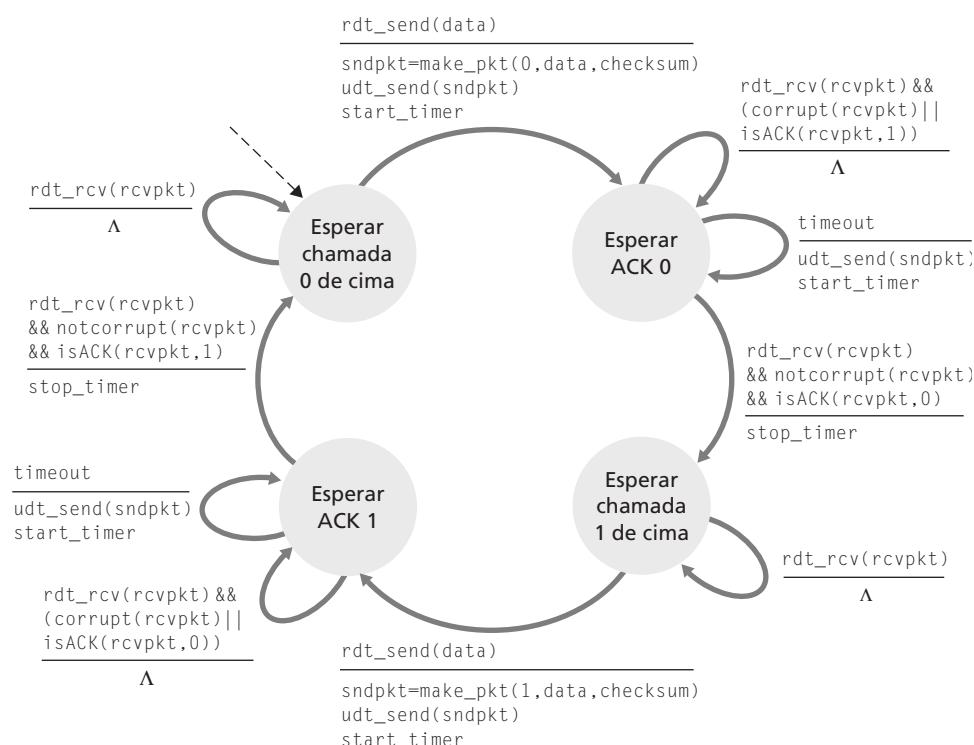


Figura 3.15 rdt3.0 remetente

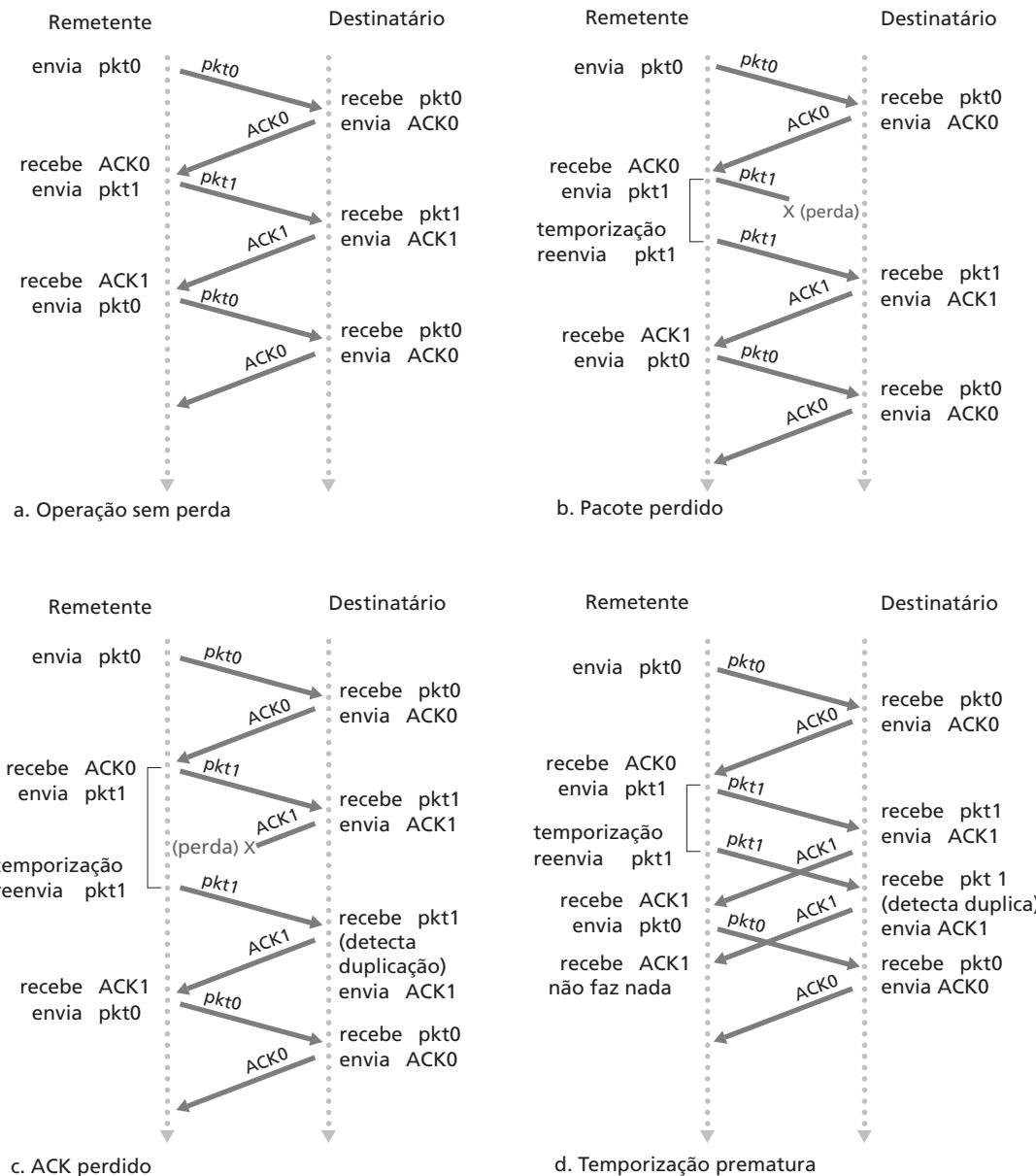


Figura 3.16 Operação do rdt3.0, o protocolo bit alternante

Agora já reunimos os elementos fundamentais de um protocolo de transferência de dados. Somas de verificação, números de sequência, temporizadores e pacotes de reconhecimento negativo e positivo — cada um desempenha um papel crucial e necessário na operação do protocolo. Temos agora em funcionamento um protocolo de transferência confiável de dados!

3.4.2 Protocolos de transferência confiável de dados com paralelismo

O protocolo rdt3.0 é correto em termos funcionais, mas é pouco provável que alguém fique contente com o desempenho dele, particularmente nas redes de alta velocidade de hoje. No coração do problema do desempenho do rdt3.0 está o fato de ele ser um protocolo do tipo pare e espere.

Para avaliar o impacto sobre o desempenho causado pelo comportamento “pare e espere”, considere um caso ideal de dois hospedeiros, um localizado na Costa Oeste dos Estados Unidos e outro na Costa Leste, como mostra a Figura 3.17.

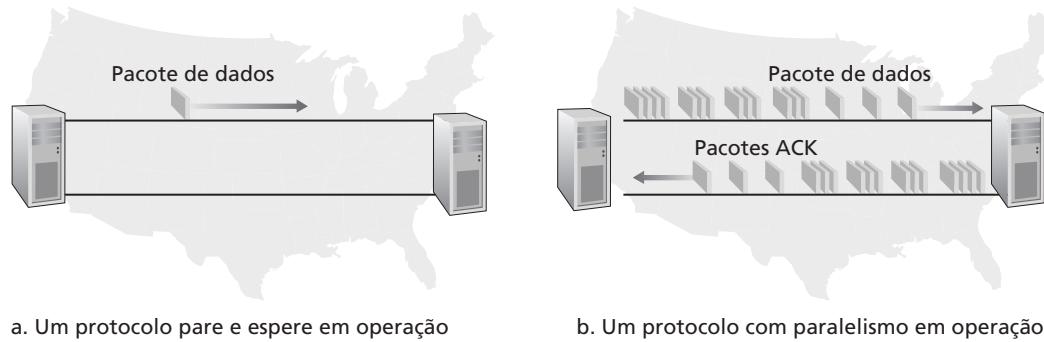


Figura 3.17 Protocolo pare e espere *versus* protocolo com paralelismo

O atraso de propagação de ida e volta à velocidade da luz, T_{prop} , entre esses dois sistemas finais é de aproximadamente 30 milissegundos. Suponha que eles estejam conectados por um canal com capacidade de transmissão, R , de 1 gigabit (10^9 bits) por segundo. Para um tamanho de pacote, L , de 1 kbyte (8 mil bits), incluindo o campo de cabeçalho e também o de dados, o tempo necessário para realmente transmitir o pacote para o enlace de 1 Gbps é:

$$t_{trans} = \frac{L}{R} = \frac{8.000 \text{ bits/pacote}}{10^9 \text{ bits/seg}} = 8 \text{ microssegundos}$$

A Figura 3.18(a) mostra que, com nosso protocolo pare e espere, se o remetente começar a enviar o pacote em $t = 0$, então em $t = L/R = 8$ microssegundos, o último bit entrará no canal do lado remetente. O pacote então faz sua jornada de 15 milissegundos atravessando o país, com o último bit do pacote emergindo no destinatário em $t = RTT/2 + L/R = 15,008$ milissegundos. Supondo, para simplificar, que pacotes ACK sejam extremamente pequenos (para podermos ignorar seu tempo de transmissão) e que o destinatário pode enviar um ACK logo que receber o último bit de um pacote de dados, o ACK emergirá de volta no remetente em $t = RTT + L/R = 30,008$ milissegundos. Nesse ponto, o remetente agora poderá transmitir a próxima mensagem. Assim, em 30,008 milissegundos, o remetente esteve enviando por apenas 0,008 milissegundo. Se definirmos a **utilização** do remetente (ou do canal) como a fração de tempo em que o remetente está realmente ocupado enviando bits para dentro do canal, a análise da Figura 3.18(a) mostra que o protocolo pare e espere tem uma utilização do remetente U_{remet} bastante desanimadora, de:

$$U_{remet} = \frac{L/R}{RTT + L/R} = \frac{0,008}{30,008} = 0,00027$$

Portanto, o remetente ficou ocupado apenas 2,7 centésimos de 1 por cento do tempo! Visto de outra maneira, ele só foi capaz de enviar 1.000 bytes em 30,008 milissegundos, uma vazão efetiva de apenas 267 kbps — mesmo estando disponível um enlace de 1 gigabit por segundo! Imagine o infeliz gerenciador de rede que acabou de pagar uma fortuna para ter capacidade de enlace da ordem de gigabits, mas consegue uma vazão de apenas 267 quilobits por segundo! Este é um exemplo gráfico de como protocolos de rede podem limitar as capacidades oferecidas pelo hardware subjacente de rede. Além disso, desprezamos também os tempos de processamento de protocolo das camadas inferiores no remetente e no destinatário, bem como os atrasos de processamento e de fila que ocorreriam em quaisquer roteadores intermediários existentes entre o remetente e o destinatário. Incluir esses efeitos serviria apenas para aumentar ainda mais o atraso e piorar ainda mais o fraco desempenho.

A solução para esse problema de desempenho em especial é simples: em vez de operar em modo pare e espere, o remetente é autorizado a enviar vários pacotes sem esperar por reconhecimentos, como mostra a Figura 3.17(b). A Figura 3.18(b) mostra que, se um remetente for autorizado a transmitir três pacotes antes de ter de

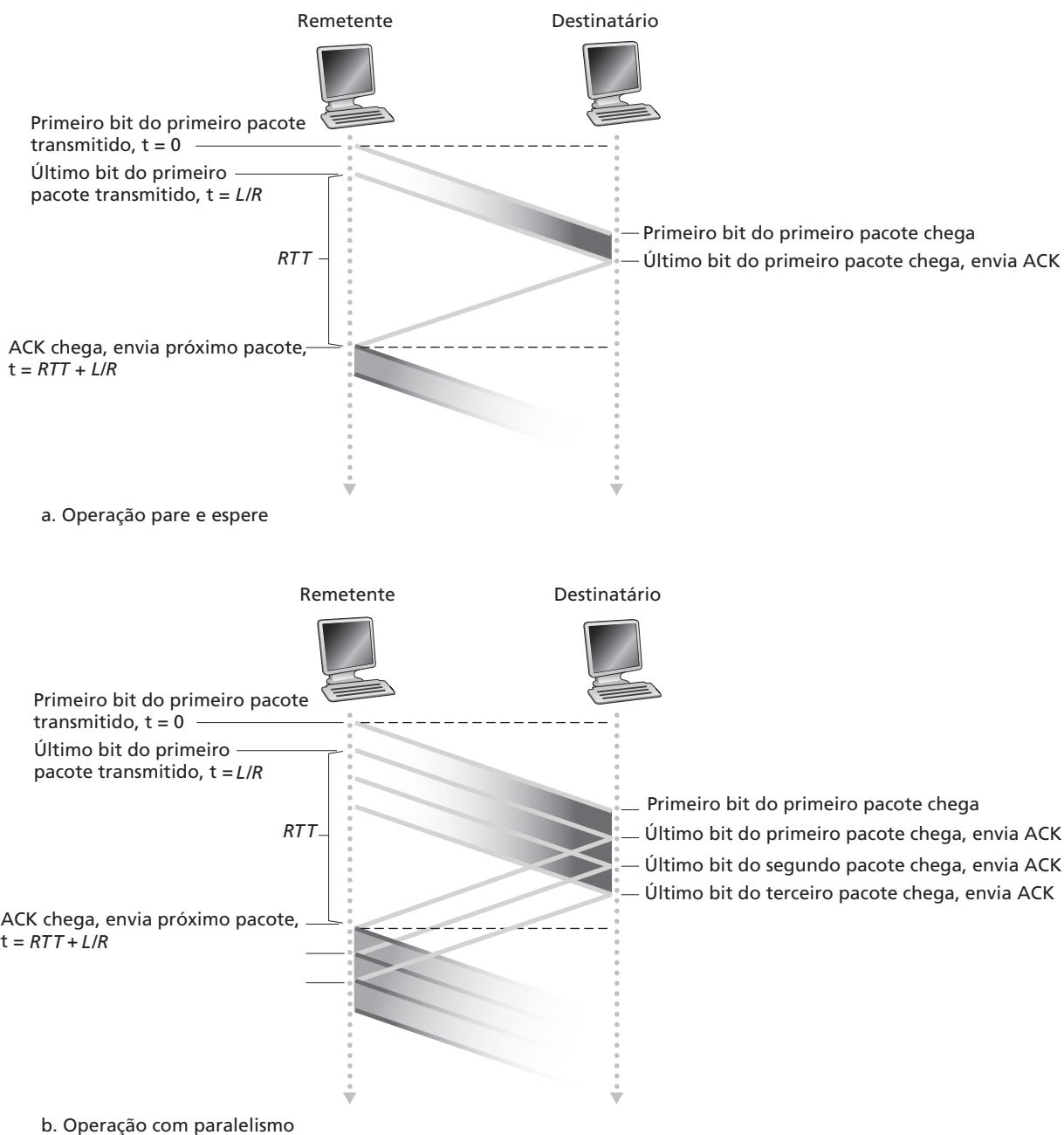


Figura 3.18 Envio com pare e espere e com paralelismo

esperar por reconhecimentos, sua utilização será essencialmente triplicada. Uma vez que os muitos pacotes em trânsito entre remetente e destinatário podem ser visualizados como se estivessem enchendo uma tubulação, essa técnica é conhecida, em inglês, como *pipelining* (tubulação). Porém, como essa expressão é difícil de traduzir para o português, preferimos usar ‘paralelismo’, embora a transmissão de dados seja de fato sequencial. O paralelismo gera as seguintes consequências para protocolos de transferência confiável de dados:

A faixa de números de sequência tem de ser ampliada, uma vez que cada pacote em trânsito (sem contar as retransmissões) precisa ter um número de sequência exclusivo e pode haver vários pacotes não reconhecidos em trânsito.

Os lados remetente e destinatário dos protocolos podem ter de reservar buffers para mais de um pacote. No mínimo, o remetente terá de providenciar buffers para pacotes que foram transmitidos, mas que ainda não foram reconhecidos. O buffer de pacotes corretamente recebidos pode também ser necessário no destinatário, como discutiremos a seguir.

A faixa de números de sequência necessária e as necessidades de buffer dependerão da maneira como um protocolo de transferência de dados responde a pacotes perdidos, corrompidos e demasiadamente atrasados. Duas abordagens básicas em relação à recuperação de erros com paralelismo podem ser identificadas: **Go-Back-N** e **repetição seletiva**.

3.4.3 Go-Back-N

Em um protocolo **Go-Back-N** (GBN), o remetente é autorizado a transmitir múltiplos pacotes (se disponíveis) sem esperar por um reconhecimento, mas fica limitado a ter não mais do que algum número máximo permitido, N , de pacotes não reconhecidos. Nesta seção, descreveremos o protocolo GBN com detalhes. Mas antes de continuar a leitura, convidamos você para se divertir com o applet GBN (incrível!) no Companion site Web.

A Figura 3.19 mostra a visão que o remetente tem da faixa de números de sequência em um protocolo GBN.

Se definirmos base como o número de sequência do mais antigo pacote não reconhecido e nextseqnum como o menor número de sequência não utilizado (isto é, o número de sequência do próximo pacote a ser enviado), então quatro intervalos na faixa de números de sequência poderão ser identificados. Os números de sequência no intervalo $[0, \text{base}-1]$ correspondem aos pacotes que já foram transmitidos e reconhecidos. O intervalo $[\text{base}, \text{nextseqnum}-1]$ corresponde aos pacotes que foram enviados, mas ainda não foram reconhecidos. Os números de sequência no intervalo $[\text{nextseqnum}, \text{base}+N-1]$ podem ser usados para pacotes que podem ser enviados imediatamente, caso cheguem dados vindos da camada superior. Finalmente, números de sequência maiores ou iguais a $\text{base}+N$ não podem ser usados até que um pacote não reconhecido que esteja pendente seja reconhecido (especificamente, o pacote cujo número de sequência é base).

Como sugere a Figura 3.19, a faixa de números de sequência permitidos para pacotes transmitidos mas ainda não reconhecidos pode ser vista como uma ‘janela’ de N sobre a faixa de números de sequência. À medida que o protocolo opera, essa janela se desloca para a frente sobre o espaço de números de sequência. Por essa razão, N é frequentemente denominado **tamanho de janela** e o protocolo GBN em si, **protocolo de janela deslizante** (*sliding-window protocol*). É possível que você esteja pensando que razão teríamos, em primeiro lugar, para limitar o número de pacotes pendentes não reconhecidos a um valor N . Por que não permitir um número ilimitado desses pacotes? Veremos na Seção 3.5 que o controle de fluxo é uma das razões para impor um limite ao remetente. Examinaremos outra razão para isso na Seção 3.7, quando estudarmos o controle de congestionamento do TCP.

Na prática, o número de sequência de um pacote é carregado em um campo de comprimento fixo no cabeçalho do pacote. Se k for o número de bits no campo de número de sequência do pacote, a faixa de números de sequência será então $[0, 2^k - 1]$. Com uma faixa finita de números de sequência, toda a aritmética que envolver números de sequência deverá ser feita usando aritmética de módulo 2^k . (Em outras palavras, o espaço do número de sequência pode ser imaginado como um anel de tamanho 2^k , em que o número de sequência $2^k - 1$

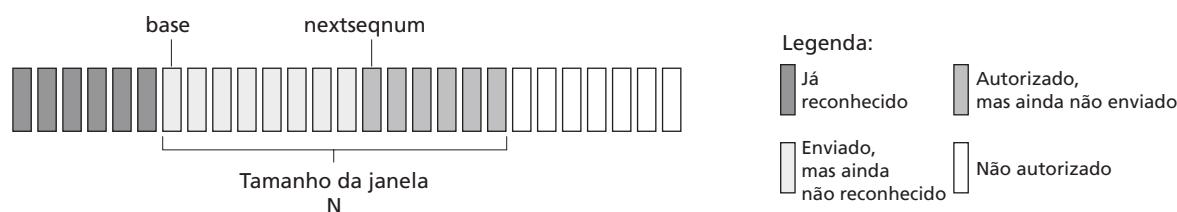


Figura 3.19 Visão do remetente para os números de sequência no protocolo Go-Back-N

é imediatamente seguido pelo número de sequência 0.) Lembre-se de que `rdt3.0` tem um número de sequência de 1 bit e uma faixa de números de sequência de [0,1]. Vários problemas ao final deste capítulo tratam das consequências de uma faixa finita de números de sequência. Veremos na Seção 3.5 que o TCP tem um campo de número de sequência de 32 bits, onde os números de sequência do TCP contam bytes na cadeia de bytes, em vez de pacotes.

As figuras 3.20 e 3.21 descrevem uma FSM estendida dos lados remetente e destinatário de um protocolo GBN baseado em ACK, mas sem NAK.

Referimo-nos a essa descrição de FSM como *FSM estendida* porque adicionamos variáveis (semelhantes às variáveis de linguagem de programação) para `base` e `nextseqnum`; também adicionamos operações sobre essas variáveis e ações condicionais que as envolvem. Note que a especificação da FSM estendida agora está começando a parecer um pouco com uma especificação de linguagem de programação. [Bochman, 1984] fornece um excelente levantamento sobre extensões adicionais às técnicas FSM, bem como sobre outras técnicas para especificação de protocolos baseadas em linguagens.

O remetente GBN deve responder a três tipos de eventos:

Chamada vinda de cima. Quando `rdt_send()` é chamado de cima, o remetente primeiramente verifica se a janela está cheia, isto é, se há N pacotes pendentes não reconhecidos. Se a janela não estiver cheia, um pacote é criado e enviado e as variáveis são adequadamente atualizadas. Se estiver cheia, o remetente apenas devolve os dados à camada superior — uma indicação implícita de que a janela está cheia. Presumivelmente, a camada superior então teria de tentar outra vez mais tarde. Em uma implementação real, o remetente muito provavelmente teria colocado esses dados em um buffer (mas não os teria enviado imediatamente) ou teria um mecanismo de sincronização (por exemplo, um semáforo ou uma flag) que permitiria que a camada superior chamassem `rdt_send()` somente quando as janelas não estivessem cheias.

Recebimento de um ACK. Em nosso protocolo GBN, um reconhecimento de pacote com número de sequência n seria tomado como um **reconhecimento cumulativo**, indicando que todos os pacotes com número de sequência até e inclusive n tinham sido corretamente recebidos no destinatário. Voltaremos a esse assunto em breve, quando examinarmos o lado destinatário do GBN.

Um esgotamento de temporização. O nome ‘Go-Back-N’ deriva do comportamento do remetente em relação a pacotes perdidos ou demasiadamente atrasados. Como no protocolo pare e espere, um temporizador é usado para recuperar a perda de dados ou reconhecer pacotes. Se ocorrer o esgotamento da temporização, o remetente reenvia todos os pacotes que tinham sido previamente enviados mas que ainda não tinham sido reconhecidos. Nosso remetente da Figura 3.20 usa apenas um único temporizador, que pode ser imaginado como um temporizador para o mais antigo pacote já transmitido mas que ainda não foi reconhecido. Se for recebido um ACK e ainda houver pacotes adicionais transmitidos mas ainda não reconhecidos, o temporizador será reiniciado. Se não houver nenhum pacote pendente não reconhecido, o temporizador será desligado.

As ações do destinatário no GBN também são simples. Se um pacote com número de sequência n for recebido corretamente e estiver na ordem (isto é, os últimos dados entregues à camada superior vierem de um pacote com número de sequência $n - 1$), o destinatário enviará um ACK para o pacote n e entregará a porção dos dados do pacote à camada superior. Em todos os outros casos, o destinatário descarta o pacote e reenvia um ACK para o pacote mais recente que foi recebido na ordem correta. Dado que os pacotes são entregues à camada superior um por vez, se o pacote k tiver sido recebido e entregue, então todos os pacotes com número de sequência menores do que k também terão sido entregues. Assim, o uso de reconhecimentos cumulativos é uma escolha natural para o GBN.

Em nosso protocolo GBN, o destinatário descarta os pacotes que chegam fora de ordem. Embora pareça bobagem e perda de tempo descartar um pacote corretamente recebido (mas fora de ordem), existem justificativas para isso. Lembre-se de que o destinatário deve entregar dados na ordem certa à camada superior. Suponha agora que o pacote n esteja sendo esperado, mas quem chega é o pacote $n + 1$. Como os dados devem ser entregues na ordem certa, o destinatário poderia conservar o pacote $n + 1$ no buffer (salvá-lo) e entregar esse pacote à camada

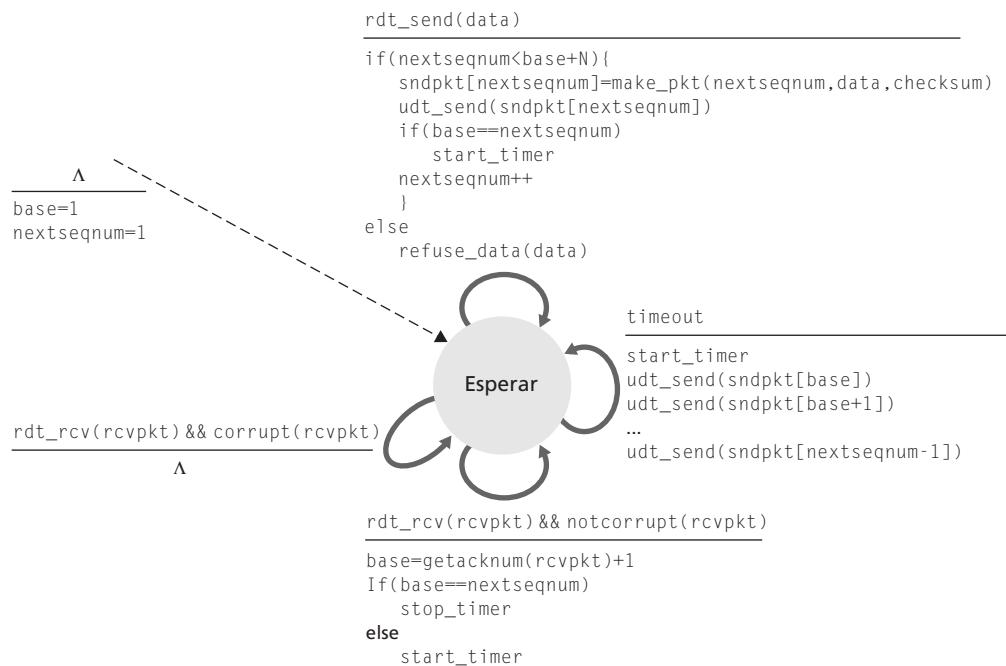


Figura 3.20 Descrição da FSM estendida do remetente GBN

superior mais tarde, após ter recebido o pacote n . Contudo, se o pacote n for perdido, os pacotes n e $n + 1$ serão ambos finalmente retransmitidos como resultado da regra de retransmissão do GBN no remetente. Assim, o destinatário pode simplesmente descartar o pacote $n + 1$. A vantagem dessa abordagem é a simplicidade da manipulação de buffers no destinatário — o destinatário não precisa colocar no buffer nenhum pacote que esteja fora de ordem. Desse modo, enquanto o remetente deve manter os limites superior e inferior de sua janela e a posição de `nextseqnum` dentro dessa janela, a única informação que o destinatário precisa manter é o número de sequência do próximo pacote esperado conforme a ordem. Esse valor é retido na variável `expectedseqnum` mostrada na FSM destinatária da Figura 3.21. Evidentemente, a desvantagem de jogar fora um pacote recebido corretamente é que a retransmissão subsequente desse pacote pode ser perdida ou ficar truncada, caso em que ainda mais retransmissões seriam necessárias.

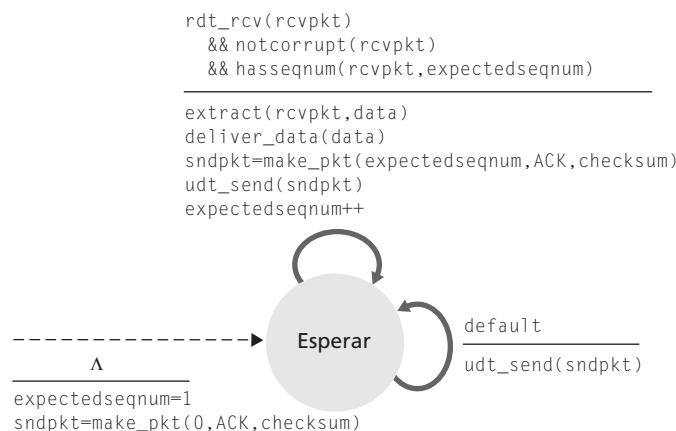


Figura 3.21 Descrição da FSM estendida do destinatário GBN

A Figura 3.22 mostra a operação do protocolo GBN para o caso de um tamanho de janela de quatro pacotes. Por causa da limitação do tamanho dessa janela, o remetente envia os pacotes de 0 a 3, mas, em seguida, tem de esperar que um ou mais desses pacotes sejam reconhecidos antes de prosseguir. E, à medida que cada ACK sucesivo (por exemplo, ACK0 e ACK1) é recebido, a janela se desloca para a frente e o remetente pode transmitir um novo pacote (pkt4 e pkt5, respectivamente). Do lado destinatário, o pacote 2 é perdido. Desse modo, verifica-se que os pacotes 3, 4 e 5 estão fora de ordem e, portanto, são descartados.

Antes de encerrarmos nossa discussão sobre o GBN, devemos ressaltar que uma implementação desse protocolo em uma pilha de protocolo provavelmente seria estruturada de modo semelhante à da FSM estendida da Figura 3.20. A implementação provavelmente também seria estruturada sob a forma de vários procedimentos que implementam as ações a serem executadas em resposta aos vários eventos que podem ocorrer. Nessa **programação baseada em eventos**, os vários procedimentos são chamados (invocados) por outros procedimentos presentes na pilha de protocolo ou como resultado de uma interrupção. No remetente, esses eventos seriam: (1) uma chamada pela entidade da camada superior invocando `rdt_send()`, (2) uma interrupção pelo temporizador e (3) uma chamada pela camada inferior invocando `rdt_rcv()` quando chega um pacote. Os exercícios de programação ao final deste capítulo lhe darão a chance de implementar de verdade essas rotinas em um ambiente de rede simulado, mas realista.

Salientamos que o protocolo GBN incorpora quase todas as técnicas que encontraremos quando estudarmos, na Seção 3.5, os componentes de transferência confiável de dados do TCP. Essas técnicas incluem a utilização de números de sequência, reconhecimentos cumulativos, somas de verificação e uma operação de esgotamento de temporização/retransmissão.

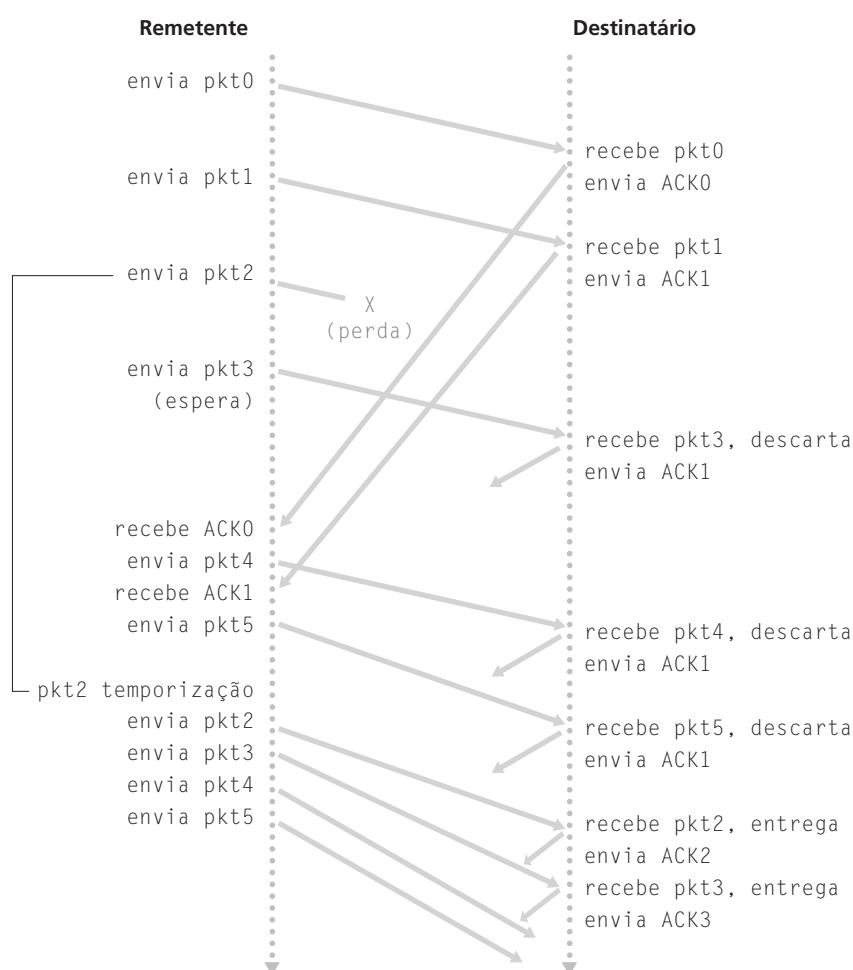


Figura 3.22 GBN em operação

3.4.4 Repetição seletiva (SR)

O protocolo GBN permite que o remetente potencialmente “encha a rede” com pacotes na Figura 3.17, evitando, assim, os problemas de utilização de canal observados em protocolos do tipo pare e espere. Há, contudo, casos em que o próprio GBN sofre com problemas de desempenho. Em especial, quando o tamanho da janela e o produto entre o atraso e a largura de banda são grandes, pode haver muitos pacotes pendentes na rede. Assim, um único erro de pacote pode fazer com que o GBN retransmita um grande número de pacotes — muitos deles desnecessariamente. À medida que aumenta a probabilidade de erros no canal, a rede pode ficar lotada com essas retransmissões desnecessárias. Imagine se, em uma conversa, toda vez que uma palavra fosse pronunciada de maneira truncada as outras mil que a circundam (por exemplo, um tamanho de janela de mil palavras) tivessem de ser repetidas. A conversa sofreria atrasos devido a todas essas palavras reiteradas.

Como o próprio nome sugere, protocolos de repetição seletiva (*selective repeat* — SR) evitam retransmissões desnecessárias porque fazem o remetente retransmitir somente os pacotes suspeitos de terem sido recebidos com erro (isto é, que foram perdidos ou corrompidos) no destinatário. Essa retransmissão individual, somente quando necessária, exige que o destinatário reconheça *individualmente* os pacotes recebidos de modo correto. Uma janela de tamanho N será usada novamente para limitar o número de pacotes pendentes não reconhecidos dentro da rede. Contudo, ao contrário do GBN, o remetente já terá recebido ACKs para alguns dos pacotes na janela. A Figura 3.23 mostra a visão que o protocolo de SR remetente tem do espaço do número de sequência. A Figura 3.24 detalha as várias ações executadas pelo protocolo SR remetente.

O protocolo SR destinatário reconhecerá um pacote corretamente recebido esteja ele ou não na ordem certa. Pacotes fora de ordem ficam no buffer até que todos os pacotes faltantes (isto é, os que têm números de sequência menores) sejam recebidos, quando então um conjunto de pacotes poderá ser entregue à camada superior na ordem correta. A Figura 3.25 apresenta as várias ações realizadas pelo protocolo SR destinatário.

A Figura 3.26 apresenta um exemplo de operação do protocolo SR quando ocorre perda de pacotes. Note que, nessa figura, o destinatário inicialmente armazena os pacotes 3, 4 e 5 e os entrega juntamente com o pacote 2 à camada superior, quando o pacote 2 é finalmente recebido.

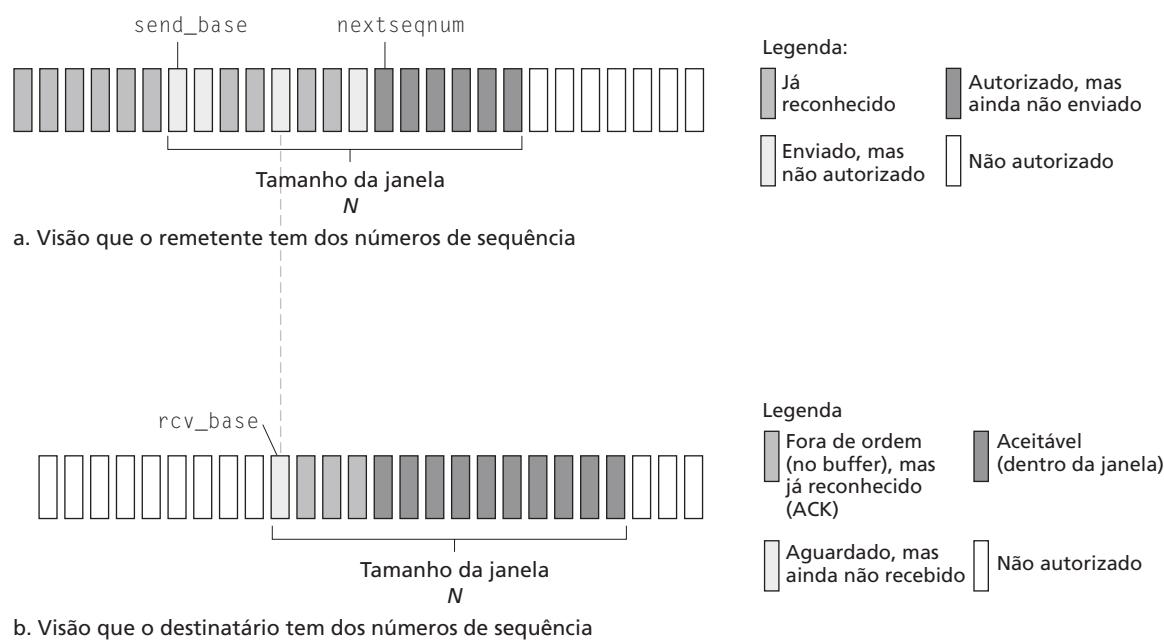


Figura 3.23 Visões que os protocolos SR remetente e destinatário têm do espaço de número de sequência

1. Dados recebidos de cima. Quando são recebidos dados de cima, o protocolo SR remetente verifica o próximo número de sequência disponível para o pacote. Se o número de sequência está dentro da janela do remetente, os dados são empacotados e enviados; do contrário, eles são armazenados ou devolvidos à camada superior para transmissão posterior, como acontece no GBN.
2. Esgotamento de temporização. Novamente são usados temporizadores para proteção contra perda de pacotes. Contudo, cada pacote agora deve ter seu próprio temporizador lógico, já que apenas um pacote será transmitido quando a temporização se esgotar. Um único hardware de temporizador pode ser usado para emular a operação de múltiplos temporizadores lógicos [Varghese, 1997].
3. ACK recebido. Se for recebido um ACK, o SR remetente marcará aquele pacote como recebido, contanto que esteja na janela. Se o número de sequência do pacote for igual a `send_base`, a base da janela se deslocará para a frente até o pacote não reconhecido que tiver o menor número de sequência. Se a janela se deslocar e houver pacotes não transmitidos com números de sequência que agora caem dentro da janela, esses pacotes serão transmitidos.

Figura 3.24 Eventos e ações do protocolo SR remetente

É importante notar que na etapa 2 da Figura 3.25 o destinatário reconhece novamente (em vez de ignorar) pacotes já recebidos com certos números de sequência que estão *abaixo* da atual base da janela. É bom que você se convença de que esse reconhecimento duplo é de fato necessário. Dados os espaços dos números de sequência do remetente e do destinatário na Figura 3.23, por exemplo, se não houver ACK para pacote com número `send_base` propagando-se do destinatário ao remetente, este acabará retransmitindo o pacote `send_base`, embora esteja claro (para nós, e não para o remetente!) que o destinatário já recebeu esse pacote. Caso o destinatário não reconhecesse esse pacote, a janela do remetente jamais se deslocaria para a frente! Esse exemplo ilustra um importante aspecto dos protocolos SR (e também de muitos outros). O remetente e o destinatário nem sempre têm uma visão idêntica do que foi recebido corretamente e do que não foi. Para protocolos SR, isso significa que as janelas do remetente e do destinatário nem sempre coincidirão.

A falta de sincronização entre as janelas do remetente e do destinatário tem importantes consequências quando nos defrontamos com a realidade de uma faixa finita de números de sequência. Considere o que poderia acontecer, por exemplo, com uma faixa finita de quatro números de sequência de pacotes (0, 1, 2, 3) e um tamanho de janela de três. Suponha que os pacotes de 0 a 2 sejam transmitidos, recebidos e reconhecidos corretamente no destinatário. Nesse ponto, a janela do destinatário está sobre o quarto, o quinto e o sexto pacotes, que têm os números de sequência 3, 0 e 1, respectivamente. Agora, considere dois cenários. No primeiro, mostrado na Figura

1. Pacote com número de sequência no intervalo $[rcv_base, rcv_base+N-1]$ foi corretamente recebido. Nesse caso, o pacote recebido cai dentro da janela do destinatário e um pacote ACK seletivo é devolvido ao remetente. Se o pacote não tiver sido recebido anteriormente, irá para o buffer. Se esse pacote tiver um número de sequência igual à base da janela destinatária (`rcv_base` na Figura 3.22), então ele, e quaisquer outros pacotes anteriormente armazenados no buffer e numerados consecutivamente (começando com `rcv_base`), serão entregues à camada superior. A janela destinatária é então deslocada para a frente de acordo com o número de pacotes entregues à camada superior. Como exemplo, considere a Figura 3.26. Quando um pacote com número de sequência `rcv_base=2` é recebido, ele e os pacotes 3, 4 e 5 podem ser entregues à camada superior.
2. Pacote com número de sequência em $[rcv_base-N, rcv_base-1]$ é recebido. Nesse caso, um ACK deve ser gerado mesmo que esse pacote já tenha sido reconhecido anteriormente pelo destinatário.
3. Qualquer outro. Ignore o pacote

Figura 3.25 Eventos e ações do protocolo SR destinatário

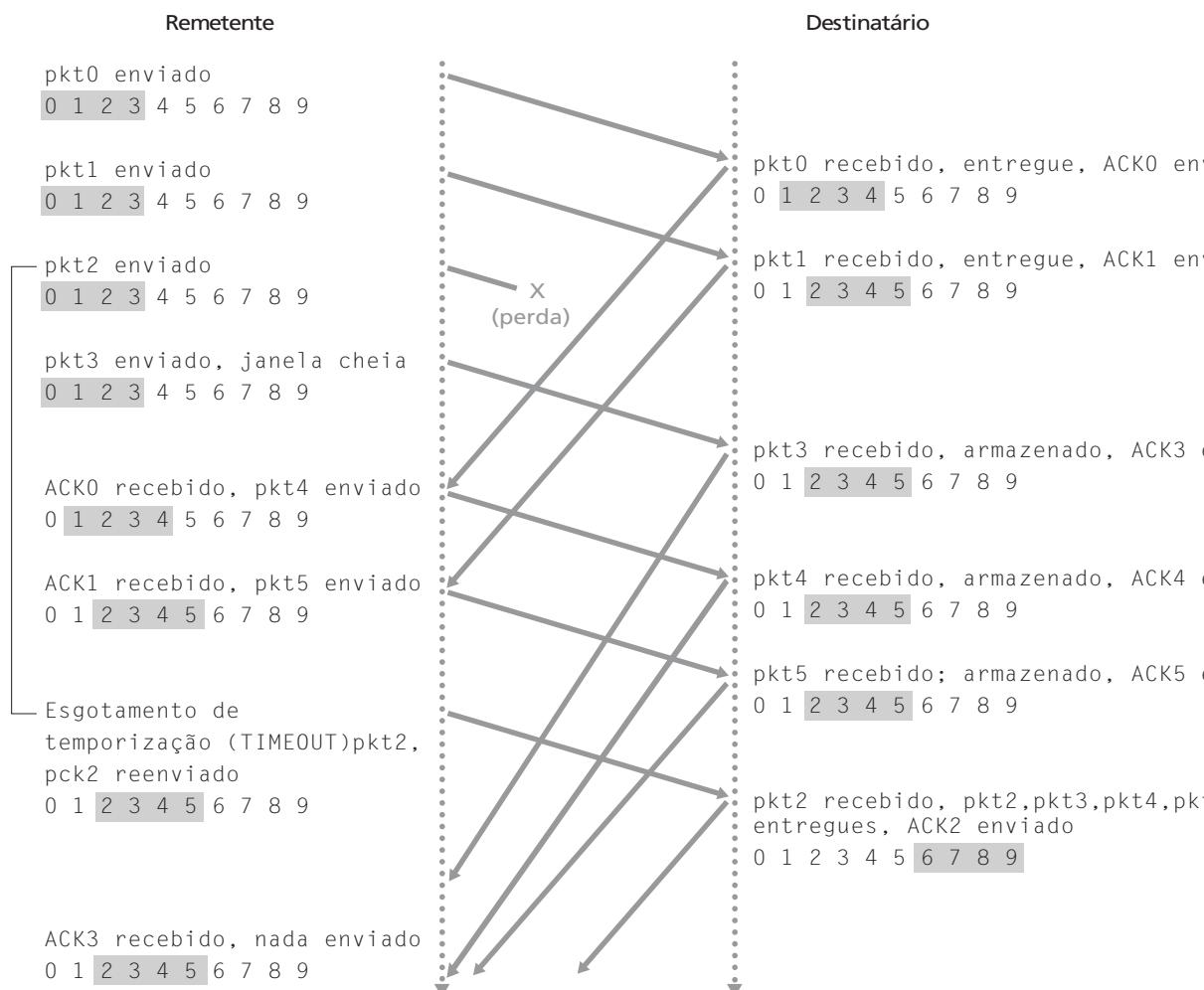


Figura 3.26 Operação SR

3.27(a), os ACKs para os três primeiros pacotes foram perdidos e o remetente retransmite esses pacotes. Assim, o que o destinatário recebe em seguida é um pacote com o número de sequência 0 — uma cópia do primeiro pacote enviado.

No segundo cenário, mostrado na Figura 3.27(b), os ACKs para os três primeiros pacotes foram entregues corretamente. Assim, o remetente desloca sua janela para a frente e envia o quarto, o quinto e o sexto pacotes com os números de sequência 3, 0 e 1, respectivamente. O pacote com o número de sequência 3 é perdido, mas o pacote com o número de sequência 0 chega — um pacote que contém dados novos.

Agora, na Figura 3.27, considere o ponto de vista do destinatário, que tem uma cortina imaginária entre o remetente e ele, já que o destinatário não pode ‘ver’ as ações executadas pelo remetente. Tudo o que o destinatário observa é a sequência de mensagens que ele recebe do canal e envia para o canal. No que concerne a ele, os dois cenários da Figura 3.27 são *idênticos*. Não há nenhum modo de distinguir a retransmissão do primeiro pacote da transmissão original do quinto pacote. Fica claro que um tamanho de janela que seja igual ao tamanho do espaço de numeração sequencial menos 1 não vai funcionar. Mas qual deve ser o tamanho da janela? Um problema ao final deste capítulo pede que você demonstre que o tamanho da janela pode ser menor ou igual à metade do tamanho do espaço de numeração sequencial para os protocolos SR.

No Companion Website, você encontrará um applet que anima a operação do protocolo SR. Tente realizar os mesmos experimentos feitos com o applet GBN. Os resultados combinam com o que você espera?

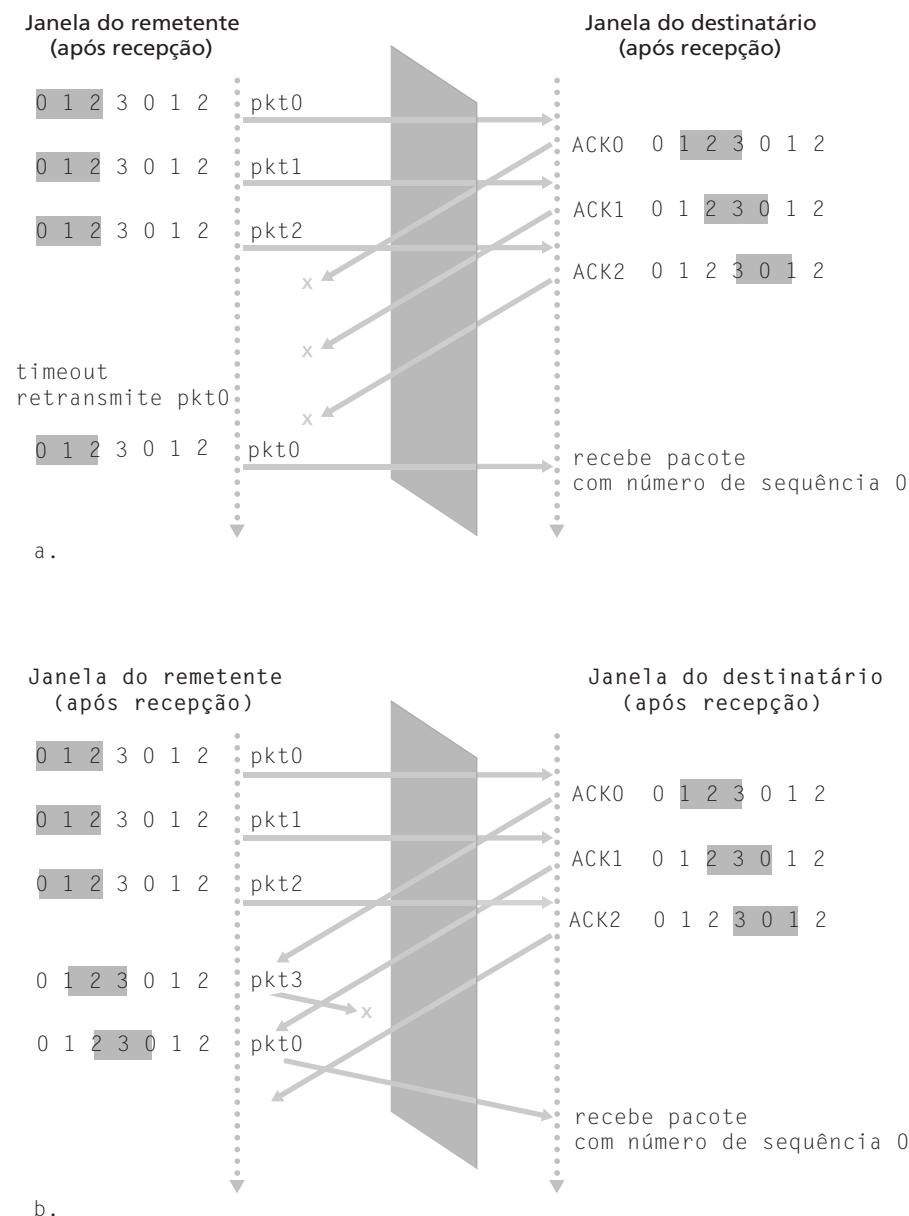


Figura 3.27 Dilema do remetente SR com janelas muito grandes: um novo pacote ou uma retransmissão?

Com isso, concluímos nossa discussão sobre protocolos de transferência confiável de dados. Percorremos um longo caminho e apresentamos numerosos mecanismos que, juntos, proveem transferência confiável de dados. A Tabela 3.1 resume esses mecanismos. Agora que já vimos todos eles em operação e podemos enxergar ‘o quadro geral’, aconselhamos que você leia novamente esta seção para perceber como esses mecanismos foram adicionados pouco a pouco, de modo a abordar modelos (realistas) de complexidade crescente do canal que conecta o remetente ao destinatário ou para melhorar o desempenho dos protocolos.

Encerraremos nossa explanação considerando uma premissa remanescente em nosso modelo de canal subjacente. Lembre-se de que admitimos que pacotes não podem ser reordenados dentro do canal entre o remetente e o destinatário. Esta é uma premissa em geral razoável quando o remetente e o destinatário estão conectados por um único fio físico. Contudo, quando o ‘canal’ que está conectando os dois é uma rede, pode ocorrer reordenação de pacotes. Uma manifestação da reordenação de pacotes é que podem aparecer cópias antigas de um pacote com

Mecanismo	Utilização, Comentários
Soma de verificação	Usada para detectar erros de bits em um pacote transmitido.
Temporizador	Usado para controlar a temporização/retransmitir um pacote, possivelmente porque o pacote (ou seu ACK) foi perdido dentro do canal. Como pode ocorrer esgotamento de temporização quando um pacote está atrasado, mas não perdido (exgotamento de temporização prematuro), ou quando um pacote foi recebido pelo destinatário mas o ACK remetente-destinatário foi perdido, um destinatário pode receber cópias duplicadas de um pacote.
Número de sequência	Usado para numeração sequencial de pacotes de dados que transitam do remetente ao destinatário. Lacunas nos números de sequência de pacotes recebidos permitem que o destinatário detecte um pacote perdido. Pacotes com números de sequência duplicados permitem que o destinatário detecte cópias duplicadas de um pacote.
Reconhecimento	Usado pelo destinatário para avisar o remetente que um pacote ou conjunto de pacotes foi recebido corretamente. Reconhecimentos normalmente portam o número de sequência do pacote, ou pacotes, que estão sendo reconhecidos. Reconhecimentos podem ser individuais ou cumulativos, dependendo do protocolo.
Reconhecimento negativo	Usado pelo destinatário para avisar o remetente que um pacote não foi recebido corretamente. Reconhecimentos negativos normalmente portam o número de sequência do pacote que não foi recebido corretamente.
Janela, paralelismo	O remetente pode ficar restrito a enviar somente pacotes com números de sequência que caiam dentro de uma determinada faixa. Permitindo que vários pacotes sejam transmitidos, ainda que não reconhecidos, a utilização do remetente pode ser aumentada em relação ao modo de operação pare e espere. Em breve veremos que o tamanho da janela pode ser estabelecido com base na capacidade do destinatário receber e fazer buffer de mensagens ou no nível de congestionamento na rede, ou em ambos.

Tabela 3.1 Resumo de mecanismos de transferência confiável de dados e sua utilização

número de sequência ou de reconhecimento x , mesmo que nem a janela do remetente nem a do destinatário contenham x . Com a reordenação de pacotes, podemos considerar que o canal essencialmente usa armazenamento de pacotes e emite-os espontaneamente em algum momento qualquer do futuro. Como números de sequência podem ser reutilizados, devemos tomar um certo cuidado para nos prevenir contra esses pacotes duplicados. A abordagem adotada na prática é garantir que um número de sequência não seja reutilizado até que o remetente esteja ‘certo’ de que nenhum pacote enviado anteriormente com número de sequência x está na rede. Isso é feito admitindo que um pacote não pode ‘viver’ na rede mais do que um certo tempo máximo fixado. As extensões do TCP para redes de alta velocidade [RFC 1323] usam um tempo de vida máximo de pacote de aproximadamente três minutos. [Sunshine, 1978] descreve um método para usar números de sequência tais que os problemas de reordenação podem ser completamente evitados.

3.5 Transporte orientado para conexão: TCP

Agora que já vimos os princípios subjacentes à transferência confiável de dados, vamos voltar para o TCP — o protocolo de transporte confiável da camada de transporte, orientado para conexão, da Internet. Nesta seção, veremos que, para poder fornecer transferência confiável de dados, o TCP conta com muitos dos princípios subjacentes discutidos na seção anterior, incluindo detecção de erro, retransmissões, reconhecimentos cumulativos, temporizadores e campos de cabeçalho para números de sequência e de reconhecimento. O TCP está definido nos RFCs 793, 1122, 1323, 2018 e 2581.

3.5.1 A conexão TCP

Dizemos que o TCP é **orientado para conexão** porque, antes que um processo de aplicação possa começar a enviar dados a outro, os dois processos precisam primeiramente se ‘apresentar’ — isto é, devem enviar alguns segmentos preliminares um ao outro para estabelecer os parâmetros da transferência de dados em questão. Como parte do estabelecimento da conexão TCP, ambos os lados da conexão iniciarão muitas “variáveis de estado” (muitas das quais serão discutidas nesta seção e na Seção 3.7) associadas com a conexão TCP.

A ‘conexão’ TCP não é um circuito TDM ou FDM fim a fim, como acontece em uma rede de comutação de circuitos. Tampouco é um circuito virtual (veja o Capítulo 1), pois o estado de conexão reside inteiramente nos dois sistemas finais. Como o protocolo TCP roda somente nos sistemas finais, e não nos elementos intermediários da rede (roteadores e comutadores de camada de enlace), os elementos intermediários não mantêm estado de conexão TCP. Na verdade, os roteadores intermediários são completamente alheios às conexões TCP; eles enxergam datagramas, e não conexões.

Uma conexão TCP provê um **serviço full-duplex**: se houver uma conexão TCP entre o processo A em um hospedeiro e o processo B em outro hospedeiro, então os dados da camada de aplicação poderão fluir de A para B ao mesmo tempo em que os dados da camada de aplicação fluem de B para A. A conexão TCP é sempre **ponto a ponto**, isto é, entre um único remetente e um único destinatário. O chamado ‘multicast’ (veja a Seção 4.7) — a transferência de dados de um remetente para vários destinatários em uma única operação de envio — não é possível com o TCP. Com o TCP, dois hospedeiros é bom; três é demais!

Vamos agora examinar como uma conexão TCP é estabelecida. Suponha que um processo que roda em um hospedeiro queira iniciar a conexão com outro processo em outro hospedeiro. Lembre-se de que o processo que está iniciando a conexão é denominado *processo cliente*, enquanto o outro processo é denominado *processo servidor*. O processo de aplicação cliente primeiramente informa à camada de transporte no cliente que ele quer estabelecer uma conexão com um processo no servidor. Lembre-se (Seção 2.7) de que um programa cliente em Java faz isso emitindo o comando

```
Socket clientSocket = new Socket ("hostname", portNumber);
```

em que *hostname* é o nome do servidor e *portNumber* identifica o processo no servidor. A camada de transporte no cliente então passa a estabelecer uma conexão TCP-servidor. Discutiremos com algum detalhe o procedimento de estabelecimento de conexão ao final desta seção. Por enquanto, basta saber que o cliente primeiramente envia um segmento TCP especial; o servidor responde com um segundo segmento TCP especial e, por fim, o cliente responde novamente com um terceiro segmento especial. Os primeiros dois segmentos não contêm nenhuma “carga útil”, isto é, nenhum dado da camada de aplicação; o terceiro desses segmentos pode carregar uma carga útil. Como três segmentos são enviados entre dois hospedeiros, esse procedimento de estabelecimento de conexão é frequentemente denominado **apresentação de três vias** (*3-way handshake*).

Uma vez estabelecida uma conexão TCP, os dois processos de aplicação podem enviar dados um para o outro. Vamos considerar o envio de dados do processo cliente para o processo servidor. O processo cliente passa uma cadeia de dados através do socket (a porta do processo), como descrito na Seção 2.7. Tão logo passem pelo socket, os dados estão nas mãos do TCP que está rodando no cliente. Como mostra a Figura 3.28, o TCP direciona seus dados para o **buffer de envio** da conexão, que é um dos buffers reservados durante a apresentação de três vias inicial. De quando em quando, o TCP arranca grandes pedaços de dados do buffer de envio. O interessante é que a especificação do TCP [RFC 793] é muito lacônica ao indicar quando o TCP deve realmente enviar dados que estão nos buffers, determinando apenas que o TCP “deve enviar aqueles dados em segmentos segundo sua própria conveniência”. A quantidade máxima de dados que pode ser retirada e colocada em um segmento é

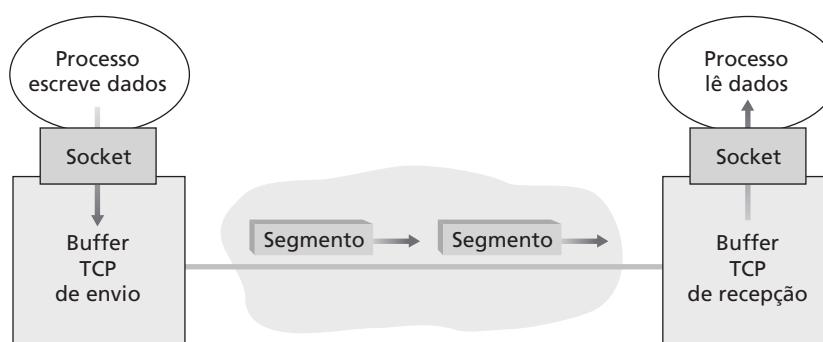


Figura 3.28 Buffers TCP de envio e de recepção

História

Vinton Cerf, Robert Kahn e TCP/IP

No início da década de 1970, as redes de comutação de pacotes começaram a proliferar. A ARPAnet — precursora da Internet — era apenas mais uma dentre tantas redes, que tinham, cada uma, seu próprio protocolo. Dois pesquisadores, Vinton Cerf e Robert Kahn, reconheceram a importância de interconectar essas redes e inventaram um protocolo inter-redes denominado TCP/IP, que quer dizer Transmission Control Protocol/Internet Protocol (protocolo de controle de transmissão/protocolo da Internet). Embora no começo Cerf e Kahn considerassem o protocolo como uma entidade única, mais tarde ele foi dividido em duas partes, TCP e IP, que operavam separadamente. Cerf e Kahn publicaram um artigo sobre o TCP/IP em maio de 1974 em *IEEE Transactions on Communication Technology* [Cerf, 1974].

O protocolo TCP/IP, que é o ‘feijão com arroz’ da Internet de hoje, foi elaborado antes dos PCs e das estações de trabalho, antes da proliferação das Ethernets e de outras tecnologias de redes locais, antes da Web, da recepção de vídeo e do bate-papo virtual. Cerf e Kahn perceberam a necessidade de um protocolo de rede que, de um lado, fornecesse amplo suporte para aplicações ainda a serem definidas e que, de outro, permitisse a interoperabilidade entre hospedeiros arbitrários e protocolos de camada de enlace.

Em 2004, Cerf e Kahn receberam o prêmio ACM Turing Award, considerado o Prêmio Nobel da Computação pelo “trabalho pioneiro sobre interligação em rede, incluindo o projeto e a implementação dos protocolos de comunicação da Internet, TCP/IP e por inspirarem liderança na área de redes.”

limitada pelo **tamanho máximo do segmento** (*maximum segment size* — MSS). O MSS normalmente é estabelecido primeiramente determinando o tamanho do maior quadro de camada de enlace que pode ser enviado pelo hospedeiro remetente local (denominado **unidade máxima de transmissão** — *maximum transmission unit* — MTU) e, em seguida, estabelecendo um MSS que garanta que um segmento TCP (quando encapsulado em um datagrama IP) caberá em um único quadro de camada de enlace. Valores comuns da MTU são 1.460 bytes, 536 bytes e 512 bytes. Também foram propostas abordagens para descobrir a MTU de Caminho (Path MTU) — o maior quadro de camada de enlace que pode ser enviado por todos os enlaces desde a fonte até o destino [RFC 1191]. Note que o MSS é a máxima quantidade de dados de camada de aplicação no segmento, e não o tamanho máximo do segmento TCP incluindo cabeçalhos. (Essa terminologia é confusa, mas temos de conviver com ela, pois já está arraigada.)

O TCP combina cada porção de dados do cliente com um cabeçalho TCP, formando, assim, **segmentos TCP**. Os segmentos são passados para baixo, para a camada de rede, onde são encapsulados separadamente dentro dos datagramas IP de camada de rede. Os datagramas IP são então enviados para dentro da rede. Quando o TCP recebe um segmento na outra extremidade, os dados do segmento são colocados no buffer de recepção da conexão, como mostra a Figura 3.28. A aplicação lê a cadeia de dados desse buffer. Cada lado da conexão tem seus próprios buffers de envio e seu próprio buffer de recepção. (Você pode ver o applet de controle de fluxo on-line em http://www.aw.com/kurose_br, que oferece uma animação dos buffers de envio e de recepção.)

Entendemos, dessa discussão, que uma conexão TCP consiste em buffers, variáveis e um socket de conexão de um processo em um hospedeiro e outro conjunto de buffers, variáveis e um socket de conexão de um processo em outro hospedeiro. Como mencionamos anteriormente, nenhum buffer nem variáveis são alocados à conexão nos elementos da rede (roteadores, comutadores e repetidores) existentes entre os hospedeiros.

3.5.2 Estrutura do segmento TCP

Agora que examinamos brevemente a conexão TCP, vamos verificar a estrutura do segmento TCP, que consiste em campos de cabeçalho e um campo de dados. O campo de dados contém uma quantidade de dados de

aplicação. Como citado antes, o MSS limita o tamanho máximo do campo de dados de um segmento. Quando o TCP envia um arquivo grande, tal como uma imagem de uma página Web, ele comumente fragmenta o segmento em pedaços de tamanho MSS (exceto o último pedaço, que muitas vezes é menor do que o MSS). Aplicações interativas, contudo, muitas vezes transmitem quantidades de dados que são menores do que o MSS. Por exemplo, com aplicações de login remoto como Telnet, o campo de dados do segmento TCP é, muitas vezes, de apenas 1 byte. Como o cabeçalho TCP tem tipicamente 20 bytes (12 bytes mais do que o cabeçalho UDP), o comprimento dos segmentos enviados por Telnet pode ser de apenas 21 bytes.

A Figura 3.29 mostra a estrutura do segmento TCP. Como acontece com o UDP, o cabeçalho inclui **números de porta de fonte e de destino**, que são usados para multiplexação e demultiplexação de dados de/para aplicações de camadas superiores e, assim como no UDP, inclui um **campo de soma de verificação**. Um cabeçalho de segmento TCP também contém os seguintes campos:

- O **campo de número de sequência** de 32 bits e o **campo de número de reconhecimento** de 32 bits são usados pelos TCPs remetente e destinatário na implementação de um serviço confiável de transferência de dados, como discutido a seguir.
- O campo de **janela de recepção** de 16 bits é usado para controle de fluxo. Veremos em breve que esse campo é usado para indicar o número de bytes que um destinatário está disposto a aceitar.
- O **campo de comprimento de cabeçalho** de 4 bits especifica o comprimento do cabeçalho TCP em palavras de 32 bits. O cabeçalho TCP pode ter comprimento variável devido ao campo de opções TCP. (O campo de opções TCP normalmente está vazio, de modo que o comprimento do cabeçalho TCP típico é 20 bytes.)
- O **campo de opções**, opcional e de comprimento variável, é usado quando um remetente e um destinatário negociam o MSS, ou como um fator de aumento de escala da janela para utilização em redes de alta velocidade. Uma opção de marca de tempo é também definida. Consulte o RFC 854 e o RFC 1323 para detalhes adicionais.
- O **campo de flag** contém 6 bits. O **bit ACK** é usado para indicar se o valor carregado no campo de reconhecimento é válido, isto é, se o segmento contém um reconhecimento para um segmento que foi recebido com sucesso. Os bits **RST**, **SYN** e **FIN** são usados para estabelecer e encerrar a conexão, como

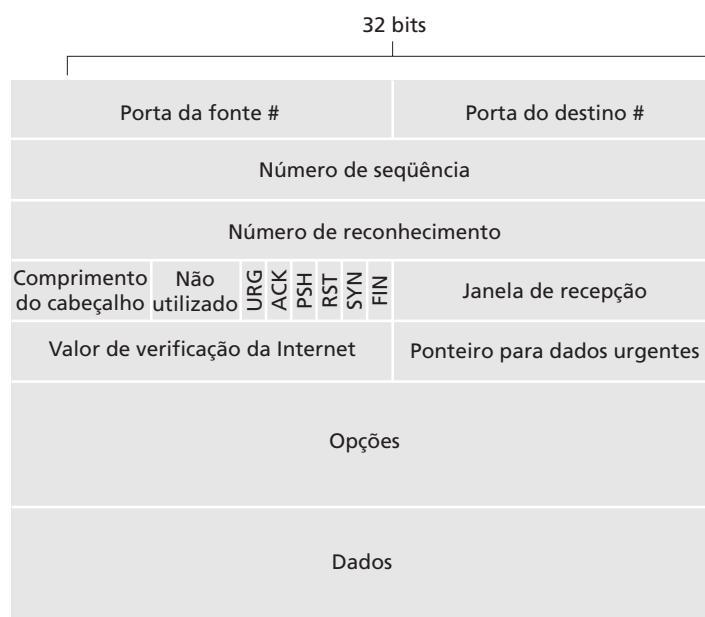


Figura 3.29 Estrutura do segmento TCP

discutiremos ao final desta seção. Marcar o bit **PSH** indica que o destinatário deve passar os dados para a camada superior imediatamente. Por fim, o bit **URG** é usado para mostrar que há dados nesse segmento que a entidade da camada superior do lado remetente marcou como ‘urgentes’. A localização do último byte desses dados urgentes é indicada pelo **campo de ponteiro de urgência** de 16 bits. O TCP deve informar à entidade da camada superior do lado destinatário quando existem dados urgentes e passar a ela um ponteiro para o final desses dados. (Na prática, o PSH, o URG e o ponteiro de dados urgentes não são usados. Contudo, mencionamos esses campos para descrever todos.)

Números de sequência e números de reconhecimento

Dois dos mais importantes campos do cabeçalho do segmento TCP são o campo de número de sequência e o campo de número de reconhecimento. Esses campos são parte fundamental do serviço de transferência confiável de dados do TCP. Mas, antes de discutirmos como esses campos são utilizados, vamos explicar exatamente o que o TCP coloca nesses campos.

O TCP vê os dados como uma cadeia de bytes não estruturada, mas ordenada. O uso que o TCP faz dos números de sequência reflete essa visão, pois esses números são aplicados sobre a cadeia de bytes transmitidos, e não sobre a série de segmentos transmitidos. O **número de sequência para um segmento** é o número do primeiro byte do segmento. Vamos ver um exemplo. Suponha que um processo no hospedeiro A queira enviar uma cadeia de dados para um processo no hospedeiro B por uma conexão TCP. O TCP do hospedeiro A vai implicitamente numerar cada byte da cadeia de dados. Suponha que a cadeia de dados consista em um arquivo composto de 500.000 bytes, que o MSS seja de 1.000 bytes e que seja atribuído o número 0 ao primeiro byte da cadeia de dados. Como mostra a Figura 3.30, o TCP constrói 500 segmentos a partir da cadeia de dados. O primeiro segmento recebe o número de sequência 0; o segundo, o número de sequência 1.000; o terceiro, o número de sequência 2.000, e assim por diante. Cada número de sequência é inserido no campo de número de sequência no cabeçalho do segmento TCP apropriado.

Vamos agora considerar os números de reconhecimento. Esses números são um pouco mais complicados do que os números de sequência. Lembre-se de que o TCP é *full-duplex*, portanto o hospedeiro A pode estar recebendo dados do hospedeiro B enquanto envia dados ao hospedeiro B (como parte da mesma conexão TCP). Cada um dos segmentos que chegam do hospedeiro B tem um número de sequência para os dados que estão fluindo de B para A. *O número de reconhecimento que o hospedeiro A atribui a seu segmento é o número de sequência do próximo byte que ele estiver aguardando do hospedeiro B.* É bom examinarmos alguns exemplos para entendermos o que está acontecendo aqui. Suponha que o hospedeiro A tenha recebido do hospedeiro B todos os bytes numerados de 0 a 535 e também que esteja prestes a enviar um segmento ao hospedeiro B. O hospedeiro A está esperando pelo byte 536 e por todos os bytes subsequentes da corrente de dados do hospedeiro B. Assim, ele coloca o número 536 no campo de número de reconhecimento do segmento que envia para o hospedeiro B.

Como outro exemplo, suponha que o hospedeiro A tenha recebido um segmento do hospedeiro B contendo os bytes de 0 a 535 e outro segmento contendo os bytes de 900 a 1.000. Por alguma razão, o hospedeiro A ainda não recebeu os bytes de 536 a 899. Nesse exemplo, ele ainda está esperando pelo byte 536 (e os superiores) para poder recriar a cadeia de dados de B. Assim, o segmento seguinte que A envia a B conterá 536 no campo de

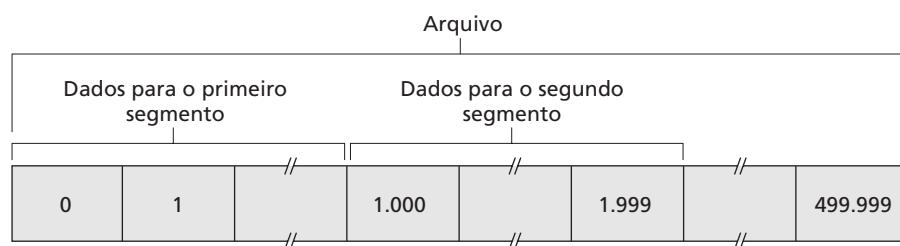


Figura 3.30 Dividindo os dados do arquivo em segmentos TCP

número de reconhecimento. Como o TCP somente reconhece bytes até o primeiro byte que estiver faltando na cadeia, dizemos que o TCP provê **reconhecimentos cumulativos**.

Esse último exemplo também revela uma questão importante, mas sutil. O hospedeiro A recebeu o terceiro segmento (bytes de 900 a 1.000) antes de receber o segundo (bytes de 536 a 899). Portanto, o terceiro segmento chegou fora de ordem. E o que um hospedeiro faz quando recebe segmentos fora de ordem em uma conexão TCP? Esta é a questão. O interessante é que os RFCs do TCP não impõem nenhuma regra para isso e deixam a decisão para quem estiver programando a implementação TCP. Há basicamente duas opções: (1) o destinatário descarta imediatamente os segmentos fora de ordem (o que, como discutimos anteriormente, pode simplificar o projeto do destinatário) ou (2) o destinatário conserva os bytes fora de ordem e espera pelos bytes faltantes para preencher as lacunas. Claro que a segunda alternativa é mais eficiente em termos de largura de banda de rede e é a abordagem adotada na prática.

Na Figura 3.30, admitimos que o número de sequência inicial era 0. Na verdade, ambos os lados de uma conexão TCP escolhem aleatoriamente um número de sequência inicial. Isso é feito para minimizar a possibilidade de um segmento de uma conexão já encerrada entre dois hospedeiros e ainda presente na rede ser tomado por um segmento válido em uma conexão posterior entre esses dois mesmos hospedeiros (que também podem estar usando os mesmos números de porta da conexão antiga) [Sunshine, 1978].

Telnet: um estudo de caso para números de sequência e números de reconhecimento

O Telnet, definido no RFC 854, é um protocolo popular de camada de aplicação utilizado para fazer login remoto. Ele roda sobre TCP e é projetado para trabalhar entre qualquer par de hospedeiros. Diferentemente das aplicações de transferência de dados em grandes blocos, que foram discutidas no Capítulo 2, o Telnet é uma aplicação interativa. Discutiremos, agora, um exemplo de Telnet, pois ele ilustra muito bem números de sequência e de reconhecimento do TCP. Observamos que muitos usuários agora preferem usar o protocolo ‘ssh’ em vez do Telnet, visto que dados enviados por uma conexão Telnet (incluindo senhas!) não são criptografados, o que torna essa aplicação vulnerável a ataques de bisbilhoteiros (como discutiremos na Seção 8.7).

Suponha que o hospedeiro A inicie uma sessão Telnet com o hospedeiro B. Como o hospedeiro A inicia a sessão, ele é rotulado de cliente, enquanto o hospedeiro B é rotulado de servidor. Cada caractere digitado pelo usuário (no cliente) será enviado ao hospedeiro remoto; este devolverá uma cópia (‘eco’) de cada caractere, que será apresentada na tela Telnet do usuário. Esse eco é usado para garantir que os caracteres vistos pelo usuário do Telnet já foram recebidos e processados no local remoto. Assim, cada caractere atravessa a rede duas vezes entre o momento em que o usuário aperta o teclado e o momento em que o caractere é apresentado em seu monitor.

Suponha agora que o usuário digite a letra ‘C’ e saia para tomar um café. Vamos examinar os segmentos TCP que são enviados entre o cliente e o servidor. Como mostra a Figura 3.31, admitamos que os números de sequência iniciais sejam 42 e 79 para cliente e servidor, respectivamente. Lembre-se de que o número de sequência de um segmento será o número de sequência do primeiro byte do seu campo de dados. Assim, o primeiro segmento enviado do cliente terá número de sequência 42; o primeiro segmento enviado do servidor terá número de sequência 79. Note que o número de reconhecimento será o número de sequência do próximo byte de dados que o hospedeiro estará aguardando. Após o estabelecimento da conexão TCP, mas antes de quaisquer dados serem enviados, o cliente ficará esperando pelo byte 79 e o servidor, pelo byte 42.

Como mostra a Figura 3.31, são enviados três segmentos. O primeiro é enviado do cliente ao servidor, contendo, em seu campo de dados, um byte com a representação ASCII para a letra ‘C’. O primeiro segmento também tem 42 em seu campo de número de sequência, como acabamos de descrever. E mais, como o cliente ainda não recebeu nenhum dado do servidor, esse segmento terá o número 79 em seu campo de número de reconhecimento.

O segundo segmento é enviado do servidor ao cliente. Esse segmento tem dupla finalidade. A primeira finalidade é fornecer um reconhecimento para os dados que o servidor recebeu. Ao colocar 43 no campo de reconhecimento, o servidor está dizendo ao cliente que recebeu com sucesso tudo até o byte 42 e agora está aguardando os bytes de 43 em diante. A segunda finalidade desse segmento é ecoar a letra ‘C’. Assim, o segundo segmento tem a representação ASCII de ‘C’ em seu campo de dados. Ele tem o número de sequência 79, que é o

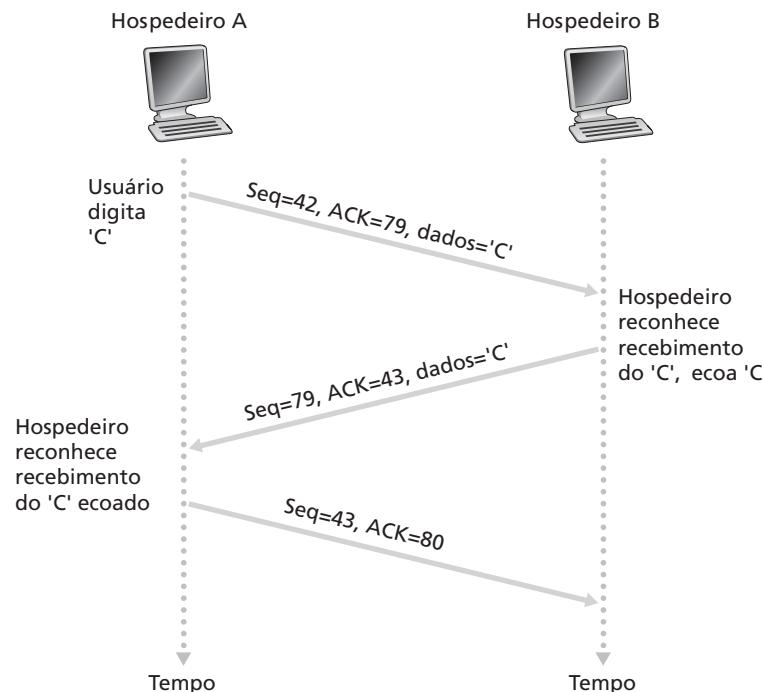


Figura 3.31 Números de sequência e de reconhecimento para uma aplicação Telnet simples sobre TCP

número de sequência inicial do fluxo de dados de servidor para cliente dessa conexão TCP, pois este é o primeiríssimo byte de dados que o servidor está enviando. Note que o reconhecimento para dados do cliente para o servidor é levado em um segmento que carrega dados do servidor para o cliente. Dizemos que este reconhecimento **pegou uma carona** no segmento de dados do servidor para o cliente. Tecnicamente, essa ‘carona’ recebe o nome de *piggyback*.

O terceiro segmento é enviado do cliente ao servidor. Seu único propósito é reconhecer os dados que recebeu do servidor. (Lembre-se de que o segundo segmento continha dados — a letra ‘C’ — do servidor para o cliente.) Esse segmento tem um campo de dados vazio (isto é, o reconhecimento não está pegando carona com nenhum dado do cliente para o servidor), tem o número 80 no campo do número de reconhecimento porque o cliente recebeu a cadeia de dados até o byte com número de sequência 79 e agora está aguardando os bytes de 80 em diante. É possível que você esteja pensando que é estranho que esse segmento também tenha um número de sequência, já que não contém dados. Mas, como o TCP tem um campo de número de sequência, o segmento precisa apresentar algum número para preenchê-lo.

3.5.3 Estimativa do tempo de viagem de ida e volta e de esgotamento de temporização

O TCP, assim como o nosso protocolo rdt da Seção 3.4, utiliza um mecanismo de controle de temporização/retransmissão para recuperar segmentos perdidos. Embora conceitualmente simples, surgem muitas questões sutis quando implementamos um mecanismo de controle de temporização/retransmissão em um protocolo real como o TCP. Talvez a pergunta mais óbvia seja a duração dos intervalos de controle. Evidentemente, esse intervalo deve ser maior do que o tempo de viagem de ida e volta da conexão (RTT), isto é, o tempo decorrido entre o envio de um segmento e seu reconhecimento. Se não fosse assim, seriam enviadas retransmissões desnecessárias. Mas quão maior deve ser o intervalo e, antes de mais nada, como o RTT deve ser estimado? Deve-se associar um temporizador a cada segmento não reconhecido? São tantas perguntas! Nesta seção, nossa discussão se baseia no trabalho de [Jacobson, 1988] sobre TCP e nas recomendações da IETF vigentes para o gerenciamento de temporizadores TCP [RFC 2988].

Estimativa do tempo de viagem de ida e volta

Vamos iniciar nosso estudo do gerenciamento do temporizador TCP considerando como esse protocolo estima o tempo de viagem de ida e volta entre remetente e destinatário, o que apresentaremos a seguir. O RTT para um segmento, denominado SampleRTT no exemplo, é a quantidade de tempo transcorrido entre o momento em que o segmento é enviado (isto é, passado ao IP) e o momento em que é recebido um reconhecimento para o segmento. Em vez de medir um SampleRTT para cada segmento transmitido, a maioria das implementações de TCP executa apenas uma medição de SampleRTT por vez. Isto é, em qualquer instante, o SampleRTT estará sendo estimado para apenas um dos segmentos transmitidos mas ainda não reconhecidos, o que resulta em um novo valor de SampleRTT para aproximadamente cada RTT. E mais, o TCP nunca computa um SampleRTT para um segmento que foi retransmitido; apenas mede-o para segmentos que foram transmitidos uma vez. (Um dos problemas ao final do capítulo perguntará por quê.)

Obviamente, os valores de SampleRTT sofrerão variação de segmento para segmento devido a congestionamento nos roteadores e a variações de carga nos sistemas finais. Por causa dessa variação, qualquer dado valor de SampleRTT pode ser atípico. Portanto, para estimar um RTT típico, é natural tomar alguma espécie de média dos valores de SampleRTT. O TCP mantém uma média, denominada EstimatedRTT, dos valores de SampleRTT. Ao obter um novo SampleRTT, o TCP atualiza EstimatedRTT de acordo com a seguinte fórmula:

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}.$$

Essa fórmula está escrita sob a forma de um comando de linguagem de programação — o novo valor de EstimatedRTT é uma combinação ponderada entre o valor anterior de EstimatedRTT e o novo valor para SampleRTT. O valor recomendado de α é $\alpha = 0,125$ (isto é, $1/8$) [RFC 2988], caso em que essa fórmula se torna:

$$\text{EstimatedRTT} = 0,875 \cdot \text{EstimatedRTT} + 0,125 \cdot \text{SampleRTT}.$$

Note que EstimatedRTT é uma média ponderada dos valores de SampleRTT. Como veremos em um exercício ao final deste capítulo, essa média ponderada atribui um peso maior às amostras recentes do que às amostras antigas. Isso é natural, pois as amostras mais recentes refletem melhor o estado atual de congestionamento da rede. Em estatística, esse tipo de média é denominada **média móvel exponencial ponderada**. A palavra ‘exponencial’ aparece na MMEP porque o peso atribuído a um dado SampleRTT diminui exponencialmente à medida que as atualizações são realizadas. Os exercícios pedirão que você derive o termo exponencial em EstimatedRTT.

A Figura 3.32 mostra os valores de SampleRTT e EstimatedRTT para um valor de $\alpha = 1/8$, para uma conexão TCP entre gaia.cs.umass.edu (em Amherst, Massachusetts) e fantasia.eurecom.fr (no sul da França). Fica claro que as variações em SampleRTT são atenuadas no cálculo de EstimatedRTT.

Além de ter uma estimativa do RTT, também é valioso ter uma medida de sua variabilidade. O [RFC 2988] define a variação do RTT, DevRTT, como uma estimativa do desvio típico entre SampleRTT e EstimatedRTT:

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

Note que DevRTT é uma MMEP da diferença entre SampleRTT e EstimatedRTT. Se os valores de SampleRTT apresentarem pouca variação, então DevRTT será pequeno; por outro lado, se houver muita variação, DevRTT será grande. O valor recomendado para β é 0,25.

Estabelecimento e gerenciamento da temporização de retransmissão

Dados valores de EstimatedRTT e DevRTT, que valor deve ser utilizado para a temporização de retransmissão do TCP? É óbvio que o intervalo deve ser maior ou igual a EstimatedRTT, caso contrário seriam enviadas retransmissões desnecessárias. Mas a temporização de retransmissão não deve ser muito maior do que EstimatedRTT, senão, quando um segmento fosse perdido, o TCP não o retransmitiria rapidamente, o que resultaria em grandes atrasos de transferência de dados. Portanto, é desejável que o valor estabelecido para a temporização seja igual a EstimatedRTT mais uma certa margem, que deverá ser grande quando houver muita

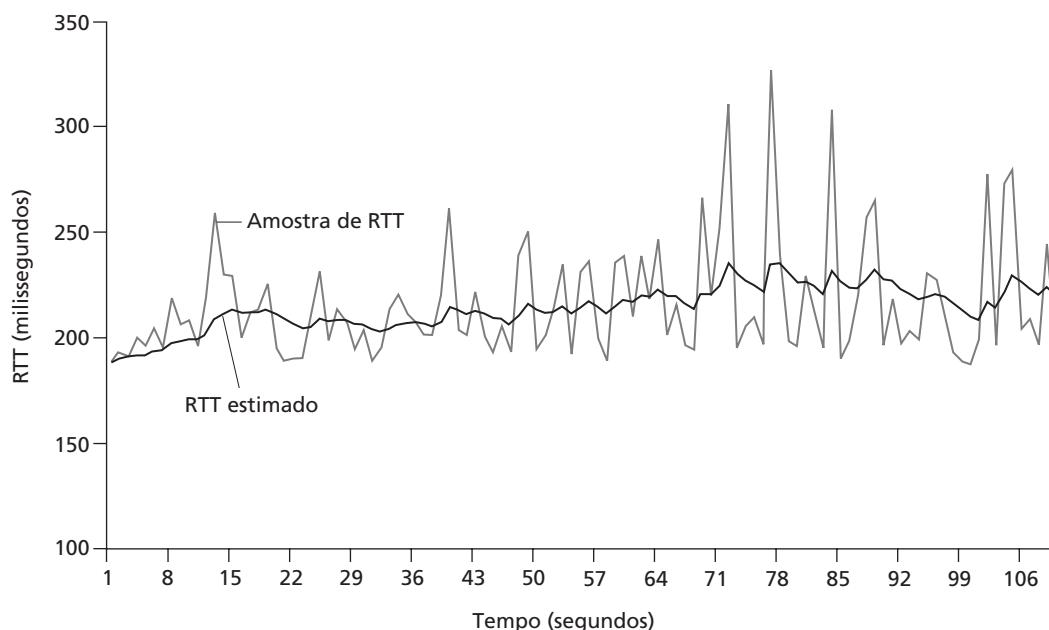


Figura 3.32 Amostras de RTTs e RTTs estimados

variação nos valores de SampleRTT e pequena quando houver pouca variação. Assim, o valor de DevRTT deve entrar em jogo. Todas essas considerações são levadas em conta no método do TCP para determinar a temporização de retransmissão:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$



O TCP fornece transferência confiável de dados usando reconhecimentos positivos e temporizadores, de modo muito parecido com o que estudamos na Seção 3.4. O protocolo reconhece dados que foram recebidos corretamente e retransmite segmentos quando entende que eles ou seus reconhecimentos correspondentes foram perdidos ou corrompidos. Certas versões do TCP também têm um mecanismo NAK implícito — com o mecanismo de retransmissão rápida do TCP. O recebimento de três ACKs duplicados para um dado segmento serve como um NAK implícito para o segmento seguinte, acionando a retransmissão daquele segmento antes que o tempo se esgote. O TCP usa números de sequência para permitir que o destinatário identifique segmentos perdidos ou duplicados. Exatamente como no caso de nosso protocolo de transferência confiável de dados rdt3.0, o TCP em si não pode determinar com certeza se um segmento, ou seu ACK, está perdido, corrompido ou excessivamente atrasado. No remetente, a resposta do TCP será a mesma: retransmitir o segmento em questão. O TCP também utiliza paralelismo, permitindo que o remetente tenha, a qualquer tempo, múltiplos segmentos transmitidos mas ainda não reconhecidos. Vimos anteriormente que o paralelismo pode melhorar muito a vazão de uma sessão quando a razão entre o tempo de transmissão do segmento e o atraso de viagem de ida e volta é pequena. O número específico de segmentos não reconhecidos que um remetente pode ter é determinado pelos mecanismos de controle de fluxo e controle de congestionamento do TCP. O controle de fluxo do TCP é discutido no final desta seção; o controle de congestionamento do TCP é discutido na Seção 3.7. Por enquanto, devemos apenas ficar cientes de que o TCP remetente usa paralelismo.

3.5.4 Transferência confiável de dados

Lembre-se de que o serviço da camada de rede da Internet (serviço IP) não é confiável. O IP não garante a entrega de datagramas na ordem correta nem a integridade de seus dados nos datagramas. Com o serviço IP, os datagramas podem transbordar dos buffers dos roteadores e jamais alcançar seu destino; podem também chegar fora de ordem. Além disso, os bits dos datagramas podem ser corrompidos (passar de 0 para 1 e vice-versa). Como os segmentos da camada de transporte são carregados pela rede por datagramas IPs, também eles podem sofrer esses mesmos problemas.

O TCP cria um **serviço de transferência confiável de dados** sobre o serviço de melhor esforço do IP. Esse serviço de transferência garante que a cadeia de dados que um processo lê a partir de seu buffer de recebimento TCP não está corrompida, não tem lacunas, não tem duplicações e está em sequência, isto é, a cadeia de bytes é exatamente a mesma cadeia de bytes enviada pelo sistema final que está do outro lado da conexão. O modo como o TCP provê transferência confiável de dados envolve muitos dos princípios estudados na Seção 3.4.

Quando desenvolvemos anteriormente técnicas de transferência confiável de dados, era conceitualmente mais fácil admitir que existia um temporizador individual associado com cada segmento transmitido mas ainda não reconhecido. Embora, em teoria, isso seja ótimo, o gerenciamento de temporizadores pode exigir considerável sobrecarga. Assim, os procedimentos recomendados no [RFC 2988] para gerenciamento de temporizadores TCP utilizam apenas um único temporizador de retransmissão, mesmo que haja vários segmentos transmitidos ainda não reconhecidos. O protocolo TCP apresentado nesta seção segue essa recomendação.

Discutiremos como o TCP provê transferência confiável de dados em duas etapas incrementais. Em primeiro lugar, apresentamos uma descrição muito simplificada de um remetente TCP que utiliza somente controle de temporizadores para se recuperar da perda de segmentos; em seguida, apresentaremos uma descrição mais complexa que utiliza reconhecimentos duplicados além de temporizadores de retransmissão. Na discussão que se segue, admitimos que os dados estão sendo enviados em uma direção somente, do hospedeiro A ao hospedeiro B, e que o hospedeiro A está enviando um arquivo grande.

A Figura 3.33 apresenta uma descrição muito simplificada de um remetente TCP.

Vemos que há três eventos importantes relacionados com a transmissão e a retransmissão de dados no TCP remetente: dados recebidos da aplicação acima; esgotamento do temporizador e recebimento de ACK. Quando ocorre o primeiro evento importante, o TCP recebe dados da camada de aplicação, encapsula-os em um segmento e passa-o ao IP. Note que cada segmento inclui um número de sequência que é o número da corrente de bytes do primeiro byte de dados no segmento, como descrito na Seção 3.5.2. Note também que, se o temporizador não estiver funcionando naquele instante para algum outro segmento, o TCP aciona o temporizador quando o segmento é passado para o IP. (Fica mais fácil se você imaginar que o temporizador está associado com o mais antigo segmento não reconhecido.) O intervalo de expiração para esse temporizador é o `TimeoutInterval`, calculado a partir de `EstimatedRTT` e `DevRTT`, como descrito na Seção 3.5.3.

O segundo evento importante é o esgotamento do temporizador. O TCP responde a esse evento retransmitindo o segmento que causou o esgotamento da temporização e então reinicia o temporizador.

O terceiro evento importante que deve ser manipulado pelo TCP remetente é a chegada de um segmento de reconhecimento (ACK) do destinatário (mais especificamente, um segmento contendo um valor de campo de ACK válido). Quando da ocorrência desse evento, o TCP compara o valor do ACK, y , com sua variável `SendBase`. A variável de estado `SendBase` do TCP é o número de sequência do mais antigo byte não reconhecido. (Assim, $SendBase - 1$ é o número de sequência do último byte que se sabe ter sido recebido pelo destinatário corretamente e na ordem certa.) Como comentamos anteriormente, o TCP usa reconhecimentos cumulativos, de modo que y reconhece o recebimento de todos os bytes antes do byte número y . Se $y > SendBase$, então o ACK está reconhecendo um ou mais bytes não reconhecidos anteriormente. Desse modo, o remetente atualiza sua variável `SendBase` e também reinicia o temporizador se houver quaisquer segmentos ainda não reconhecidos.

Alguns cenários interessantes

Acabamos de descrever uma versão muito simplificada do modo como o TCP provê transferência confiável de dados, mas mesmo essa descrição tão simplificada tem muitas sutilezas. Para ter uma boa percepção de como

/* Suponha que o remetente não seja compelido pelo fluxo de TCP ou controle de congestionamento, que o tamanho dos dados vindos de cima seja menor do que o MSS e que a transferência de dados ocorra apenas em uma direção.*/

```

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch (event)

        event: data received from the application above
            create TCP segment with sequence number NextSeqNum
            if (timer currently not running)
                start timer
            pass segment to IP
            NextSeqNum=NextSeqNum+length (data)
            break;

        event: timer timeout
            retransmit not-yet-acknowledged segment with
                smallest sequence number
            start timer
            break;

        event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase=y
                If (there are currently any not-yet-acknowledged segments)
                    start timer
            }
}

```

Figura 3.33 Remetente TCP simplificado

esse protocolo funciona, vamos agora examinar alguns cenários simples. A Figura 3.34 ilustra o primeiro cenário em que um hospedeiro A envia um segmento ao hospedeiro B. Suponha que esse segmento tenha número de sequência 92 e contenha 8 bytes de dados. Após enviá-lo, o hospedeiro A espera por um segmento de B com número de reconhecimento 100. Embora o segmento de A seja recebido em B, o reconhecimento de B para A se perde. Nesse caso, ocorre o evento de expiração do temporizador e o hospedeiro A retransmite o mesmo segmento. É claro que, quando recebe a retransmissão, o hospedeiro B observa, pelo número de sequência, que o segmento contém dados que já foram recebidos. Assim, o TCP no hospedeiro B descarta os bytes do segmento retransmitido.

Em um segundo cenário, mostrado na Figura 3.35, o hospedeiro A envia dois segmentos seguidos. O primeiro segmento tem número de sequência 92 e 8 bytes de dados. O segundo segmento tem número de sequência 100 e 20 bytes de dados. Suponha que ambos cheguem intactos em B e que B envie dois reconhecimentos separados para cada um desses segmentos. O primeiro desses reconhecimentos tem número de reconhecimento 100; o segundo, número de reconhecimento 120. Suponha agora que nenhum dos reconhecimentos chegue ao hospedeiro A antes do esgotamento do temporizador. Quando ocorre o evento de expiração do temporizador,

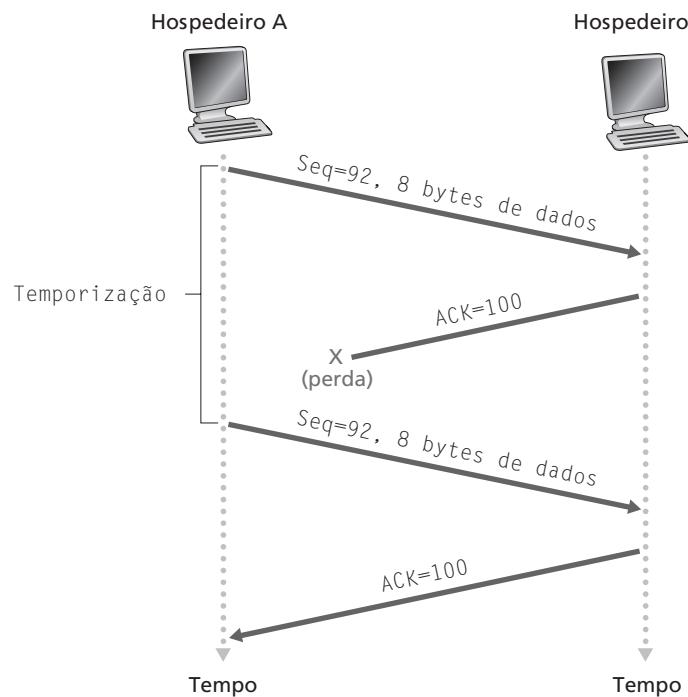


Figura 3.34 Retransmissão devido a um reconhecimento perdido

o hospedeiro A reenvia o primeiro segmento com número de sequência 92 e reinicia o temporizador. Contanto que o ACK do segundo segmento chegue antes que o temporizador expire novamente, o segundo segmento não será retransmitido.

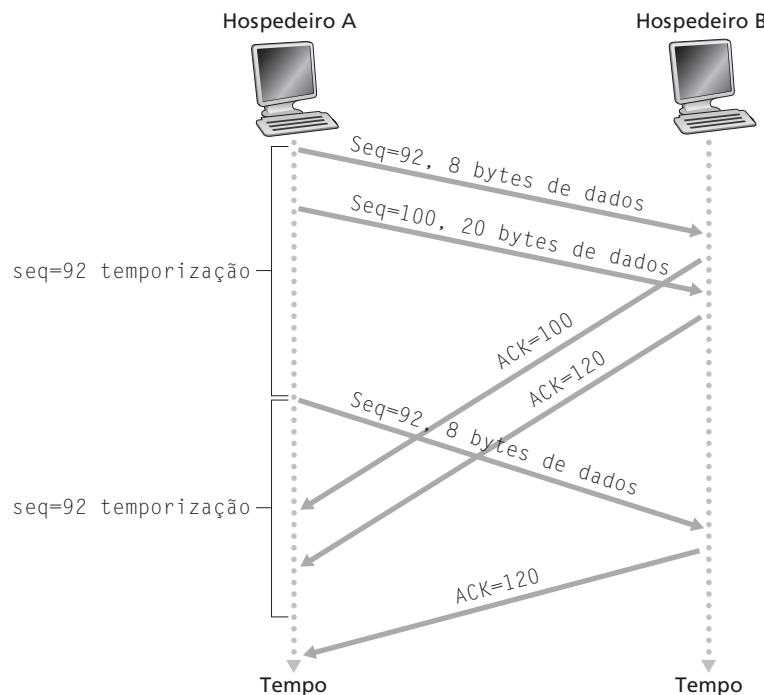


Figura 3.35 Segmento 100 não retransmitido

Em um terceiro e último cenário, suponha que o hospedeiro A envie dois segmentos, exatamente como no segundo exemplo. O reconhecimento do primeiro segmento é perdido na rede, mas, um pouco antes do evento de expiração, o hospedeiro A recebe um reconhecimento com número 120. O hospedeiro A, portanto, sabe que B recebeu *tudo* até o byte 119; portanto, ele não reenvia nenhum dos dois segmentos. Esse cenário está ilustrado na Figura 3.36.

Duplicação do tempo de expiração

Discutiremos agora algumas modificações empregadas por grande parte das implementações do TCP. A primeira refere-se à duração do tempo de expiração após a expiração de um temporizador. Nessa modificação, sempre que ocorre o evento de expiração do temporizador, o TCP retransmite o segmento ainda não reconhecido que tenha o menor número de sequência, como descrevemos anteriormente. Mas, a cada retransmissão, o TCP ajusta o próximo tempo de expiração para o dobro do valor anterior em vez de derivá-lo dos últimos EstimatedRTT e DevRTT (como descrito na Seção 3.5.3). Por exemplo, suponha que o TimeoutInterval associado com o mais antigo segmento ainda não reconhecido seja 0,75 segundo quando o temporizador expirar pela primeira vez. O TCP então retransmite esse segmento e ajusta o novo tempo de expiração para 1,5 segundo. Se o temporizador expirar novamente 1,5 segundo mais tarde, o TCP retransmitirá novamente esse segmento, agora ajustando o tempo de expiração para 3,0 segundos. Assim, o tempo aumenta exponencialmente após cada retransmissão. Todavia, sempre que o temporizador é iniciado após qualquer um dos outros dois eventos (isto é, dados recebidos da aplicação acima e ACK recebido), o TimeoutInterval será derivado dos valores mais recentes de EstimatedRTT e DevRTT.

Essa modificação provê uma forma limitada de controle de congestionamento. (Maneiras mais abrangentes de controle de congestionamento no TCP serão estudadas na Seção 3.7). A causa mais provável da expiração do temporizador é o congestionamento na rede, isto é, um número muito grande de pacotes chegando a uma (ou mais) fila de roteadores no caminho entre a fonte e o destino, o que provoca descarte de pacotes e/ou longos atrasos de fila. Se as fontes continuarem a retransmitir pacotes persistentemente durante um congestionamento, ele pode piorar. Para que isso não aconteça, o TCP age mais educadamente: cada remetente retransmite após intervalos cada vez mais longos. Veremos que uma ideia semelhante a essa é utilizada pela Ethernet, quando estudarmos CSMA/CD no Capítulo 5.

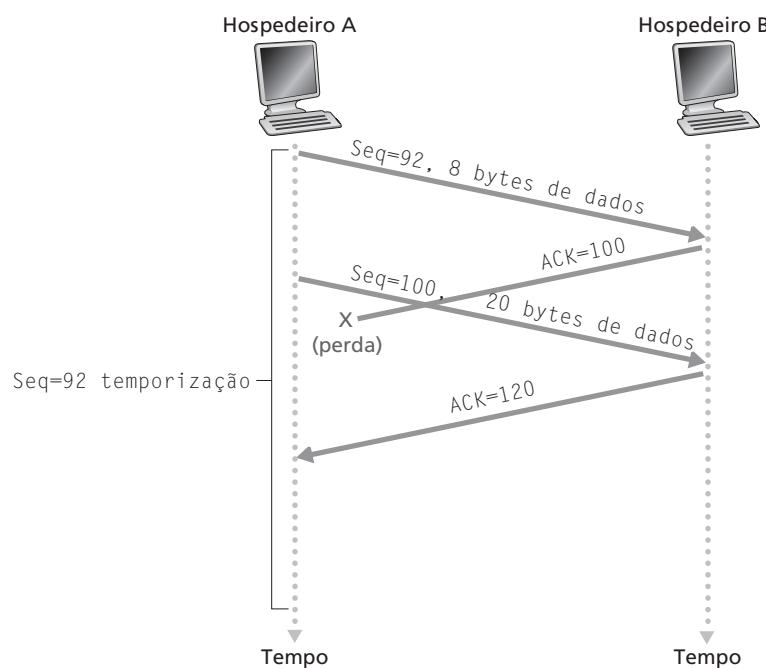


Figura 3.36 Um reconhecimento cumulativo evita retransmissão do primeiro segmento

Retransmissão rápida

Um dos problemas de retransmissões acionadas por expiração de temporizador é que o período de expiração pode ser relativamente longo. Quando um segmento é perdido, esse longo período força o remetente a atrasar o reenvio do pacote perdido, consequentemente aumentando o atraso fim a fim. Felizmente, o remetente pode com frequência detectar perda de pacote bem antes de ocorrer o evento de expiração, observando os denominados ACKs duplicados. Um **ACK duplicado** é um ACK que reconhece novamente um segmento para o qual o remetente já recebeu um reconhecimento anterior. Para entender a resposta do remetente a um ACK duplicado, devemos examinar por que o destinatário envia um ACK duplicado em primeiro lugar. A Tabela 3.2 resume a política de geração de ACKs do TCP destinatário [RFC 1122, RFC 2581].

Quando um TCP destinatário recebe um segmento com um número de sequência que é maior do que o número de sequência subsequente, esperado e na ordem, ele detecta uma lacuna na corrente de dados — isto é, a falta de um segmento. Essa lacuna poderia ser o resultado de segmentos perdidos ou reordenados dentro da rede. Uma vez que o TCP não usa reconhecimentos negativos, o destinatário não pode enviar um reconhecimento negativo explícito de volta ao remetente. Em vez disso, ele simplesmente reconhece novamente (isto é, gera um ACK duplicado) o último byte de dados que recebeu na ordem. (Note que a Tabela 3.2 admite o caso em que o destinatário não descarta segmentos fora de ordem.)

Como um remetente quase sempre envia um grande número de segmentos, um atrás do outro, se um segmento for perdido, provavelmente existirão muitos ACKs duplicados, também um após o outro. Se o TCP remetente receber três ACKs duplicados para os mesmos dados, ele tomará isso como indicação de que o segmento que se seguiu ao segmento reconhecido três vezes foi perdido. (Nos exercícios de fixação consideraremos por que o remetente espera três ACKs duplicados e não apenas um.) No caso de receber três ACKs duplicados, o TCP remetente realiza uma **retransmissão rápida** [RFC 2581], retransmitindo o segmento que falta *antes* da expiração do temporizador do segmento. Isso é mostrado na Figura 3.37, em que o segundo segmento é perdido, e então retransmitido antes da expiração do temporizador. Para o TCP com retransmissão rápida, o seguinte trecho de codificação substitui o evento ACK recebido na Figura 3.33:

```
event: ACK received, with ACK field value of y
    if (y > SendBase) {
        SendBase=y
        if (there are currently any not yet
            acknowledged segments)
            start timer
    }
    else { /* a duplicate ACK for already ACKed
        segment */
        increment number of duplicate ACKs
        received for y
        if (number of duplicate ACKS received
            for y==3) {
            /* TCP fast retransmit */
            resend segment with sequence number y
        }
        break;
    }
```

Observamos anteriormente que muitas questões sutis vêm à tona quando um mecanismo de controle de temporização/retransmissão é implementado em um protocolo real como o TCP. Os procedimentos acima, cuja evolução é resultado de mais de 15 anos de experiência com temporizadores TCP, devem convencê-lo de que, indubitavelmente, é isso que acontece!

Evento	Ação do TCP Destinatário
Chegada de segmento na ordem com número de sequência esperado. Todos os dados até o número de sequência esperado já reconhecidos.	ACK retardado. Espera de até 500 milissegundos pela chegada de um outro segmento na ordem. Se o segmento seguinte na ordem não chegar nesse intervalo, envia um ACK.
Chegada de segmento na ordem com número de sequência esperado. Um outro segmento na ordem esperando por transmissão de ACK.	Envio imediato de um único ACK cumulativo, reconhecendo ambos os segmentos na ordem.
Chegada de um segmento fora da ordem com número de sequência mais alto do que o esperado. Lacuna detectada.	Envio imediato de um ACK duplicado, indicando número de sequência do byte seguinte esperado (que é a extremidade mais baixa da lacuna).
Chegada de um segmento que preenche, parcial ou completamente, a lacuna nos dados recebidos.	Envio imediato de um ACK, contanto que o segmento comece na extremidade mais baixa da lacuna.

Tabela 3.2 Recomendação para geração de ACKs TCP [RFC 1122, RFC 2581]

Go-Back-N ou repetição seletiva?

Vamos encerrar nosso estudo do mecanismo de recuperação de erros do TCP considerando a seguinte pergunta: o TCP é um protocolo GBN ou SR? Lembre-se de que, no TCP, os reconhecimentos são cumulativos e segmentos corretamente recebidos, mas fora da ordem, não são reconhecidos (ACK) individualmente pelo destinatário. Consequentemente, como mostra a Figura 3.33 (veja também a Figura 3.19), o TCP remetente precisa tão somente lembrar o menor número de sequência de um byte transmitido mas não reconhecido (SendBase) e o número de sequência do byte seguinte a ser enviado (NextSeqNum). Nesse sentido, o TCP se parece muito com um protocolo ao estilo do GBN. Porém, há algumas diferenças surpreendentes entre o TCP e o GBN. Muitas implementações do TCP armazenarão segmentos recebidos corretamente, mas fora da ordem [Stevens, 1994].

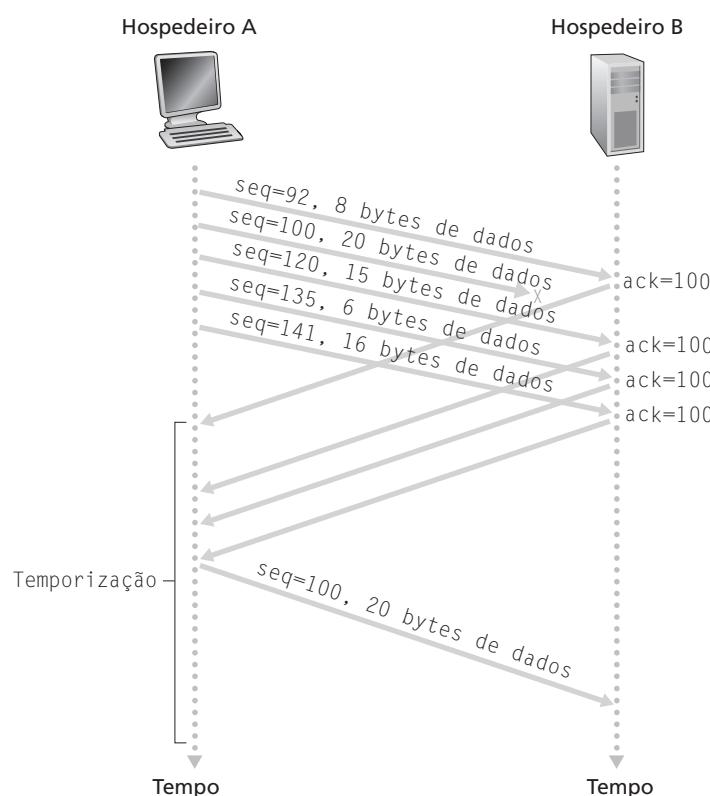


Figura 3.37 Retransmissão rápida: retransmitir o segmento que falta antes da expiração do temporizador do segmento

Considere também o que acontece quando o remetente envia uma sequência de segmentos $1, 2, \dots, N$ e todos os segmentos chegam ao destinatário na ordem e sem erro. Além disso, suponha que o reconhecimento para o pacote $n < N$ se perca, mas que os $N - 1$ reconhecimentos restantes cheguem ao remetente antes do esgotamento de suas respectivas temporizações. Nesse exemplo, o GBN retransmitiria não somente o pacote n , mas também todos os pacotes subsequentes $n + 1, n + 2, \dots, N$. O TCP, por outro lado, retransmitiria no máximo um segmento, a saber, o segmento n . E mais, o TCP nem ao menos retransmitiria o segmento n se o reconhecimento para o segmento $n + 1$ chegasse antes do final da temporização para o segmento n .

Uma modificação proposta para o TCP, denominada **reconhecimento seletivo** [RFC 2018], permite que um destinatário TCP reconheça seletivamente segmentos fora de ordem, em vez de apenas reconhecer cumulativamente o último segmento recebido corretamente e na ordem. Quando combinado com retransmissão seletiva — isto é, saltar a retransmissão de segmentos que já foram reconhecidos seletivamente pelo destinatário —, o TCP se parece muito com nosso protocolo SR genérico. Assim, o mecanismo de recuperação de erros do TCP provavelmente é mais bem caracterizado como um híbrido dos protocolos GBN e SR.

3.5.5 Controle de fluxo

Lembre-se de que os hospedeiros de cada lado de uma conexão TCP reservam um buffer de recepção para a conexão. Quando a conexão TCP recebe bytes que estão corretos e em sequência, ele coloca os dados no buffer de recepção. O processo de aplicação associado lerá os dados a partir desse buffer, mas não necessariamente no momento em que são recebidos. Na verdade, a aplicação receptora pode estar ocupada com alguma outra tarefa e nem ao menos tentar ler os dados até muito depois da chegada deles. Se a aplicação for relativamente lenta na leitura dos dados, o remetente pode muito facilmente saturar o buffer de recepção da conexão por enviar demasiados dados muito rapidamente.

O TCP provê um **serviço de controle de fluxo** às suas aplicações, para eliminar a possibilidade de o remetente saturar o buffer do destinatário. Assim, controle de fluxo é um serviço de compatibilização de velocidades — compatibiliza a taxa à qual o remetente está enviando com a aquela à qual a aplicação receptora está lendo. Como notamos anteriormente, um TCP remetente também pode ser estrangulado devido ao congestionamento dentro da rede IP. Esse modo de controle do remetente é denominado **controle de congestionamento**, um tópico que será examinado detalhadamente nas seções 3.6 e 3.7. Mesmo que as ações executadas pelo controle de fluxo e pelo controle de congestionamento sejam similares (a regulagem do remetente), fica evidente que elas são executadas por razões muito diferentes. Infelizmente, muitos autores usam os termos de modo intercambiável, e o leitor esperto tem de tomar muito cuidado para distinguir os dois casos. Vamos agora discutir como o TCP provê seu serviço de controle de fluxo. Para podermos enxergar o quadro geral, sem nos fixarmos nos detalhes, nesta seção admitiremos que essa implementação do TCP descarta segmentos fora da ordem.

O TCP provê serviço de controle de fluxo fazendo com que o *remetente* mantenha uma variável denominada **janela de recepção**. Informalmente, a janela de recepção é usada para dar ao remetente uma ideia do espaço de buffer livre disponível no destinatário. Como o TCP é *full-duplex*, o remetente de cada lado da conexão mantém uma janela de recepção distinta. Vamos examinar a janela de recepção no contexto de uma transferência de arquivo. Suponha que o hospedeiro A esteja enviando um arquivo grande ao hospedeiro B por uma conexão TCP. O hospedeiro B aloca um buffer de recepção a essa conexão; denominemos seu tamanho *RcvBuffer*. De tempos em tempos, o processo de aplicação no hospedeiro B faz a leitura do buffer. São definidas as seguintes variáveis:

LastByteRead = o número do último byte na cadeia de dados lido do buffer pelo processo de aplicação em B.

LastByteRcvd = o número do último byte na cadeia de dados que chegou da rede e foi colocado no buffer de recepção de B.

Como o TCP não tem permissão para saturar o buffer alocado, devemos ter:

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer}$$

A janela de recepção, denominada *RcvWindow*, é ajustada para a quantidade de espaço disponível no buffer:

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

Como o espaço disponível muda com o tempo, Rwnd é dinâmica. A variável Rwnd está ilustrada na Figura 3.38.

Como a conexão usa a variável Rwnd para prover o serviço de controle de fluxo? O hospedeiro A diz ao hospedeiro B quanto espaço disponível ele tem no buffer da conexão colocando o valor corrente de Rwnd no campo de janela de recepção de cada segmento que envia a B. Inicialmente, o hospedeiro B estabelece $rwnd = RcvBuffer$. Note que, para conseguir isso, o hospedeiro B deve monitorar diversas variáveis específicas da conexão.

O hospedeiro A, por sua vez, monitora duas variáveis, LastByteSent e LastByteAcked, cujos significados são óbvios. Note que a diferença entre essas duas variáveis, $LastByteSent - LastByteAcked$, é a quantidade de dados não reconhecidos que A enviou para a conexão. Mantendo a quantidade de dados não reconhecidos menor do que o valor de Rwnd, o hospedeiro A tem certeza de que não está fazendo transbordar o buffer de recepção no hospedeiro B. Assim, o hospedeiro A tem de certificar-se, durante toda a duração da conexão, de que:

$$LastByteSent - LastByteAcked \leq rwnd$$

Há um pequeno problema técnico com esse esquema. Para percebê-lo, suponha que o buffer de recepção do hospedeiro B fique tão cheio que $Rwnd = 0$. Após anunciar ao hospedeiro A que $Rwnd = 0$, imagine que B não tenha *nada* para enviar ao hospedeiro A. Agora considere o que acontece. Enquanto o processo de aplicação em B esvazia o buffer, o TCP não envia novos segmentos com novos valores Rwnd para o hospedeiro A. Na verdade, o TCP enviará um segmento ao hospedeiro A somente se tiver dados ou um reconhecimento para enviar. Por conseguinte, o hospedeiro A nunca será informado de que foi aberto algum espaço no buffer de recepção do hospedeiro B: ele ficará bloqueado e não poderá transmitir mais dados! Para resolver esse problema, a especificação do TCP requer que o hospedeiro A continue a enviar segmentos com um byte de dados quando a janela de recepção de B for zero. Esses segmentos serão reconhecidos pelo receptor. Finalmente o buffer começará a envaziar, e os reconhecimentos conterão um valor diferente de zero em $rwnd$.

O Companion Website do livro (http://www.aw.com/kurose_br) fornece um applet interativo em Java que ilustra a operação da janela de recepção do TCP.

Agora que descrevemos o serviço de controle de fluxo do TCP, mencionaremos brevemente que o UDP não provê controle de fluxo. Para entender a questão, considere o envio de uma série de segmentos UDP de um processo no hospedeiro A para um processo no hospedeiro B. Para uma implementação UDP típica, o UDP anexará os segmentos a um buffer de tamanho finito que ‘precede’ o socket correspondente (isto é, o socket para o processo). O processo lê um segmento inteiro do buffer por vez. Se o processo não ler os segmentos com rapidez suficiente, o buffer transbordará e os segmentos serão descartados.

3.5.6 Gerenciamento da conexão TCP

Nesta subseção, examinamos mais de perto como uma conexão TCP é estabelecida e encerrada. Embora esse tópico talvez não pareça particularmente interessante, é importante, porque o estabelecimento da conexão TCP tem um peso significativo nos atrasos percebidos (por exemplo, ao navegar pela Web). Além disso, muitos dos ataques mais comuns a redes — entre eles o incrivelmente popular ataque de inundação SYN — exploram

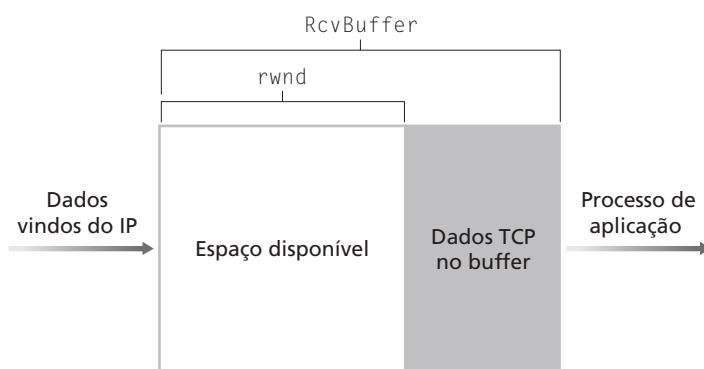


Figura 3.38 A janela de recepção (Rwnd) e o buffer de recepção (RcvBuffer)

vulnerabilidades no gerenciamento da conexão TCP. Em primeiro lugar, vamos ver como essa conexão é estabelecida. Suponha que um processo que roda em um hospedeiro (cliente) queira iniciar uma conexão com outro processo em outro hospedeiro (servidor). O processo de aplicação cliente primeiramente informa ao TCP cliente que quer estabelecer uma conexão com um processo no servidor. O TCP no cliente então estabelece uma conexão TCP com o TCP no servidor da seguinte maneira:

Etapa 1. O lado cliente do TCP primeiramente envia um segmento TCP especial ao lado servidor do TCP. Esse segmento especial não contém nenhum dado de camada de aplicação, mas um dos bits de flag no seu cabeçalho (veja a Figura 3.29), o bit SYN, é ajustado para 1. Por essa razão, esse segmento é denominado um segmento SYN. Além disso, o cliente escolhe aleatoriamente um número de sequência inicial (`client_isn`) e coloca esse número no campo de número de sequência do segmento TCP SYN inicial. Esse segmento é encapsulado em um datagrama IP e enviado ao servidor. A aleatoriedade adequada da escolha de `client_isn` de modo a evitar certos ataques à segurança tem despertado considerável interesse [CERT 2001-09].

Etapa 2. Assim que o datagrama IP contendo o segmento TCP SYN chega ao hospedeiro servidor (admitindo-se que ele realmente chegue!), o servidor extraí o segmento TCP SYN do datagrama, aloca buffers e variáveis TCP à conexão e envia um segmento de aceitação de conexão ao TCP cliente. (Veremos, no Capítulo 8, que a alocação desses buffers e variáveis, antes da conclusão da terceira etapa da apresentação de três vias, torna o TCP vulnerável a um ataque de recusa de serviço conhecido como inundação SYN.) Esse segmento de aceitação de conexão também não contém nenhum dado de camada de aplicação. Contudo, contém três informações importantes no cabeçalho do segmento: o bit SYN está com valor 1; o campo de reconhecimento do cabeçalho do segmento TCP está ajustado para `client_isn+1`; e, por fim, o servidor escolhe seu próprio número de sequência inicial (`server_isn`) e coloca esse valor no campo de número de sequência do cabeçalho do segmento TCP. Esse segmento de aceitação de conexão está dizendo, com efeito, “Recebi seu pacote SYN para começar uma conexão com seu número de sequência inicial `client_isn`. Concordo em estabelecer essa conexão. Meu número de sequência inicial é `server_isn`”. O segmento de concessão da conexão às vezes é denominado **segmento SYNACK**.

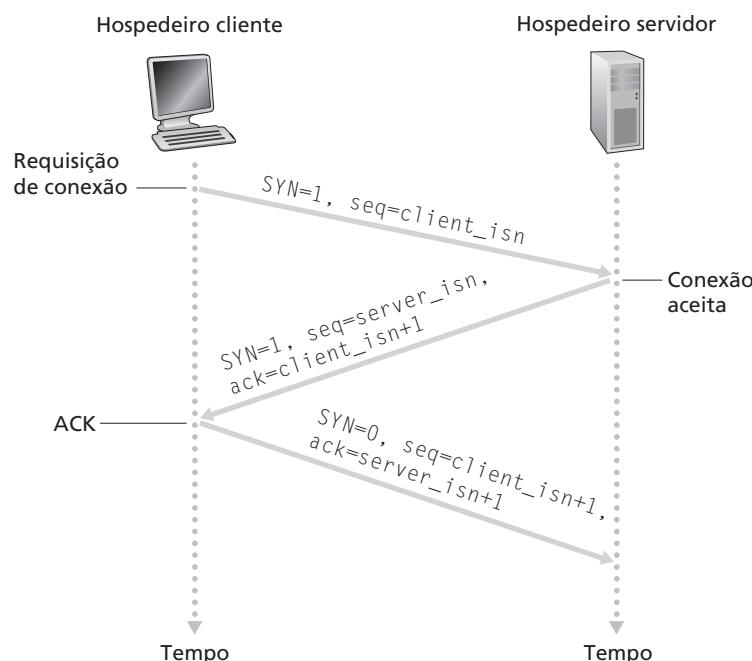


Figura 3.39 Apresentação de três vias do TCP: troca de segmentos

Etapa 3. Ao receber o segmento SYNACK, o cliente também reserva buffers e variáveis para a conexão. O hospedeiro cliente então envia ao servidor mais um segmento. Esse último segmento reconhece o segmento de confirmação da conexão do servidor (o cliente o faz colocando o valor `server_isn+1` no campo de reconhecimento do cabeçalho do segmento TCP). O bit SYN é ajustado para 0, já que a conexão está estabelecida. A terceira etapa da apresentação de três vias pode conduzir os dados cliente/servidor na carga útil do segmento.

Completadas as três etapas, os hospedeiros cliente e servidor podem enviar segmentos contendo dados um ao outro. Em cada um desses futuros segmentos, o bit SYN estará ajustado para 0. Note que, para estabelecer a conexão, três pacotes são enviados entre dois hospedeiros, como ilustra a Figura 3.39. Por essa razão, esse procedimento de estabelecimento de conexão é frequentemente denominado **apresentação de três vias**. Vários aspectos da apresentação de três vias do TCP são tratados nos exercícios ao final deste capítulo (Por que são necessários os números de sequência iniciais? Por que é preciso uma apresentação de três vias, e não apenas de duas vias?) É interessante notar que um alpinista e seu amarrador (que fica mais abaixo e cuja tarefa é passar a corda de segurança ao alpinista) usa um protocolo de comunicação de apresentação de três vias idêntico ao do TCP para garantir que ambos os lados estejam prontos antes de o alpinista iniciar a escalada.

Tudo o que é bom dura pouco, e o mesmo é válido para uma conexão TCP. Qualquer um dos dois processos que participam de uma conexão TCP pode encerrar a conexão. Quando a conexão termina, os ‘recursos’ (isto é, os buffers e as variáveis) nos hospedeiros são liberados. Como exemplo, suponha que o cliente decida encerrar a conexão, como mostra a Figura 3.40. O processo de aplicação cliente emite um comando para fechar. Isso faz com que o TCP cliente envie um segmento TCP especial ao processo servidor, cujo bit de flag no cabeçalho do segmento, denominado bit FIN (veja a Figura 3.39), tem valor ajustado em 1. Quando o servidor recebe esse segmento, ele envia de volta ao cliente um segmento de reconhecimento. O servidor então envia seu próprio segmento de encerramento, que tem o bit FIN ajustado em 1. Por fim, o cliente reconhece o segmento de encerramento do servidor. Nesse ponto, todos os recursos dos dois hospedeiros estão liberados.

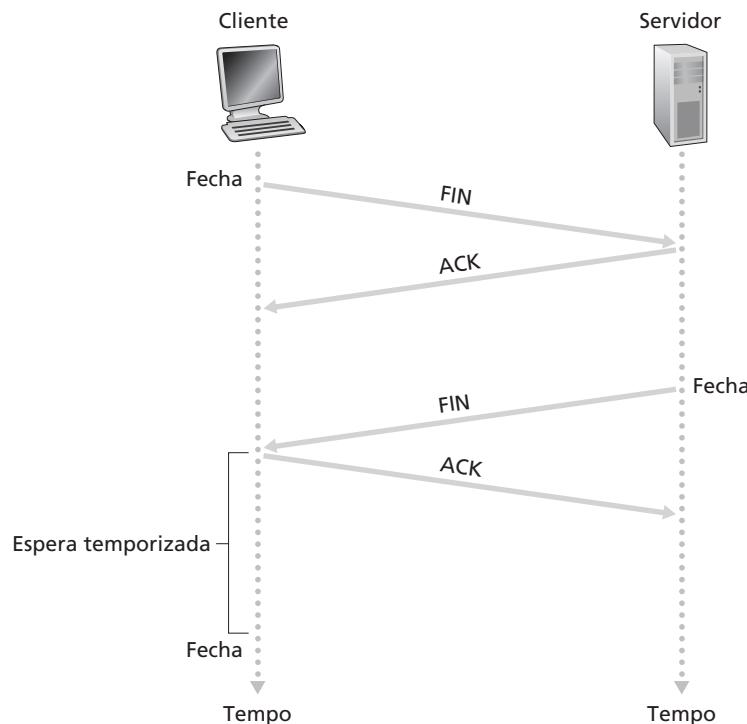


Figura 3.40 Encerramento de uma conexão TCP

Durante a vida de uma conexão TCP, o protocolo TCP que roda em cada hospedeiro faz transições pelos vários **estados do TCP**. A Figura 3.41 ilustra uma sequência típica de estados do TCP visitados pelo TCP cliente. O TCP cliente começa no estado CLOSED. A aplicação no lado cliente inicia uma nova conexão TCP (criando um objeto socket como nos exemplos em Java do Capítulo 2). Isso faz com que o TCP no cliente envie um segmento SYN ao TCP no servidor. Após ter enviado o segmento SYN, o TCP cliente entra no estado SYN_SENT e, enquanto isso, o TCP cliente espera por um segmento do TCP servidor que inclui um reconhecimento para o segmento anterior do cliente, e tem o bit SYN ajustado para o valor 1. Assim que recebe esse segmento, o TCP cliente entra no estado ESTABLISHED, quando pode enviar e receber segmentos TCP que contêm carga útil de dados (isto é, gerados pela aplicação).

Suponha que a aplicação cliente decida que quer fechar a conexão. (Note que o servidor também tem a alternativa de fechá-la.) Isso faz com que o TCP cliente envie um segmento TCP com o bit FIN ajustado em 1 e entre no estado FIN_WAIT_1. No estado FIN_WAIT_1, o TCP cliente espera por um segmento TCP do servidor com um reconhecimento. Quando recebe esse segmento, o TCP cliente entra no estado FIN_WAIT_2. No estado FIN_WAIT_2, ele espera por outro segmento do servidor com o bit FIN ajustado para 1. Após receber esse segmento, o TCP cliente reconhece o segmento do servidor e entra no estado TIME_WAIT. O estado TIME_WAIT permite que o TCP cliente reenvie o reconhecimento final, caso o ACK seja perdido. O tempo passado no estado TIME_WAIT depende da implementação, mas os valores típicos são 30 segundos, 1 minuto e 2 minutos. Após a espera, a conexão se encerra formalmente e todos os recursos do lado cliente (inclusive os números de porta) são liberados.

A Figura 3.42 ilustra a série de estados normalmente visitados pelo TCP do lado servidor, admitindo-se que é o cliente quem inicia o encerramento da conexão. As transições são autoexplicativas. Nesses dois diagramas de transição de estados, mostramos apenas como uma conexão TCP é normalmente estabelecida e fechada. Não descrevemos o que acontece em certos cenários patológicos, por exemplo, quando ambos os lados de uma conexão querem fechar ao mesmo tempo. Se estiver interessado em aprender mais sobre esse assunto e sobre outros mais avançados referentes ao TCP, consulte o abrangente livro de [Stevens, 1994].

Nossa discussão acima concluiu que o cliente e o servidor estão preparados para se comunicar, isto é, que o servidor está ouvindo na porta pela qual o cliente envia seu segmento SYN.

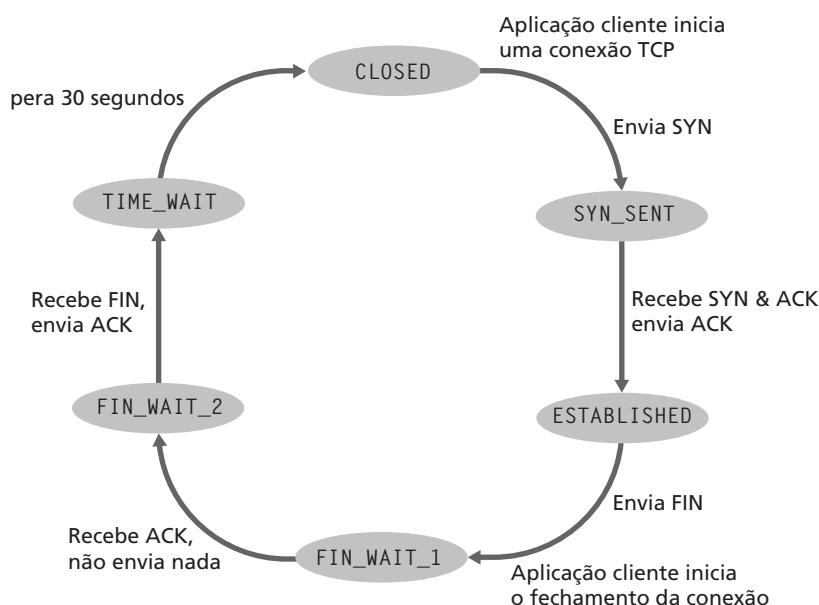


Figura 3.41 Uma sequência típica de estados do TCP visitados por um TCP cliente



Segurança em foco

O ataque Syn Flood

Vimos em nossa discussão sobre a apresentação de três vias do TCP que um servidor aloca e inicializa as variáveis da conexão e os buffers em resposta ao SYN recebido. O servidor, então, envia um SYNACK em resposta e aguarda um segmento ACK do cliente, o terceiro e último passo na apresentação antes de uma conexão ser completamente estabelecida. Se o cliente não enviar um ACK para completar o terceiro passo da apresentação de três vias, com o tempo (geralmente após um minuto ou mais) o servidor finalizará a conexão semiaberta e recuperará os recursos alocadas.

Esse protocolo de gerenciamento da conexão TCP abre caminho para um ataque DoS clássico, ou seja, o ataque SYN flood. Neste ataque, o vilão envia um grande número de segmentos SYN TCP, sem concluir a terceira etapa de apresentação. O ataque pode ser ampliado enviando os SYNs de diversas fontes, criando um ataque DDoS (recusa de serviço distribuído) SYN flood. Com esse acúmulo de segmentos SYN, os recursos de conexão do servidor podem rapidamente se esgotar já que são alocadas (mas nunca usadas) para conexões semiabertas. Esgotando-se os recursos do servidor, o serviço é negado ao verdadeiro cliente. Esses ataques SYN flood [CERT SYN, 1996] estavam entre os primeiros ataques DoS documentados pelo CERT [CERT, 2009].

O SYN flood é um ataque potencialmente destruidor. Felizmente, há uma proteção eficaz, denominada SYN Cookies [Skoudis, 2006; Cisco SYN, 2009; Bernstein, 2009], agora implementada na maioria dos sistemas operacionais. O SYN Cookie age da seguinte maneira:

- Quando o servidor recebe um segmento SYN, não se sabe se ele vem de um usuário verdadeiro ou se é parte desse ataque. Então, o servidor não cria uma conexão TCP semiaberta para esse SYN. Em vez disso, o servidor cria um número de sequência inicial TCP, uma função hash de endereços de fonte e endereços de destino IP e números de porta do segmento SYN, assim como de um número secreto somente conhecido pelo usuário. (O usuário utiliza o mesmo número secreto para um grande número de conexões.) Esse número de sequência inicial criado cuidadosamente é o assim chamado "cookie". O servidor, então, envia um pacote SYNACK com esse número de sequência especial. É importante mencionar que o servidor não se lembra do cookie ou de qualquer outra informação correspondente ao SYN.
- Se o cliente for verdadeiro, então um segmento ACK retornará. O servidor, ao receber esse ACK, precisa verificar se ele corresponde a algum SYN enviado anteriormente. Como isto é feito se o servidor não guarda nenhuma memória sobre os segmentos SYN? Como você deve ter imaginado, esse processo é realizado com o cookie. Para um ACK legítimo, especificamente, o valor no campo de reconhecimento é igual ao número de sequência no SYNACK mais um (veja Figura 3.39). O servidor, então, executará a mesma função utilizando os mesmos campos no segmento ACK e número secreto. Se o resultado da função mais um for o mesmo que o número de reconhecimento, o servidor conclui que o ACK corresponde a um segmento SYN anterior e, portanto, é válido. O servidor, então, cria uma conexão totalmente aberta com um socket.
- Por outro lado, se o cliente não retorna um segmento ACK, então o SYN original não causou nenhum dano ao servidor, uma vez que este não alocou nenhum recurso para ele!

Os SYN cookies eliminam, efetivamente, a ameaça de um ataque SYN flood. Uma variação desse ataque é o cliente malicioso retornar um segmento ACK válido para cada segmento SYNACK que o servidor gera. Isto fará com que o servidor estabeleça conexões TCP totalmente abertas, mesmo se seu sistema operacional utilizar SYN cookies. Se centenas de clientes estiverem sendo usados (ataque DDoS), cada um de uma fonte de endereço IP diferente, então é difícil o servidor reconhecer as fontes verdadeiras e as maliciosas. Assim, pode ser mais difícil de se proteger desse "ataque de apresentação completado" do que o clássico ataque SYN flood.

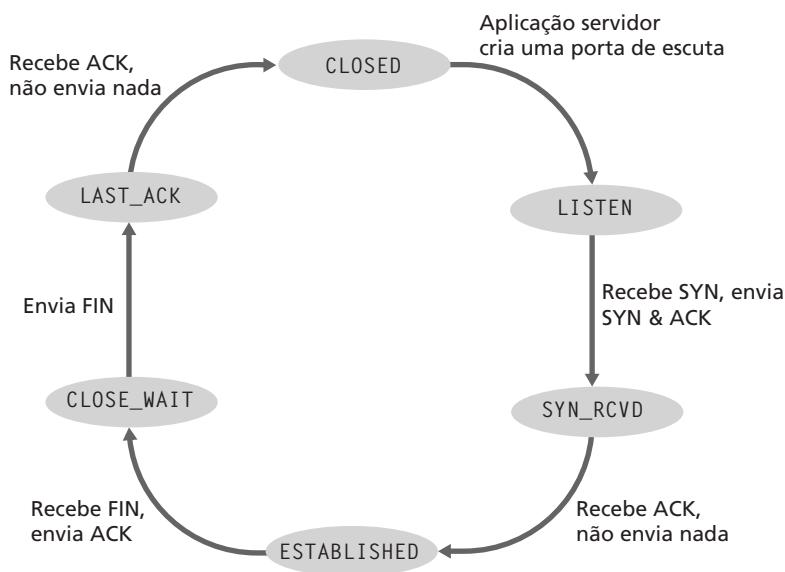


Figura 3.42 Uma sequência típica de estados do TCP visitados por um TCP do lado do servidor

Vamos considerar o que acontece quando um hospedeiro recebe um segmento TCP cujos números de porta ou endereço IP não são compatíveis com nenhuma das portas existentes no hospedeiro. Por exemplo, suponha que um hospedeiro receba um pacote TCP SYN com porta de destino 80, mas não está aceitando conexões nessa porta (isto é, não está rodando um servidor Web na porta 80). Então, ele enviará à fonte um segmento especial de reinicialização. Esse segmento TCP tem o bit de flag RST ajustado para 1 (veja Seção 3.5.2). Assim, quando um hospedeiro envia um segmento de reinicialização ele está dizendo à fonte: “Eu não tenho um socket para esse segmento. Favor não enviá-lo novamente”. Quando um hospedeiro recebe um pacote UDP cujo número de porta de destino não é compatível com as portas de um UDP em curso, ele envia um datagrama ICMP especial, como será discutido no Capítulo 4.

Agora que obtivemos uma boa compreensão sobre gerenciamento da conexão TCP, vamos voltar à ferramenta de varredura de porta nmap e analisar mais precisamente como ela funciona. Para explorar uma porta TCP, digamos que a porta 6789, o nmap enviará ao computador-alvo um segmento TCP SYN com a porta de destino. Os três possíveis resultados são:

- O computador-fonte recebe um segmento TCP SYNACK de um computador-alvo. Como isso significa que uma aplicação está sendo executada com a porta TCP 6789 no computador-alvo, o nmap retorna “aberto”.
- O computador-fonte recebe um segmento TCP RST de um computador-alvo. Isto significa que o segmento SYN atingiu o computador-alvo, mas este não está executando uma aplicação com a porta TCP 6789. Mas o atacante, pelo menos, sabe que os segmentos destinados ao computador na porta 6789 não estão bloqueados pelo firewall no percurso entre o computador-fonte e o alvo. (Firewalls são abordados no Capítulo 8).
- A fonte não recebe nada. Isto, provavelmente, significa que o segmento SYN foi bloqueado pelo firewall e nunca atingiu o computador-alvo.

O nmap é uma ferramenta potente, que pode “examinar o local” não somente para abrir portas TCP, mas também para abrir portas UDP, para firewalls e suas configurações, e até mesmo para as versões de aplicações e sistemas operacionais. A maior parte disto é feito através da manipulação dos segmentos de gerenciamento da conexão TCP [Skoudis, 2006]. Caso você esteja sentado próximo a uma máquina Linux, talvez você queira fazer uma experiência com o nmap apenas digitando o nome da ferramenta na linha de comando. É possível fazer download do nmap para outros sistemas operacionais do site <http://insecure.org/nmap>.

Com isso, concluímos nossa introdução ao controle de erro e controle de fluxo em TCP. Voltaremos ao TCP na Seção 3.7 e então examinaremos mais detalhadamente o controle de congestionamento. Antes, contudo, vamos analisar a questão do controle de congestionamento em um contexto mais amplo.

3.6 Princípios de controle de congestionamento

Nas seções anteriores, examinamos os princípios gerais e também os mecanismos específicos do TCP usados para prover um serviço de transferência confiável de dados em relação à perda de pacotes. Mencionamos anteriormente que, na prática, essa perda resulta, caracteristicamente, de uma saturação de buffers de roteadores à medida que a rede fica congestionada. Assim, a retransmissão de pacotes trata de um sintoma de congestionamento de rede (a perda de um segmento específico de camada de transporte), mas não trata da causa do congestionamento da rede: demasiadas fontes tentando enviar dados a uma taxa muito alta. Para tratar da causa do congestionamento de rede, são necessários mecanismos para regular os remetentes quando esse congestionamento ocorre.

Nesta seção, consideramos o problema do controle de congestionamento em um contexto geral, buscando entender por que o congestionamento é algo ruim, como o congestionamento de rede se manifesta no desempenho recebido por aplicações da camada superior e várias medidas que podem ser adotadas para evitar o congestionamento de rede ou reagir a ele. Esse estudo mais geral do controle de congestionamento é apropriado, já que, como acontece com a transferência confiável de dados, o congestionamento é um dos “dez maiores” da lista de problemas fundamentalmente importantes no trabalho em rede. Concluímos esta seção com uma discussão sobre o controle de congestionamento no serviço de **taxa de bits disponível** (*available bit-rate* — ABR) em redes com **modo de transferência assíncrono** (ATM). A seção seguinte contém um estudo detalhado do algoritmo de controle de congestionamento do TCP.

3.6.1 As causas e os custos do congestionamento

Vamos começar nosso estudo geral do controle de congestionamento examinando três cenários de complexidade crescente nos quais ocorre o congestionamento. Em cada caso, examinaremos, primeiramente, por que ele ocorre e, depois, seu custo (no que se refere aos recursos não utilizados integralmente e ao baixo desempenho recebido pelos sistemas finais). Não focalizaremos (ainda) como reagir ao congestionamento, ou evitá-lo; preferimos estudar uma questão mais simples, que é entender o que acontece quando hospedeiros aumentam sua taxa de transmissão e a rede fica congestionada.

Cenário 1: dois remetentes, um roteador com buffers infinitos

Começamos considerando o que talvez seja o cenário de congestionamento mais simples possível: dois hospedeiros (A e B), cada um com uma conexão que compartilha um único trecho de rede entre a fonte e o destino, como mostra a Figura 3.43.

Vamos admitir que a aplicação no hospedeiro A esteja enviando dados para a conexão (por exemplo, passando dados para o protocolo de camada de transporte por um socket) a uma taxa média de λ_{in} bytes/segundo. Esses dados são originais no sentido de que cada unidade de dados é enviada para dentro do socket apenas uma vez. O protocolo de camada de transporte subjacente é simples. Os dados são encapsulados e enviados; não há recuperação de erros (por exemplo, retransmissão), controle de fluxo, nem controle de congestionamento. Desprezando a sobrecarga adicional causada pela adição de informações de cabeçalhos de camada de transporte e de camadas mais baixas, a taxa à qual o hospedeiro A oferece tráfego ao roteador nesse primeiro cenário é λ_{in} bytes/segundo. O hospedeiro B funciona de maneira semelhante, e admitimos, por simplicidade, que ele também esteja enviando dados a uma taxa de λ_{in} bytes/segundo. Os pacotes dos hospedeiros A e B passam por um roteador e por um enlace de saída compartilhado de capacidade R . O roteador tem buffers que lhe permitem armazenar os pacotes que chegam quando a taxa de chegada de pacotes excede a capacidade do enlace de saída. No primeiro cenário, admitimos que o roteador tenha capacidade de armazenamento infinita.

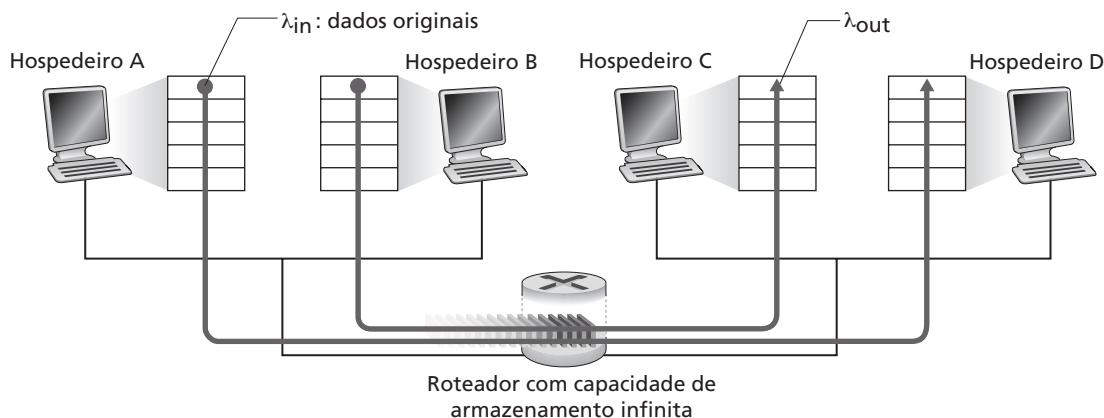


Figura 3.43 Cenário de congestionamento 1: duas conexões compartilhando um único roteador com número infinito de buffers

A Figura 3.44 apresenta o desempenho da conexão do hospedeiro A nesse primeiro cenário. O gráfico da esquerda dessa figura apresenta a **vazão por conexão** (número de bytes por segundo no destinatário) como uma função da taxa de envio da conexão. Para uma taxa de transmissão entre 0 e $R/2$, a vazão no destinatário é igual à velocidade de envio do remetente — tudo o que o remetente envia é recebido no destinatário com um atraso finito. Quando a velocidade de envio estiver acima de $R/2$, contudo, a vazão será somente $R/2$. Esse limite superior da vazão é consequência do compartilhamento da capacidade do enlace entre duas conexões. O enlace simplesmente não consegue entregar os pacotes a um destinatário com uma taxa em regime que excede $R/2$. Não importa quão altas sejam as taxas de envio ajustadas nos hospedeiros A e B, eles jamais alcançarão uma vazão maior do que $R/2$.

Alcançar uma vazão de $R/2$ por conexão pode até parecer uma coisa boa, pois o enlace está sendo integralmente utilizado para entregar pacotes no destinatário. No entanto, o gráfico do lado direito da Figura 3.44 mostra as consequências de operar próximo à capacidade máxima do enlace. À medida que a taxa de envio se aproxima de $R/2$ (partindo da esquerda), o atraso médio fica cada vez maior. Quando a taxa de envio ultrapassa $R/2$, o número médio de pacotes na fila no roteador é ilimitado e o atraso médio entre a fonte e o destino se torna infinito (admitindo que as conexões operem a essas velocidades de transmissão durante um período de tempo infinito e que a capacidade de armazenamento também seja infinita). Assim, embora operar a uma vazão agregada próxima a R possa ser ideal do ponto de vista da vazão, está bem longe de ser ideal do ponto de vista do atraso. Mesmo

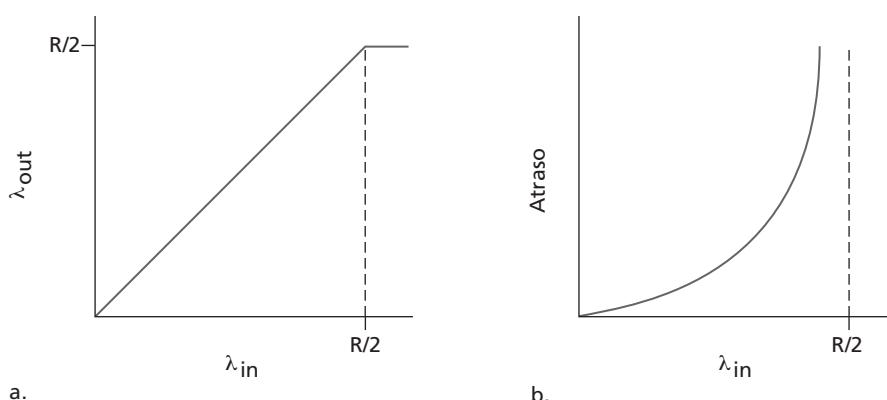


Figura 3.44 Cenário de congestionamento 1: vazão e atraso em função da taxa de envio do hospedeiro

nesse cenário (extremamente) idealizado, já descobrimos um custo da rede congestionada — há grandes atrasos de fila quando a taxa de chegada de pacotes se aproxima da capacidade do enlace.

Cenário 2: dois remetentes, um roteador com buffers finitos

Vamos agora modificar ligeiramente o cenário 1 dos dois modos seguintes (veja a Figura 3.45). Em primeiro lugar, admitamos que a capacidade de armazenamento do roteador seja finita. Em uma situação real, essa suposição teria como consequência o descarte de pacotes que chegam a um buffer que já está cheio. Em segundo lugar, admitimos que cada conexão seja confiável. Se um pacote contendo um segmento de camada de transporte for descartado no roteador, o remetente por fim o retransmitirá. Como os pacotes podem ser retransmitidos, agora temos de ser mais cuidadosos com o uso da expressão “taxa de envio”. Especificamente, vamos novamente designar a taxa com que a aplicação envia dados originais para dentro do socket como λ_{in} bytes/segundo. A taxa com que a camada de transporte envia segmentos (contendo dados originais e dados retransmitidos) para dentro da rede será denominada λ'_{in} bytes/segundo. Essa taxa (λ'_{in}) às vezes é denominada **carga oferecida** à rede.

O desempenho obtido no cenário 2 agora dependerá muito de como a retransmissão é realizada. Primeiramente, considere o caso não realista em que o hospedeiro A consiga, de algum modo (fazendo mágica!), determinar se um buffer do roteador está livre ou não. Assim, o hospedeiro A envia um pacote somente quando o buffer estiver livre. Nesse caso, não ocorreria nenhuma perda, λ_{in} seria igual a λ'_{in} e a vazão da conexão seria igual a λ_{in} . Esse caso é mostrado pela curva superior da Figura 3.46(a).

Do ponto de vista da vazão, o desempenho é ideal — tudo o que é enviado é recebido. Note que, nesse cenário, a taxa média de envio do hospedeiro não pode ultrapassar $R/2$, já que admitimos que nunca ocorre perda de pacote.

Considere, em seguida, o caso um pouco mais realista em que o remetente retransmite somente quando sabe, com certeza, que o pacote foi perdido. (Novamente, essa suposição é um pouco forçada. Contudo, é possível ao hospedeiro remetente ajustar seu temporizador de retransmissão para uma duração longa o suficiente para ter razoável certeza de que um pacote que não foi reconhecido foi perdido.) Nesse caso, o desempenho pode ser parecido com o que é mostrado na Figura 3.46(b). Para avaliar o que está acontecendo aqui, considere o caso em que a carga oferecida, λ_{in} (a taxa de transmissão dos dados originais mais as retransmissões) é igual a $R/2$. De acordo com a Figura 3.46(b), nesse valor de carga oferecida, a velocidade com a qual os dados são entregues à aplicação do destinatário é $R/3$. Assim, de $0,5R$ unidade de dados transmitida, $0,333R$ byte/segundo (em média) são dados originais e $0,166R$ byte/segundo (em média) são dados retransmitidos. *Observamos aqui outro custo de*

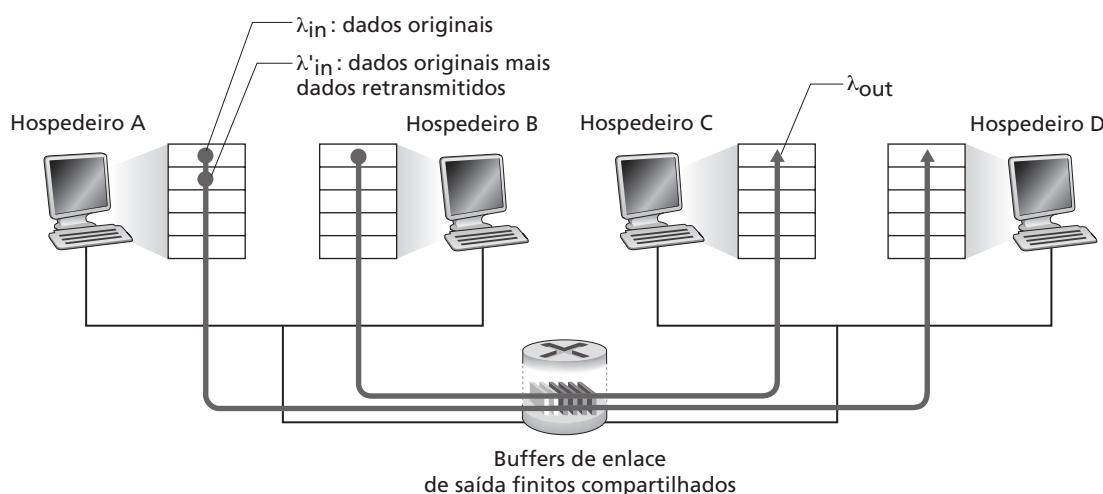


Figura 3.45 Cenário 2: dois hospedeiros (com retransmissões) e um roteador com buffers finitos

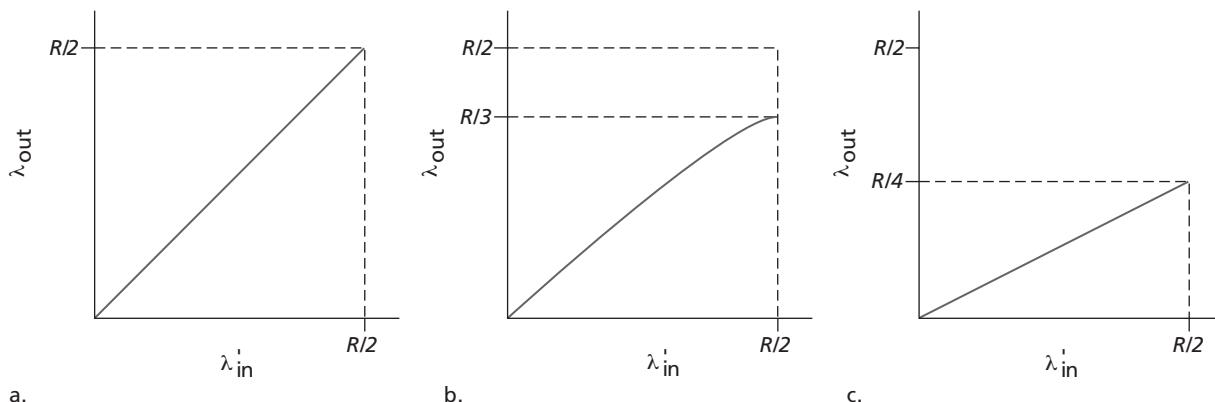


Figura 3.46 Desempenho no cenário 2 com buffers finitos

uma rede congestionada — o remetente deve realizar retransmissões para compensar os pacotes descartados (perdidos) devido ao esgotamento do buffer.

Finalmente, vamos considerar o caso em que a temporização do remetente se esgote prematuramente e ele retransmita um pacote que ficou atrasado na fila, mas que ainda não está perdido. Aqui, tanto o pacote de dados original quanto a retransmissão podem alcançar o destinatário. É claro que o destinatário precisa apenas de uma cópia desse pacote e descartará a retransmissão. Nesse caso, o trabalho realizado pelo roteador ao repassar a cópia retransmitida do pacote original é desperdiçado, pois o destinatário já terá recebido a cópia original desse pacote. Em vez disso, seria melhor o roteador usar a capacidade de transmissão do enlace para enviar um pacote diferente. *Eis aqui mais um custo da rede congestionada — retransmissões desnecessárias feitas pelo remetente em face de grandes atrasos podem fazer com que um roteador use sua largura de banda de enlace para repassar cópias desnecessárias de um pacote.* A Figura 3.4.6(c) mostra a vazão versus a carga oferecida admitindo-se que cada pacote seja enviado (em média) duas vezes pelo roteador. Visto que cada pacote é enviado duas vezes, a vazão terá um valor assintótico de $R/4$ à medida que a carga oferecida se aproximar de $R/2$.

Cenário 3: quatro remetentes, roteadores com buffers finitos e trajetos com múltiplos roteadores

Em nosso cenário final de congestionamento, quatro hospedeiros transmitem pacotes sobre trajetos sobrepostos que apresentam dois saltos, como ilustrado na Figura 3.47. Novamente admitamos que cada hospedeiro use um mecanismo de temporização/retransmissão para implementar um serviço de transferência confiável de dados, que todos os hospedeiros tenham o mesmo valor de λ_{in} e que todos os enlaces dos roteadores tenham capacidade de R bytes/segundo.

Vamos considerar a conexão do hospedeiro A ao hospedeiro C que passa pelos roteadores R1 e R2. A conexão A–C compartilha o roteador R1 com a conexão D–B e o roteador 2 com a conexão B–D. Para valores extremamente pequenos de λ_{in} , esgotamentos de buffers são raros (como acontecia nos cenários de congestionamento 1 e 2) e a vazão é quase igual à carga oferecida. Para valores de λ_{in} ligeiramente maiores, a vazão correspondente é também maior, pois mais dados originais estão sendo transmitidos para a rede e entregues no destino, e os esgotamentos ainda são raros. Assim, para valores pequenos de λ_{in} , um aumento em λ_{in} resulta em um aumento em λ_{out} .

Como já analisamos o caso de tráfego extremamente baixo, vamos examinar aquele em que λ_{in} (e, portanto, λ'_{in}) é extremamente alto. Considere o roteador R2. O tráfego A–C que chega ao roteador R2 (após ter sido repassado de R1) pode ter uma taxa de chegada em R2 de, no máximo, R , que é a capacidade do enlace de R1 a R2, não importando qual seja o valor de λ_{in} . Se λ'_{in} for extremamente alto para todas as conexões (incluindo a conexão B–D), então a taxa de chegada do tráfego B–D em R2 poderá ser muito maior do que a taxado tráfego A–C. Como os tráfegos A–C e B–D têm de competir no roteador R2 pelo espaço limitado de buffer, a quantidade de tráfego A–C que consegue passar por R2 (isto é, que não se perde devido ao congestionamento de buffer) diminui cada vez mais à medida que

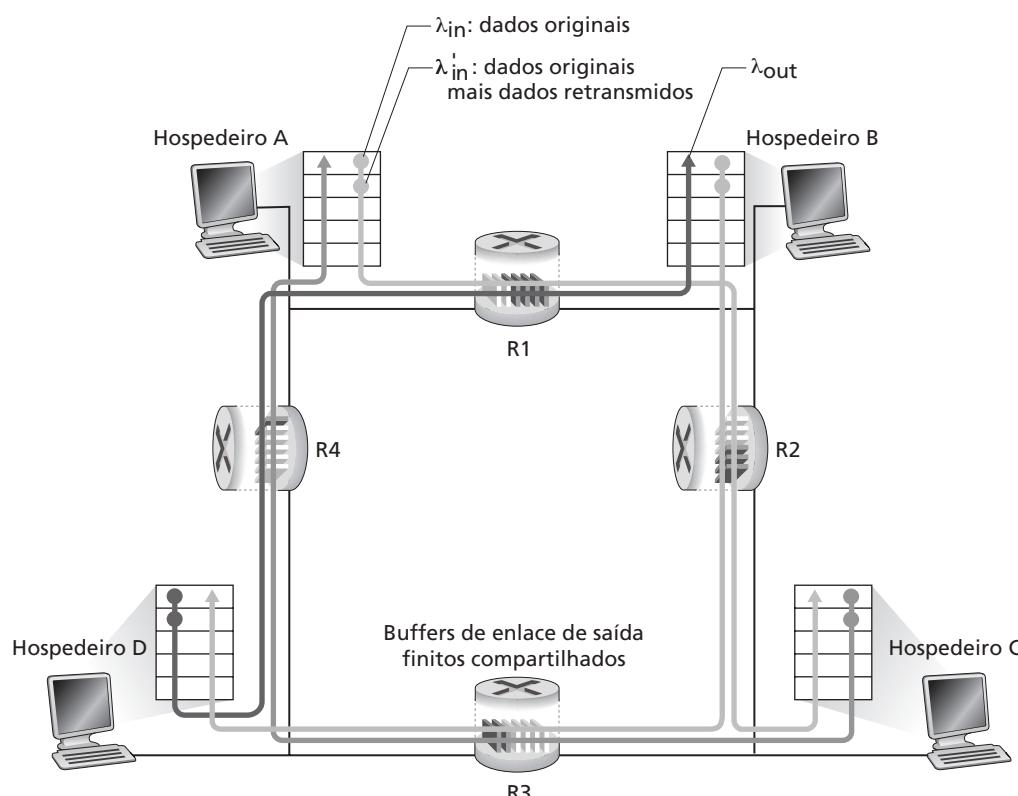


Figura 3.47 Quatro remetentes, roteadores com buffers finitos e trajetos com vários saltos

a carga oferecida de B-D vai ficando maior. No limite, quando a carga oferecida se aproxima do infinito, um buffer vazio em R2 é imediatamente preenchido por um pacote B-D e a vazão da conexão A-C em R2 cai a zero. Isso, por sua vez, implica que a vazão final de A-C vai a zero no limite de tráfego pesado. Essas considerações dão origem ao comportamento da carga oferecida versus a vazão mostrada na Figura 3.48.

A razão para o decréscimo final da vazão com o crescimento da carga oferecida é evidente quando consideramos a quantidade de trabalho desperdiçado realizado pela rede. No cenário de alto tráfego que acabamos de descrever, sempre que um pacote é descartado em um segundo roteador, o trabalho realizado pelo primeiro para enviar o pacote ao segundo acaba sendo ‘desperdiçado’. A rede teria se saído igualmente bem (melhor dizendo,

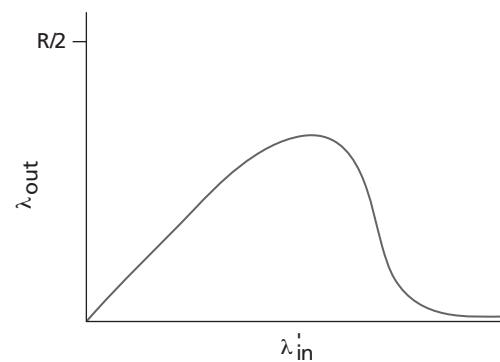


Figura 3.48 Desempenho obtido no cenário 3, com buffers finitos e trajetos com múltiplos roteadores

igualmente mal) se o primeiro roteador tivesse simplesmente descartado aquele pacote e tivesse ficado inativo. Mais objetivamente, a capacidade de transmissão utilizada no primeiro roteador para enviar o pacote ao segundo teria sido maximizada para transmitir um pacote diferente. (Por exemplo, ao selecionar um pacote para transmissão, seria melhor que um roteador desse prioridade a pacotes que já atravessaram alguns roteadores anteriores.) Portanto, vemos aqui mais um custo, o do descarte de pacotes devido ao congestionamento — quando um pacote é descartado ao longo de um caminho, a capacidade de transmissão que foi usada em cada um dos enlaces anteriores para repassar o pacote até o ponto em que foi descartado acaba sendo desperdiçada.

3.6.2 Mecanismos de controle de congestionamento

Na Seção 3.7, examinaremos detalhadamente os mecanismos específicos do TCP para o controle de congestionamento. Nesta subseção, identificaremos os dois procedimentos mais comuns adotados, na prática, para o controle de congestionamento. Além disso, examinaremos arquiteturas específicas de rede e protocolos de controle de congestionamento que incorporam esses procedimentos.

No nível mais amplo, podemos distinguir mecanismos de controle de congestionamento conforme a camada de rede ofereça ou não assistência explícita à camada de transporte com a finalidade de controle de congestionamento:

Controle de congestionamento fim a fim. Nesse método para o controle de congestionamento, a camada de rede não fornece nenhum suporte explícito à camada de transporte com a finalidade de controle de congestionamento. Até mesmo a presença de congestionamento na rede deve ser intuída pelos sistemas finais com base apenas na observação do comportamento da rede (por exemplo, perda de pacotes e atraso). Veremos na Seção 3.7 que o TCP deve necessariamente adotar esse método fim a fim para o controle de congestionamento, uma vez que a camada IP não fornece realimentação de informações aos sistemas finais quanto ao congestionamento da rede. A perda de segmentos TCP (apontada por uma temporização ou por três reconhecimentos duplicados) é tomada como indicação de congestionamento, e o TCP reduz o tamanho da janela de acordo com isso. Veremos também que as novas propostas para o TCP usam valores de atraso de viagem de ida e volta crescentes como indicadores de aumento do congestionamento da rede.

Controle de congestionamento assistido pela rede. Com esse método, os componentes da camada de rede (isto é, roteadores) fornecem realimentação específica de informações ao remetente a respeito do estado de congestionamento na rede. Essa realimentação pode ser tão simples como um único bit indicando o congestionamento em um enlace. Adotada nas primeiras arquiteturas de rede IBM SNA [Schwartz, 1982] e DEC DECnet [Jain, 1989; Ramakrishnan, 1990], essa abordagem foi proposta recentemente para redes TCP/IP [Floyd TCP, 1994; RFC 3168]; ela é usada também no controle de congestionamento em ATM com serviço de transmissão ABR, como discutido a seguir. A realimentação mais sofisticada de rede também é possível. Por exemplo, uma forma de controle de congestionamento ATM ABR que estudaremos mais adiante permite que um roteador informe explicitamente ao remetente a velocidade de transmissão que ele (o roteador) pode suportar em um enlace de saída. O protocolo XCP [Katabi, 2002] provê um retorno (feedback) calculado pelo roteador para cada fonte, transmitido no cabeçalho do pacote, referente ao modo que essa fonte deve aumentar ou diminuir sua taxa de transmissão.

Para controle de congestionamento assistido pela rede, a informação sobre congestionamento é normalmente realimentada da rede para o remetente por um de dois modos, como mostra a Figura 3.49. Realimentação direta pode ser enviada de um roteador de rede ao remetente. Esse modo de notificação normalmente toma a forma de um **pacote de congestionamento** (*choke packet*) (que, em essência, diz: “Estou congestionado!”). O segundo modo de notificação ocorre quando um roteador marca/atualiza um campo em um pacote que está fluindo do remetente ao destinatário para indicar congestionamento. Ao receber um pacote marcado, o destinatário informa ao remetente a indicação de congestionamento. Note que esse último modo de notificação leva, no mínimo, o tempo total de uma viagem de ida e volta.

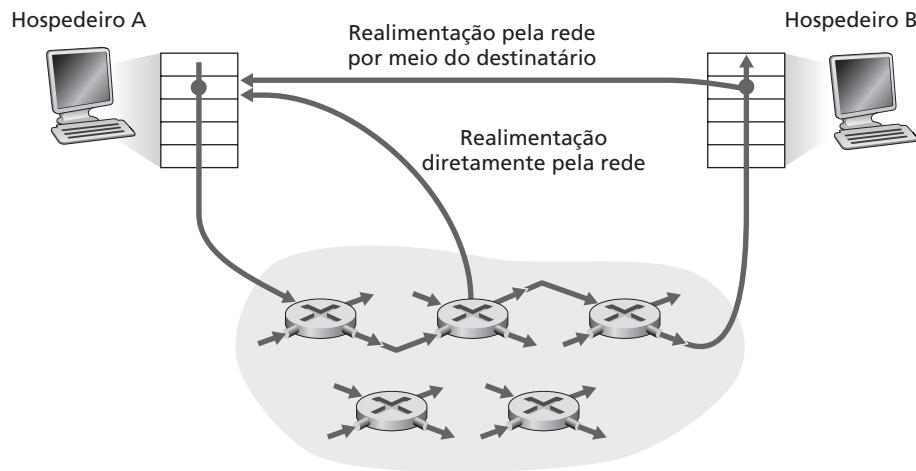


Figura 3.49 Dois caminhos de realimentação para informação sobre congestionamento indicado pela rede

3.6.3 Exemplo de controle de congestionamento assistido pela rede: controle de congestionamento ATM ABR

Concluímos esta seção com um breve estudo de caso do algoritmo de controle de congestionamento ATM ABR. Salientamos que nossa meta aqui *não* é, de modo algum, descrever detalhadamente aspectos da arquitetura ATM, mas mostrar um protocolo que adota uma abordagem de controle de congestionamento notavelmente diferente da adotada pelo protocolo TCP da Internet. Na verdade, apresentamos a seguir apenas os poucos aspectos da arquitetura ATM necessários para entender o controle de congestionamento ABR.

Fundamentalmente, o ATM adota uma abordagem orientada para circuito virtual (CV) da comutação de pacotes. Lembre, da nossa discussão no Capítulo 1, que isso significa que cada comutador no caminho entre a fonte e o destino manterá estado sobre o CV entre a fonte e o destino. Esse estado por CV permite que um comutador monitore o comportamento de remetentes individuais (por exemplo, monitorando sua taxa média de transmissão) e realize ações específicas de controle de congestionamento (tais como sinalizar explicitamente ao remetente para que ele reduza sua taxa quando o comutador fica congestionado). Esse estado por CV em comutadores de rede torna o ATM idealmente adequado para realizar controle de congestionamento assistido pela rede.

O ABR foi projetado como um serviço de transferência de dados elástico que faz lembrar, de certo modo, o TCP. Quando a rede está vazia, o serviço ABR deve ser capaz de aproveitar a vantagem da largura de banda disponível. Quando a rede está congestionada, deve limitar sua taxa de transmissão a algum valor mínimo predefinido de taxa de transmissão. Um tutorial detalhado sobre controle de congestionamento e gerenciamento de tráfego ATM ABR é fornecido por [Jain, 1996].

A Figura 3.50 mostra a estrutura do controle de congestionamento para ATM ABR. Nessa discussão, adotaremos a terminologia ATM (por exemplo, usaremos o termo ‘comutador’ em vez de ‘roteador’ e o termo ‘célula’ em vez de ‘pacote’). Com o serviço ATM ABR, as células de dados são transmitidas de uma fonte a um destino por meio de uma série de comutadores intermediários. Intercaladas às células de dados estão as **células de gerenciamento de recursos**, ou **células RM** (resource-management cells); essas células podem ser usadas para portar informações relacionadas ao congestionamento entre hospedeiros e comutadores. Quando uma célula RM chega a um destinatário, ela será ‘virada’ e enviada de volta ao remetente (possivelmente após o destinatário ter modificado o conteúdo dela). Também é possível que um comutador gere, ele mesmo, uma célula RM e a envie diretamente a uma fonte. Assim, células RM podem ser usadas para fornecer realimentação direta da rede e também realimentação da rede por intermédio do destinatário, como mostra a Figura 3.50.

O controle de congestionamento ATM ABR é um método baseado em taxa. O remetente estima explicitamente uma taxa máxima na qual pode enviar e se autorregula de acordo com ela. O serviço ABR provê três mecanismos para sinalizar informações relacionadas a congestionamento dos comutadores ao destinatário:

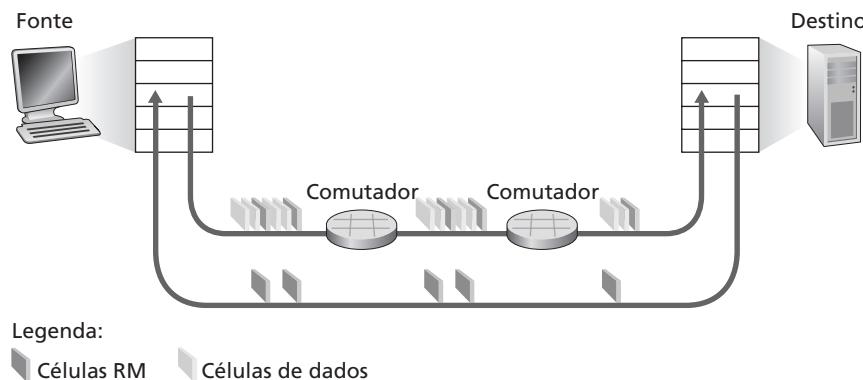


Figura 3.50 Estrutura de controle de congestionamento ATM ABR

Bit EFCI. Cada célula de dados contém um **bit EFCI de indicação explícita de congestionamento** à frente (*explicit forward congestion indication*). Um comutador de rede congestionado pode modificar o bit EFCI dentro de uma célula para 1, a fim de sinalizar congestionamento ao hospedeiro destinatário, que deve verificar o bit EFCI em todas as células de dados recebidas. Quando uma célula RM chega ao destinatário, se a célula de dados recebida mais recentemente tiver o bit EFCI com valor 1, o destinatário modifica o bit de indicação de congestionamento (o bit CI) da célula RM para 1 e envia a célula RM de volta ao remetente. Usando o bit EFCI em células de dados e o bit CI em células RM, um remetente pode ser notificado sobre congestionamento em um comutador da rede.

Bits CI e NI. Como observado anteriormente, células RM remetente/destinatário estão intercaladas com células de dados. A taxa de intercalação de células RM é um parâmetro ajustável, sendo uma célula RM a cada 32 células de dados o valor default. Essas células RM têm um bit de indicação de congestionamento (**CI**) e um bit **NI** (*no increase* — não aumentar) que podem ser ajustados por um comutador de rede congestionado. Especificamente, um comutador pode modificar para 1 o bit NI de uma célula RM que está passando quando a condição de congestionamento é leve e pode modificar o bit CI para 1 sob severas condições de congestionamento. Quando um hospedeiro destinatário recebe uma célula RM, ele a enviará de volta ao remetente com seus bits CI e NI inalterados (exceto que o CI pode ser ajustado para 1 pelo destinatário como resultado do mecanismo EFCI descrito antes).

Ajuste de ER. Cada célula RM contém também um **campo ER** (*explicit rate* — taxa explícita) de dois bytes. Um comutador congestionado pode reduzir o valor contido no campo ER de uma célula RM que está passando. Desse modo, o campo ER será ajustado para a taxa mínima suportável por todos os comutadores no trajeto fonte-destino.

Uma fonte ATM ABR ajusta a taxa na qual pode enviar células como uma função dos valores de CI, NI e ER de uma célula RM devolvida. As regras para fazer esse ajuste de taxa são bastante complicadas e um tanto tediosas. O leitor poderá consultar [Jain, 1996] se quiser saber mais detalhes.

3.7 Controle de congestionamento no TCP

Nesta seção, voltamos ao estudo do TCP. Como aprendemos na Seção 3.5, o TCP provê um serviço de transferência confiável entre dois processos que rodam em hospedeiros diferentes. Outro componente extremamente importante do TCP é seu mecanismo de controle de congestionamento. Como indicamos na seção anterior, o TCP deve usar controle de congestionamento fim a fim em vez de controle de congestionamento assistido pela rede, já que a camada IP não fornece aos sistemas finais realimentação explícita relativa ao congestionamento da rede.

A abordagem adotada pelo TCP é obrigar cada remetente a limitar a taxa à qual enviam tráfego para sua conexão como uma função do congestionamento de rede percebido. Se um remetente TCP perceber que há pouco congestionamento no caminho entre ele e o destinatário, aumentará sua taxa de envio; se perceber que há congestionamento ao longo do caminho, reduzirá sua taxa de envio. Mas essa abordagem levanta três questões. A primeira é como um remetente TCP limita a taxa à qual envia tráfego para sua conexão? A segunda é como um remetente TCP percebe que há congestionamento entre ele e o destinatário? E a terceira, que algoritmo o remetente deve utilizar para modificar sua taxa de envio como uma função do congestionamento fim a fim?

Em primeiro lugar, vamos examinar como um remetente TCP limita a taxa de envio à qual envia tráfego para sua conexão. Na Seção 3.5, vimos que cada lado de uma conexão TCP consiste em um buffer de recepção, um buffer de envio e diversas variáveis (`LastByteRead`, `rwnd` e assim por diante). O mecanismo de controle de congestionamento que opera no remetente monitora uma variável adicional, a **janela de congestionamento**. A janela de congestionamento, denominada `cwnd`, impõe uma limitação à taxa à qual um remetente TCP pode enviar tráfego para dentro da rede. Especificamente, a quantidade de dados não reconhecidos em um hospedeiro não pode exceder o mínimo de `cwnd` e `rwnd`, ou seja:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$$

Para focalizar a discussão no controle de congestionamento (ao contrário do controle de fluxo), daqui em diante vamos admitir que o buffer de recepção TCP seja tão grande que a limitação da janela de recepção pode ser desprezada; assim, a quantidade de dados não reconhecidos no remetente estará limitada exclusivamente por `cwnd`. Vamos admitir também que o remetente sempre tenha dados para enviar, isto é, que todos os segmentos dentro da janela de congestionamento sejam enviados.

A restrição acima limita a quantidade de dados não reconhecidos no remetente e, por conseguinte, limita indiretamente a taxa de envio do remetente. Para entender melhor, considere uma conexão na qual perdas e atrasos de transmissão de pacotes sejam desprezíveis. Então, em linhas gerais, no início de cada RTT a limitação permite que o remetente envie `cwnd` bytes de dados para a conexão; ao final do RTT, o remetente recebe reconhecimentos para os dados. Assim, a taxa de envio do remetente é aproximadamente `cwnd/RTT bytes por segundo`. Portanto, ajustando o valor de `cwnd`, o remetente pode ajustar a taxa à qual envia dados para sua conexão.

Em seguida, vamos considerar como um remetente TCP percebe que há congestionamento no caminho entre ele e o destino. Definimos “evento de perda” em um remetente TCP como a ocorrência de um esgotamento de temporização ou do recebimento de três ACKs duplicados do destinatário (lembre-se da nossa discussão, na Seção 3.5.4, do evento de temporização apresentado na Figura 3.33 e da subsequente modificação para incluir transmissão rápida quando do recebimento de três ACKs duplicados). Quando há congestionamento excessivo, então um (ou mais) buffers de roteadores ao longo do caminho transborda, fazendo com que um datagrama (contendo um segmento TCP) seja descartado. Esse datagrama descartado, por sua vez, resulta em um evento de perda no remetente — ou um esgotamento de temporização ou o recebimento de três ACKs duplicados — que é tomado por ele como uma indicação de congestionamento no caminho remetente-destinatário.

Já consideramos como é detectado o congestionamento; agora vamos considerar o caso mais otimista de uma rede livre de congestionamento, isto é, quando não ocorre um evento de perda. Nesse caso, o TCP remetente receberá reconhecimentos para segmentos anteriormente não reconhecidos. Como veremos, o TCP considerará a chegada desses reconhecimentos como uma indicação de que tudo está bem — os segmentos que estão sendo transmitidos para a rede estão sendo entregues com sucesso no destinatário — e usará reconhecimentos para aumentar o tamanho de sua janela de congestionamento (e, por conseguinte, sua taxa de transmissão). Note que, se os reconhecimentos chegarem ao remetente a uma taxa relativamente baixa (por exemplo, se o atraso no caminho fim a fim for alto ou se nele houver um enlace de baixa largura de banda), então a janela de congestionamento será aumentada a uma taxa relativamente baixa. Por outro lado, se os reconhecimentos chegarem a uma taxa alta, então a janela de congestionamento será aumentada mais rapidamente. Como o TCP utiliza reconhecimentos para acionar (ou regular) o aumento de tamanho de sua janela de congestionamento, diz-se que o TCP é **autorregulado**.

Dado o mecanismo de ajustar o valor de cwnd para controlar a taxa de envio, permanece a importante pergunta: Como um remetente TCP deve determinar a taxa a que deve enviar? Se os remetentes TCP enviam coletivamente muito rápido, eles podem congestionar a rede, levando ao tipo de congestionamento que vimos na Figura 3.8. Realmente, a versão de TCP que vamos estudar brevemente foi desenvolvida em resposta aos congestionamentos da Internet observados [Jacobson, 1988] sob versões anteriores do TCP. Entretanto, se os remetentes forem muito cautelosos e enviarem lentamente, eles podem subutilizar a largura de banda na rede; ou seja, os remetentes TCP podem enviar a uma taxa mais alta sem congestionar a rede. Então, como os remetentes TCP determinam suas taxas de envio de um modo que não congestionem a rede mas, ao mesmo tempo, façam uso de toda a largura de banda? Os remetentes TCP são claramente coordenados, ou existe uma abordagem distribuída na qual eles podem ajustar suas taxas de envio baseando-se somente nas informações locais? O TCP responde a essas perguntas utilizando os seguintes princípios:

Um segmento perdido implica em congestionamento, portanto, a taxa do remetente TCP deve diminuir quando um segmento é perdido. Lembre-se da nossa discussão na Seção 3.5.4, que um evento de esgotamento do temporizador ou o recebimento de quatro reconhecimentos para um certo segmento (um ACK original e, depois, três ACKs duplicados) é interpretado como uma indicação de “evento de perda” absoluto do segmento subsequente ao segmento ACK quadruplicado, acionando uma retransmissão do segmento perdido. De um ponto de vista do controle de congestionamento, a pergunta é como o remetente TCP deve diminuir sua janela de congestionamento e, portanto, sua taxa de envio, em resposta ao suposto evento de perda.

Um segmento reconhecido indica que a rede está enviando os segmentos do remetente ao destinatário e, por isso, a taxa do remetente pode aumentar quando um ACK chegar para um segmento anteriormente não reconhecido. A chegada de reconhecimentos é tida como uma indicação absoluta de que tudo está bem — os segmentos estão sendo enviados com sucesso do remetente ao destinatário, fazendo, assim, com que a rede não fique congestionada. Dessa forma, o tamanho da janela de congestionamento pode ser elevado.

Busca por largura de banda. Dado os ACKs que indicam um percurso de fonte a destino sem congestionamento, e eventos de perda que indicam um percurso congestionado, a estratégia do TCP de ajustar sua taxa de transmissão é aumentar sua taxa em resposta aos ACKs que chegam até que ocorra um evento de perda, momento em que a taxa de transmissão diminui. Desse modo, o remetente TCP aumenta sua taxa de transmissão para buscar a taxa a qual o congestionamento se inicia, recua dessa taxa e novamente faz a busca para ver se a taxa de início do congestionamento foi alterada. O comportamento do remetente TCP é análogo a uma criança que pede (e ganha) cada vez mais doces até finalmente receber um “Não！”, recuar, mas começar a pedir novamente pouco tempo depois. Observe que não há nenhuma sinalização explícita de congestionamento pela rede — os ACKs e eventos de perda servem como sinais implícitos — e que cada remetente TCP atua em informações locais em momentos diferentes de outros remetentes TCP.

Após essa visão geral sobre controle de congestionamento no TCP, agora podemos considerar os detalhes renomado algoritmo de controle de congestionamento no TCP, sendo primeiramente descrito em [Jacobson, 1988] e padronizado em [RFC 2581]. O algoritmo possui três componentes principais: (1) partida lenta, (2) contenção de congestionamento e (3) recuperação rápida. A partida lenta e a contenção de congestionamento são componentes obrigatórios do TCP, diferenciando em como eles aumentam o tamanho do cwnd em resposta a ACKs recebidos. Abordaremos em poucas palavras que a partida lenta aumenta o tamanho do cwnd de forma mais rápida (apesar do nome!) do que a contenção de congestionamento. A recuperação rápida é recomendada, mas não exigida, para remetentes TCP.

Partida lenta

Quando uma conexão TCP começa, o valor de cwnd normalmente é inicializado em 1 MSS [RFC 3390], resultando em uma taxa inicial de envio de aproximadamente MSS/RTT. Como exemplo, se MSS = 500 bytes e RTT = 200 milissegundos, então a taxa de envio inicial resultante é aproximadamente 20 kbps apenas. Como a largura de banda disponível para a conexão pode ser muito maior do que MSS/RTT, o remetente TCP ia gostar

de aumentar a quantia de largura de banda rapidamente. Dessa forma, no estado de partida lenta, o valor de cwnd começa em 1 MSS e aumenta 1 MSS toda vez que um segmento transmitido é reconhecido. No exemplo da Figura 3.51, o TCP envia o primeiro segmento para a rede e aguarda um reconhecimento. Quando o reconhecimento chega, o remetente TCP aumenta a janela de congestionamento para 1 MSS e envia dois segmentos de tamanho máximo. Esses segmentos são reconhecidos, e o remetente aumenta a janela de congestionamento para 1 MSS para cada reconhecimento de segmento, fornecendo uma janela de congestionamento de 4 MSS e assim por diante. Esse processo resulta em uma multiplicação da taxa de envio a cada RTT. Assim, a taxa de envio TCP se inicia lenta, mas cresce exponencialmente durante a fase de partida lenta.

Mas em que momento esse crescimento exponencial termina? A partida lenta apresenta diversas respostas para essa pergunta. Primeiro, se houver um evento de perda (ou seja, um congestionamento) indicado por um esgotamento de temporização, o remetente TCP estabelece o valor de cwnd para 1 e inicia o processo de partida lenta novamente. Ele também estabelece o valor de uma segunda variável de estado, ssthresh (abreviação de “slow start threshold [limiar de partida lenta]”) para $cwnd /2$ — metade do valor da janela de congestionamento quando este foi detectado. O segundo modo pelo qual a partida lenta pode terminar é ligado diretamente ao valor de ssthresh. Visto que ssthresh é metade do valor de cwnd quando o congestionamento foi detectado pela última vez, pode ser uma atitude precipitada continuar duplicando cwnd ao atingir ou ultrapassar o valor de ssthresh. Assim, quando o valor de cwnd se igualar ao de ssthresh, a partida lenta termina e o TCP é alterado para o modo de prevenção de congestionamento. Como veremos, o TCP aumenta cwnd com mais cautela quando está no modo de prevenção de congestionamento. O último modo pelo qual a partida lenta pode terminar é se três ACKs duplicados forem detectados, caso no qual o TPC apresenta uma retransmissão rápida (veja Seção 3.5.4) e entra no estado de recuperação rápida, como discutido abaixo. O comportamento do TCP na partida lenta está resumido na descrição FSM do controle de congestionamento no TCP na Figura 3.52. O algoritmo de partida lenta foi inicialmente proposto por [Jacobson, 1998]; uma abordagem semelhante à partida lenta também foi proposta de maneira independente em [Jain, 1986].

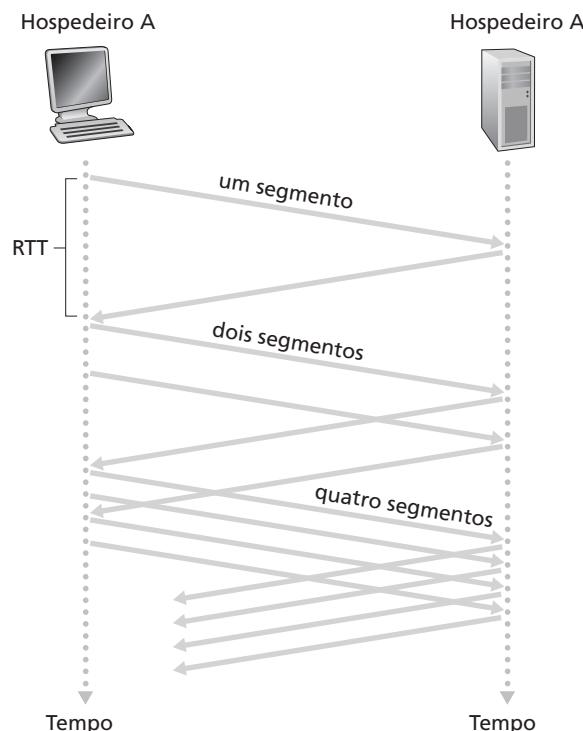


Figura 3.51 Partida lenta TCP

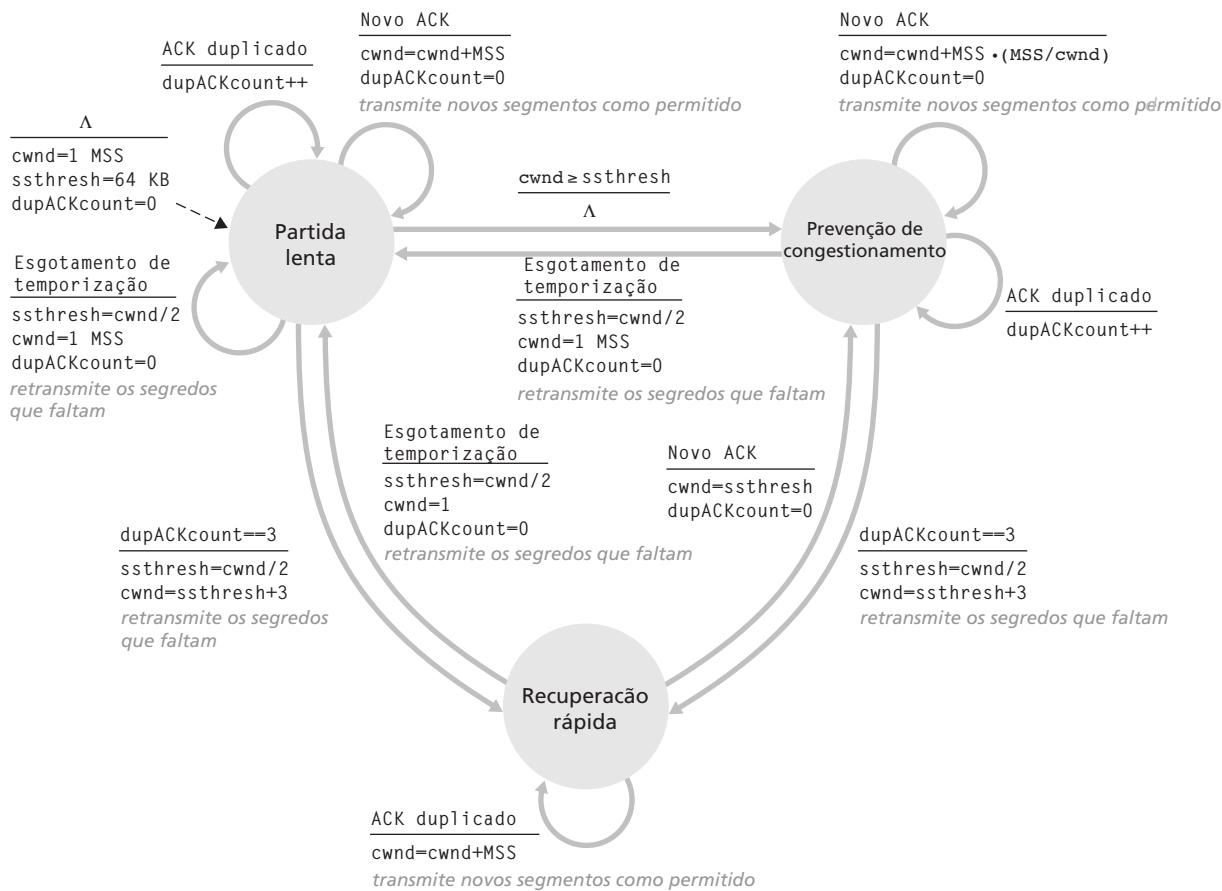


Figura 3.52 Descrição FSM do controle de congestionamento no TCP

Prevenção de congestionamento

Ao entrar no estado de prevenção de congestionamento, o valor de $cwnd$ é aproximadamente metade de seu valor quando o congestionamento foi encontrado pela última vez — o congestionamento poderia estar por perto! Desta forma, em vez de duplicar o valor de $cwnd$ a cada RTT, o TCP adota uma abordagem mais conservadora e aumenta o valor de $cwnd$ por meio de um único MSS a cada RTT [RFC 2581]. Isso pode ser realizado de diversas formas. Uma abordagem comum é o remetente aumentar $cwnd$ por MSS bytes ($MSS/cwnd$) no momento em que um novo reconhecimento chegar. Por exemplo, se o MSS possui 1.460 bytes e $cwnd$, 14.600 bytes, então 10 segmentos estão sendo enviados dentro de um RTT. Cada ACK que chega (considerando um ACK por segmento) aumenta o tamanho da janela de congestionamento em 1/10 MSS e, assim, o valor da janela de congestionamento terá aumentado para um MSS após os ACKs quando todos os segmentos tiverem sido recebidos.

Mas em que momento o aumento linear da prevenção de congestionamento (de 1 MSS por RTT) deve terminar? O algoritmo de prevenção de congestionamento TCP se comporta da mesma forma quando ocorre um esgotamento de temporização. Como no caso da partida lenta: o valor de $cwnd$ é ajustado para 1 MSS, e o valor de $ssthresh$ é atualizado para metade do valor de $cwnd$ quando ocorreu o evento de perda. Lembre-se, entretanto, de que um evento de perda também pode ser acionado por um evento ACK duplicado triplo. Neste caso, a rede continua a enviar segmentos do remetente ao destinatário (como indicado pelo recebimento de ACKs duplicados). Portanto, o comportamento do TCP para esse tipo de evento de perda deve ser menos drástico do que com uma perda de esgotamento de temporização: O TCP reduz o valor de $cwnd$ para metade (adicionando em 3 MSS a mais para contabilizar os ACKs duplicados triplos recebidos) e registra o valor de

`ssthresh` como metade do de `cwnd` quando os ACKs duplicados triplos foram recebidos. Então, entra-se no estado de recuperação rápida.

Recuperação rápida

Na recuperação rápida, o valor de `cwnd` é aumentado por 1 MSS para cada ACK duplicado recebido no segmento perdido que fez com que o TCP entrasse no modo de recuperação rápida. Consequentemente, quando um ACK chega ao segmento perdido, o TCP entra no modo de prevenção de congestionamento após reduzir `cwnd`. Se um evento de esgotamento de temporização ocorrer, a recuperação rápida é alterada para o modo partida lenta após desempenhar as mesmas ações que a partida lenta e a prevenção de congestionamento: o valor de `cwnd` é ajustado para 1 MSS, e o valor de `ssthresh`, para metade do valor de `cwnd` no momento em que o evento de perda ocorreu.

A recuperação rápida é recomendada, mas não exigida, para o protocolo TCP [RFC 2581]. É interessante o fato de que uma antiga versão do TCP, conhecida como TCP Tahoe, reduz incondicionalmente sua janela de congestionamento para 1 MSS e entrou na fase de partida lenta após um evento de perda de esgotamento do temporizador ou de ACK duplicado triplo. A versão atual do TCP, a TCP Reno, incluiu a recuperação rápida.

A Figura 3.53 ilustra a evolução da janela de congestionamento do TCP para as versões Reno e Tahoe. Nessa figura, o limiar é, inicialmente, igual a 8 MSS. Nas primeiras oito sessões de transmissão, as duas versões possuem ações idênticas. A janela de congestionamento se eleva exponencialmente rápido durante a partida lenta e atinge o limiar na quarta sessão de transmissão. A janela de congestionamento, então, se eleva linearmente até que ocorra um evento ACK duplicado triplo, logo após a oitava sessão de transmissão. Observe que a janela de congestionamento é $12 \cdot \text{MSS}$ quando ocorre o evento de perda. O valor de `ssthresh` é, então, ajustado para $0,5 \cdot \text{cwnd} = 6 \cdot \text{MSS}$. Sob o TCP Reno, a janela de congestionamento é ajustada para `cwnd = 6 \cdot \text{MSS}`, e depois cresce linearmente.

A Figura 3.52 apresenta a descrição FSM completa dos algoritmos de controle de congestionamento — partida lenta, prevenção de congestionamento e recuperação rápida. A figura também indica onde pode ocorrer transmissão de novos segmentos ou segmentos retransmitidos. Embora seja importante diferenciar controle/retransmissão de erro TCP de controle de congestionamento no TCP, também é importante avaliar como esses dois aspectos do TCP estão inseparavelmente ligados.

Controle de congestionamento no TCP: retrospectiva

Após nos aprofundarmos em detalhes sobre partida lenta, prevenção de congestionamento e recuperação rápida, vale a pena agora voltar e ver a floresta por entre as árvores. Desconsiderando o período inicial de partida lenta, quando uma conexão se inicia, e supondo que as perdas são indicadas por ACKs duplicados triplos e não por esgotamentos de temporização, o controle de congestionamento no TCP consiste em um aumento linear

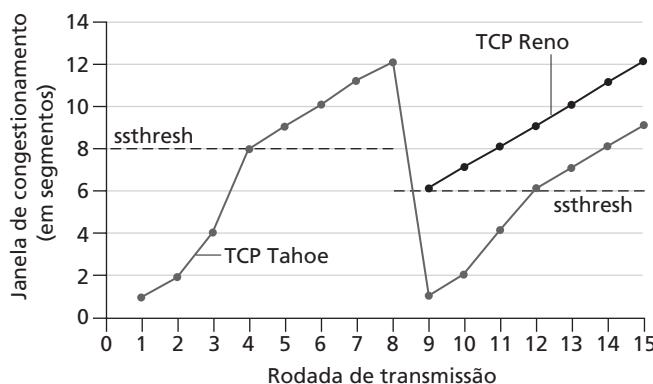


Figura 3.53 Evolução da janela de congestionamento do TCP (Tahoe e Reno)

(aditivo) em $cwnd$ de 1 MMS por RTT e, então, uma redução à metade (diminuição multiplicativa) de $cwnd$ em um evento ACK duplicado triplo. Por esta razão, o controle de congestionamento no TCP é frequentemente denominado aumento aditivo, diminuição multiplicativa (AIMD). O controle de congestionamento AIMD faz surgir o comportamento semelhante a “dentes de serra”, mostrado na Figura 3.54, a qual também ilustra de forma interessante nossa intuição anterior sobre a “busca” do TCP por largura de banda — o TCP aumenta linearmente o tamanho de sua janela de congestionamento (e, portanto, sua taxa de transmissão) até que um evento ACK duplicado triplo ocorra. Então, ele reduz o tamanho de sua janela por um fator de dois mas começa novamente a aumentá-la linearmente, buscando saber se há uma largura de banda adicional disponível.

Como observado anteriormente, a maioria das implementações TCP, atualmente, utiliza o algoritmo Reno [Padhye, 2001]. Foram propostas muitas variações do algoritmo Reno [RFC 3782; RFC 2018]. O algoritmo Vegas [Brakmo, 1995; Ahn, 1995] tenta evitar o congestionamento enquanto mantém uma boa vazão. A ideia básica de tal algoritmo é (1) detectar congestionamento nos roteadores entre a fonte e o destino *antes* que ocorra a perda de pacote e (2) diminuir linearmente a taxa ao ser detectada essa perda de pacote iminente, que pode ser prevista por meio da observação do RTT. Quanto maior for o RTT dos pacotes, maior será o congestionamento nos roteadores. O Linux suporta um número de algoritmos de controle de congestionamento (incluindo o TCP Reno e o TCP Vegas) e permite que um administrador de sistema configure qual versão do TCP será utilizada. A versão padrão do TCP no Linux versão 2.6.18 foi definida para CUBIC [Ha, 2008], uma versão do TCP desenvolvida para aplicações de alta largura de banda. O algoritmo AIMD do TCP foi desenvolvido com base na quantidade significativa de percepção de engenharia e experimentação com controle de congestionamento em redes operacionais. Dez anos após a criação do TCP, análises teóricas mostraram que o algoritmo de controle de congestionamento do TCP serve como um algoritmo de otimização assíncrona distribuída que resulta em diversos aspectos importantes sobre usuário e desempenho da rede sendo otimizados simultaneamente [Kelly, 1998]. Uma teoria aprimorada sobre controle de congestionamento tem sido desenvolvida desde então [Srikant, 2004].

Descrição macroscópica da dinâmica do TCP

Dado o comportamento semelhante a dentes de serra do TCP, é natural considerar qual seria a vazão média (isto é, a taxa média) de uma conexão TCP ativa há longo tempo. Nessa análise, vamos ignorar as fases de partida lenta que ocorrem após eventos de esgotamento de temporização. (Essas fases normalmente são muito curtas, visto que o remetente sai delas com rapidez exponencial.) Durante um determinado intervalo de viagem de ida e volta, a taxa à qual o TCP envia dados é uma função da janela de congestionamento e do RTT corrente. Quando o tamanho da janela for w bytes, e o tempo de viagem de ida e volta corrente for RTT segundos, a taxa de transmissão do TCP será aproximadamente w/RTT . Então, o TCP faz uma sondagem em busca de alguma largura de

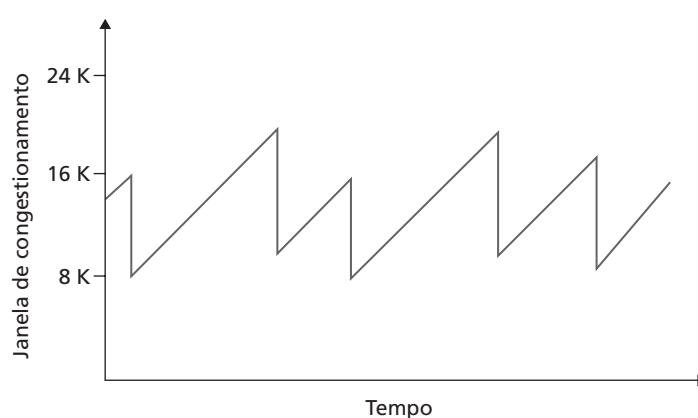


Figura 3.54 Controle de congestionamento por aumento aditivo, diminuição multiplicativa

banda adicional aumentando w de 1 MSS a cada RTT até ocorrer um evento de perda. Seja W o valor de w quando ocorre um evento de perda. Admitindo que RTT e W são aproximadamente constantes durante o período da conexão, a taxa de transmissão fica na faixa de $W/(2 \cdot RTT)$ a W/RTT .

Essas suposições levam a um modelo macroscópico muito simplificado para o comportamento do TCP em regime. A rede descarta um pacote da conexão quando a taxa aumenta para W/RTT ; então a taxa é reduzida à metade e, em seguida, aumenta de um MSS/RTT a cada RTT até alcançar W/RTT novamente. Esse processo se repete continuamente. Como a vazão do TCP (isto é, a taxa) aumenta linearmente entre os dois valores extremos, temos:

$$\text{Vazão média de uma conexão} = \frac{0,75 \cdot W}{RTT}$$

Usando esse modelo muito idealizado para a dinâmica de regime permanente do TCP, podemos também derivar uma interessante expressão que relaciona a taxa de perda de uma conexão com sua largura de banda disponível [Mahdavi, 1997]. Essa derivação está delineada nos exercícios de fixação. Um modelo mais sofisticado que demonstrou empiricamente estar de acordo com dados medidos é apresentado em [Padhye, 2000].

TCPs futuros

É importante perceber que o controle de congestionamento no TCP evoluiu ao longo dos anos e, na verdade, continua a evoluir. Um resumo do controle de congestionamento no TCP no final da década de 1990 pode ser encontrado no [RFC 2581]; se quiser uma discussão sobre desenvolvimentos recentes do controle de congestionamento no TCP, veja [Floyd, 2001]. O que era bom para a Internet quando a maior parte das conexões TCP carregava tráfego SMTP, FTP e Telnet não é necessariamente bom para a Internet de hoje, dominada pelo HTTP, ou para uma Internet futura, com serviços que ainda nem sonhamos.

A necessidade da evolução contínua do TCP pode ser ilustrada considerando as conexões TCP de alta velocidade que são necessárias para aplicações de computação em grade [Foster, 2002]. Por exemplo, considere uma conexão TCP com segmentos de 1.500 bytes e RTT de 100 ms, e suponha que queremos enviar dados por essa conexão a 10 Gbps. Seguindo o [RFC 3649] e utilizando a fórmula de vazão do TCP apresentada anteriormente, notamos que, para alcançar uma vazão de 10 Gbps, o tamanho médio da janela de congestionamento precisaria ser 83.333 segmentos. São muitos segmentos, e pensar que um deles poderia ser perdido em trânsito nos deixa bastante preocupados. O que aconteceria no caso de uma perda? Ou, em outras palavras, que fração dos segmentos transmitidos poderia ser perdida e ainda assim permitisse que o algoritmo de controle de congestionamento no TCP delineado na Tabela 3.52 alcançasse a desejada taxa de 10 Gbps? Nos exercícios de fixação deste capítulo você percorrerá a dedução de uma fórmula que relaciona a vazão de uma conexão TCP em função da taxa de perda (L), do tempo de viagem de ida e volta (RTT) e do tamanho máximo de segmento (MSS):

$$\text{Vazão média de uma conexão} = \frac{1,22 \cdot MSS}{RTT\sqrt{L}}$$

Usando essa fórmula, podemos ver que, para alcançar uma vazão de 10 Gbps, o algoritmo de controle de congestionamento no TCP de hoje pode tolerar uma probabilidade de perda de segmentos de apenas $2 \cdot 10^{-10}$ (equivalente a um evento de perda a cada 5 bilhões de segmentos) — uma taxa muito baixa. Essa observação levou muitos pesquisadores a investigar novas versões do TCP especificamente projetadas para esses ambientes de alta velocidade; [Jin, 2004; RFC 3649; Kelly, 2003; Ha, 2008] apresentam discussões sobre esses esforços.

3.7.1 Equidade

Considere K conexões TCP, cada uma com um caminho fim a fim diferente, mas todas passando pelo gargalo em um enlace com taxa de transmissão de R bps (aqui, *gargalo em um enlace* quer dizer que nenhum dos outros enlaces ao longo do caminho de cada conexão está congestionado e que todos dispõem de abundante capacidade de transmissão em comparação à capacidade de transmissão do enlace com gargalo). Suponha que cada conexão

está transferindo um arquivo grande e que não há tráfego UDP passando pelo enlace com gargalo. Dizemos que um mecanismo de controle de congestionamento é *justo* se a taxa média de transmissão de cada conexão for aproximadamente R/K ; isto é, cada uma obtém uma parcela igual da largura de banda do enlace.

O algoritmo AIMD do TCP é justo, considerando, em especial, que diferentes conexões TCP podem começar em momentos diferentes e, assim, ter tamanhos de janela diferentes em um dado instante? [Chiu, 1989] explica, de um modo elegante e intuitivo, por que o controle de congestionamento converge para fornecer um compartilhamento justo da largura de banda do enlace entre conexões TCP concorrentes.

Vamos considerar o caso simples de duas conexões TCP compartilhando um único enlace com taxa de transmissão R , como mostra a Figura 3.55. Vamos admitir que as duas conexões tenham os mesmos MSS e RTT (de modo que, se o tamanho de suas janelas de congestionamento for o mesmo, então terão a mesma vazão) e uma grande quantidade de dados para enviar e que nenhuma outra conexão TCP ou datagramas UDP atravesse esse enlace compartilhado. Vamos ignorar também a fase de partida lenta do TCP e admitir que as conexões TCP estão operando em modo prevenção de congestionamento (AIMD) todo o tempo.

A Figura 3.56 mostra a vazão alcançada pelas duas conexões TCP. Se for para o TCP compartilhar equitativamente a largura de banda do enlace entre as duas conexões, então a vazão alcançada deverá cair ao longo da linha a 45 graus (igual compartilhamento da largura de banda) que parte da origem. Idealmente, a soma das duas vazões seria igual a R . (Com certeza, não é uma situação desejável cada conexão receber um compartilhamento igual, mas igual a zero, da capacidade do enlace!) Portanto, o objetivo é que as vazões alcançadas fiquem em algum lugar perto da intersecção da linha de igual compartilhamento da largura de banda com a linha de utilização total da largura de banda da Figura 3.56.

Suponha que os tamanhos de janela TCP sejam tais que, em um determinado instante, as conexões 1 e 2 alcancem as vazões indicadas pelo ponto A na Figura 3.56. Como a quantidade de largura de banda do enlace consumida conjuntamente pelas duas conexões é menor do que R , não ocorrerá nenhuma perda e ambas as conexões aumentarão suas janelas de 1 por RTT como resultado do algoritmo de prevenção de congestionamento do TCP. Assim, a vazão conjunta das duas conexões continua ao longo da linha a 45 graus (aumento igual para as duas), começando no ponto A. Finalmente, a largura de banda do enlace consumida em conjunto pelas duas conexões será maior do que R e, assim, também fatalmente, ocorrerá perda de pacote. Suponha que as conexões 1 e 2 experimentem perda de pacote quando alcançarem as vazões indicadas pelo ponto B. As conexões 1 e 2 então reduzirão suas janelas por um fator de 2. Assim, as vazões resultantes são as do ponto C, a meio caminho do vetor que começa em B e termina na origem. Como a utilização conjunta da largura de banda é menor do que R no ponto C, as duas conexões novamente aumentam suas vazões ao longo da linha a 45 graus que começa no ponto C. Mais cedo ou mais tarde ocorrerá perda, por exemplo, no ponto D, e as duas conexões novamente reduzirão o tamanho de suas janelas por um fator de 2 — e assim por diante. Você pode ter certeza de que a largura de banda alcançada pelas duas conexões flutuará ao longo da linha de igual compartilhamento da largura de banda. E também estar certo de que as duas conexões convergirão para esse comportamento, não importando onde elas começem no espaço bidimensional! Embora haja uma série de suposições idealizadas por trás desse cenário, ainda assim ele dá uma ideia intuitiva de por que o TCP resulta em igual compartilhamento da largura de banda entre conexões.

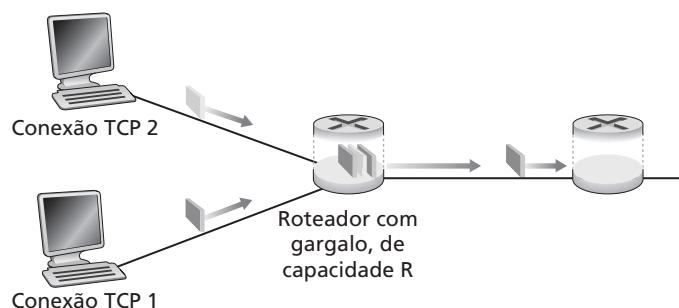


Figura 3.55 Duas conexões TCP compartilhando um único enlace congestionado

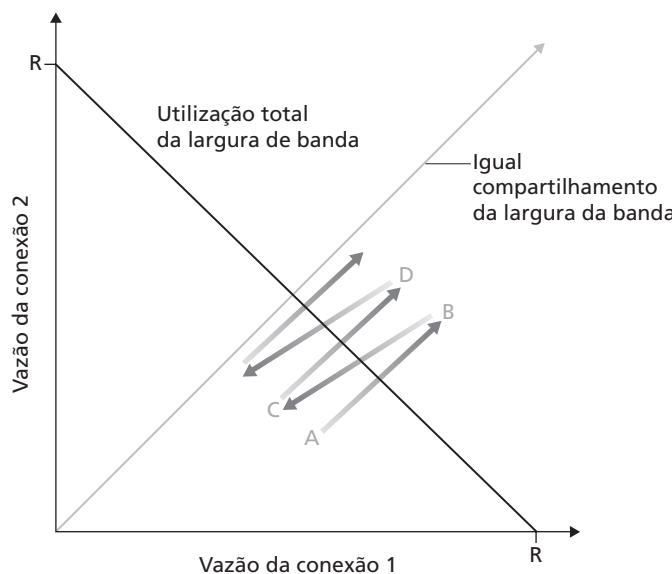


Figura 3.56 Vazão alcançada pelas conexões TCP 1 e TCP 2

Em nosso cenário idealizado, admitimos que somente conexões TCP atravessem o enlace com gargalo, que as conexões tenham o mesmo valor de RTT e que uma única conexão TCP esteja associada com um par hospedeiro/destinatário. Na prática, essas condições não são comumente encontradas e, assim, é possível que as aplicações cliente-servidor obtenham porções muito desiguais da largura de banda do enlace. Em especial, foi demonstrado que, quando várias conexões compartilham um único enlace com gargalo, as sessões cujos RTTs são menores conseguem obter a largura de banda disponível naquele enlace mais rapidamente (isto é, abre suas janelas de congestionamento mais rapidamente) à medida que o enlace fica livre. Assim, conseguem vazões mais altas do que conexões com RTTs maiores [Lakshman, 1997].

Equidade e UDP

Acabamos de ver como o controle de congestionamento no TCP regula a taxa de transmissão de uma aplicação por meio do mecanismo de janela de congestionamento. Muitas aplicações de multimídia, como telefone por Internet e videoconferência, muitas vezes não rodam sobre TCP exatamente por essa razão — elas não querem que sua taxa de transmissão seja limitada, mesmo que a rede esteja muito congestionada. Ao contrário, preferem rodar sobre UDP, que não tem controle de congestionamento. Quando rodam sobre esse protocolo, as aplicações podem passar seus áudios e vídeos para a rede a uma taxa constante e, ocasionalmente, perder pacotes, em vez de reduzir suas taxas a níveis ‘justos’ em horários de congestionamento e não perder nenhum deles. Da perspectiva do TCP, as aplicações de multimídia que rodam sobre UDP não são justas — elas não cooperam com as outras conexões nem ajustam suas taxas de transmissão de maneira adequada. Como o controle de congestionamento no TCP reduzirá sua taxa de transmissão quando houver aumento de congestionamento (perda), enquanto fontes UDP não precisam fazer o mesmo, é possível que essas fontes desalojem o tráfego TCP. Uma área importante da pesquisa hoje é o desenvolvimento de mecanismos de controle de congestionamento que impeçam que o tráfego de UDP leve a vazão da Internet a uma parada repentina [Floyd, 1999; Floyd, 2000; Kohler, 2006].

Equidade e conexões TCP paralelas

Mas, mesmo que pudéssemos obrigar o tráfego UDP a se comportar com equidade, o problema ainda não estaria completamente resolvido. Isso porque não há nada que impeça uma aplicação de rodar sobre TCP usando múltiplas conexões paralelas. Por exemplo, browsers Web frequentemente usam múltiplas conexões

TCP paralelas para transferir os vários objetos de uma página Web. (O número exato de conexões múltiplas pode ser configurado na maioria dos browsers.) Quando usa múltiplas conexões paralelas, uma aplicação consegue uma fração maior da largura de banda de um enlace congestionado. Como exemplo, considere um enlace de taxa R que está suportando nove aplicações cliente-servidor em curso, e cada uma das aplicações está usando uma conexão TCP. Se surgir uma nova aplicação que também utilize uma conexão TCP, então cada aplicação conseguirá aproximadamente a mesma taxa de transmissão igual a $R/10$. Porém, se, em vez disso, essa nova aplicação usar 11 conexões TCP paralelas, então ela conseguirá uma alocação injusta de mais do que $R/2$. Como a penetração do tráfego Web na Internet é grande, as múltiplas conexões paralelas não são incomuns.

3.8 Resumo

Começamos este capítulo estudando os serviços que um protocolo de camada de transporte pode prover às aplicações de rede. Por um lado, o protocolo de camada de transporte pode ser muito simples e oferecer serviços básicos às aplicações, provendo apenas uma função de multiplexação/demultiplexação para processos comunicantes. O protocolo UDP da Internet é um exemplo desse serviço básico de um protocolo de camada de transporte. Por outro lado, um protocolo de camada de transporte pode fornecer uma variedade de garantias às aplicações, como entrega confiável de dados, garantias contra atrasos e garantias de largura de banda. Não obstante, os serviços que um protocolo de transporte pode prover são frequentemente limitados pelo modelo de serviço do protocolo subjacente de camada de rede. Se o protocolo de camada de rede não puder proporcionar garantias contra atraso ou garantias de largura de banda para segmentos da camada de transporte, então o protocolo de camada de transporte não poderá fornecer essas garantias para as mensagens enviadas entre processos.

Aprendemos na Seção 3.4 que um protocolo de camada de transporte pode prover transferência confiável de dados mesmo que a camada de rede subjacente seja não confiável. Vimos que há muitos pontos sutis na transferência confiável de dados, mas que a tarefa pode ser realizada pela combinação cuidadosa de reconhecimentos, temporizadores, retransmissões e números de sequência.

Embora tenhamos examinado a transferência confiável de dados neste capítulo, devemos ter em mente que essa transferência pode ser fornecida por protocolos de camada de enlace, de rede, de transporte ou de aplicação. Qualquer uma das camadas superiores da pilha de protocolos pode implementar reconhecimentos, temporizadores, retransmissões e números de sequência e prover transferência confiável de dados para a camada situada acima dela. Na verdade, com o passar dos anos, engenheiros e cientistas da computação projetaram e implementaram, independentemente, protocolos de camada de enlace, de rede, de transporte e de aplicação que fornecem transferência confiável de dados (embora muitos desses protocolos tenham desaparecido silenciosamente).

Na Seção 3.5, examinamos detalhadamente o TCP, o protocolo de camada de transporte confiável orientado para conexão da Internet. Aprendemos que o TCP é complexo e envolve a conexão, controle de fluxo, estimativa de tempo de viagem de ida e volta, bem como transferência confiável de dados. Na verdade, o TCP é mais complexo do que nossa descrição — intencionalmente, não discutimos uma variedade de ajustes, acertos e melhorias que estão implementados em várias versões do TCP. Toda essa complexidade, no entanto, fica escondida da aplicação de rede. Se um cliente em um hospedeiro quiser enviar dados de maneira confiável para outro hospedeiro, ele simplesmente abre uma porta TCP para o servidor e passa dados para dentro dessa porta. A aplicação cliente-servidor felizmente fica alheia a toda a complexidade do TCP.

Na Seção 3.6, examinamos o controle de congestionamento de uma perspectiva mais ampla e, na Seção 3.7, demonstramos como o TCP implementa controle de congestionamento. Aprendemos que o controle de congestionamento é imperativo para o bem-estar da rede. Sem ele, uma rede pode facilmente ficar travada, com pouco ou nenhum dado sendo transportado fim a fim. Na Seção 3.7, aprendemos também que o TCP implementa um mecanismo de controle de congestionamento fim a fim que aumenta aditivamente sua taxa de transmissão quando a rede se congestionar.

do julga que o caminho da conexão TCP está livre de congestionamento e reduz multiplicativamente sua taxa de transmissão quando ocorre perda. Esse mecanismo também luta para dar a cada conexão TCP que passa por um enlace congestionado uma parcela igual da largura de banda da conexão. Examinamos ainda com um certo detalhe o impacto do estabelecimento da conexão TCP e da partida lenta sobre a latência. Observamos que, em muitos cenários importantes, o estabelecimento da conexão e a partida lenta contribuem de modo significativo para o atraso fim a fim. Enfatizamos mais uma vez que, embora tenha evoluído com o passar dos anos, o controle de congestionamento no TCP permanece como uma área de pesquisa intensa e, provavelmente, continuará a evoluir nos anos vindouros.

Nossa discussão sobre os protocolos específicos de transporte da Internet neste capítulo se focalizou no UDP e no TCP — os dois “burros de carga” da camada de transporte da Internet. Entretanto, duas décadas de experiência com esses dois protocolos identificaram circunstâncias nas quais nenhum deles é apropriado de maneira ideal. Desse modo, pesquisadores se ocuparam em desenvolver protocolos da camada de transporte adicionais, muitos dos quais são, agora, padrões propostos pelo IETF.

O Protocolo de Controle de Congestionamento de Datagrama (DCCP) [RFC4340] oferece um serviço de baixo consumo, orientado a mensagem e não confiável de forma semelhante ao UDP, mas com uma forma selecionada de aplicação de controle de congestionamento que é compatível com o TCP. Caso uma aplicação necessitar de uma transferência de dados confiável ou semiconfiável, então isso seria realizado dentro da própria aplicação, talvez utilizando os mecanismos que estudamos na Seção 3.4. O DCCP é utilizado em aplicações, como o fluxo de mídia (veja Capítulo 7), que podem se aproveitar da escolha entre conveniência e confiança do fornecimento de dados, mas que querem ser sensíveis ao congestionamento da rede.

O Protocolo de Controle de Fluxo de Transmissão (SCTP) [RFC 2960, RFC 3286] é um protocolo confiável e orientado a mensagens que permite que diferentes “fluxos” de aplicação sejam multiplexados através de uma única conexão SCTP (método conhecido como “múltiplos fluxos”). De um ponto de vista confiável, os diferentes fluxos dentro da conexão são controlados separadamente, de modo que uma perda de pacote em um fluxo não afete o fornecimento de dados em outros fluxos. O SCTP também permite que os dados sejam transferidos por meio de dois percursos de saída quando um hospedeiro está conectado a duas ou mais redes, fornecimento opcional de dados inadequados, e muitos outros recursos. Os algoritmos de controle de congestionamento e de fluxo do SCTP são basicamente os mesmos do TCP.

O protocolo TRFC (*TCP — Friendly Rate Control*) [RFC 5348] é mais um protocolo de controle de congestionamento do que um protocolo da camada de transporte completo. Ele especifica um mecanismo de controle de congestionamento que poderia ser utilizado em outro protocolo de transporte como o DCCP (de fato, um dos dois protocolos de aplicação disponível no DCCP é o TRFC). O objetivo do TRFC é estabilizar o comportamento dos “dentes de serra” (veja Figura 3.54) no controle de congestionamento no TCP, enquanto mantém uma taxa de envio de longo prazo “razoavelmente” próxima à do TCP. Com uma taxa de envio mais estável do que a do TCP, o TRFC é adequado às aplicações multimídia, como telefonia IP ou fluxo de mídia, em que uma taxa estável é importante. O TRFC é um protocolo baseado em equação que utiliza a taxa de perda de pacote calculada como suporte a uma equação [Padhye, 2000] que estima qual seria a vazão do TCP se uma sessão TCP sofrer taxa de perda. Essa taxa é então adotada como a taxa-alvo de envio do TRFC.

Somente o futuro dirá se o DCCP, o SCTP ou o TRFC possuirá uma implementação difundida. Enquanto esses protocolos fornecem claramente recursos avançados sobre o TCP e o UDP, estes já provaram a si mesmos ser “bons o bastante” com o passar dos anos. A vitória do “melhor” sobre o “bom o bastante” dependerá de uma mistura complexa de considerações técnicas, sociais e comerciais.

No Capítulo 1, dissemos que uma rede de computadores pode ser dividida em ‘periferia da rede’ e ‘núcleo da rede’. A periferia da rede abrange tudo o que acontece nos sistemas finais. Com o exame da camada de aplicação e da camada de transporte, nossa discussão sobre a periferia da rede está completa. É hora de explorar o núcleo da rede! Essa jornada começa no próximo capítulo, em que estudaremos a camada de rede, e continua no Capítulo 5, em que estudaremos a camada de enlace.



Exercícios de fixação

Capítulo 3 Questões de revisão

Seções 3.1 a 3.3

1. Suponha que uma camada de rede forneça o seguinte serviço. A camada de rede no computador-fonte aceita um segmento de tamanho máximo de 1.200 bytes e um endereço de computador-alvo da camada de transporte. Esta, então, garante encaminhar o segmento para a camada de transporte no computador-alvo. Suponha que muitos processos de aplicação de rede possam estar sendo executados no computador-alvo.
 - a. Crie, da forma mais simples, o protocolo da camada de transporte possível que levará os dados da aplicação para o processo desejado no computador-alvo. Suponha que o sistema operacional do computador-alvo determinou um número de porta de 4 bytes para cada processo de aplicação em execução.
 - b. Modifique esse protocolo de modo que ele forneça um “endereço de retorno” para o processo-alvo.
 - c. Em seus protocolos, a camada de transporte “tem de fazer algo” no núcleo da rede de computadores?
2. Considere um planeta onde todos possuam uma família com seis membros, cada família viva em sua própria casa, cada casa possua um endereço único e cada pessoa em certa casa possua um único nome. Suponha que esse planeta possua um serviço postal que entregue cartas da casa-fonte à casa-alvo. O serviço exige que (i) a carta esteja em um envelope e que (ii) o endereço da casa-alvo (e nada mais) esteja escrito claramente no envelope. Suponha que cada família possua um membro representante que recebe e distribui cartas para outros membros da família. As cartas não apresentam necessariamente qualquer indicação dos destinatários das cartas.
 - a. Utilizando a solução do Problema 1 como inspiração, descreva um protocolo que os representantes possam utilizar para entregar cartas de um membro remetente de uma família para um membro destinatário de outra família.
 - b. Em seu protocolo, o serviço postal precisa abrir o envelope e verificar a carta para fornecer o serviço?
3. Considere uma conexão TCP entre o hospedeiro A e o hospedeiro B. Suponha que os segmentos TCP que trafegam do hospedeiro A para o hospedeiro B tenham número de porta da fonte x e

número de porta do destino y . Quais são os números de porta da fonte e do destino para os segmentos que trafegam do hospedeiro B para o hospedeiro A?

4. Descreva por que um desenvolvedor de aplicação pode escolher rodar uma aplicação sobre UDP em vez de sobre TCP.
5. Por que o tráfego de voz e de vídeo é frequentemente enviado por meio do UDP e não do TCP na Internet de hoje? (Dica: A resposta que procuramos não tem nenhuma relação com o mecanismo de controle de congestionamento no TCP.)
6. É possível que uma aplicação desfrute de transferência confiável de dados mesmo quando roda sobre UDP? Caso a resposta seja afirmativa, como isso acontece?
7. Suponha que um processo no Computador C possua um socket UDP com número de porta 6789 e que o Computador A e o Computador B, individualmente, enviem um segmento UDP ao Computador C com número de porta de destino 6789. Esses dois segmentos serão encaminhados para o mesmo socket no Computador C? Se sim, como o processo no Computador C saberá que esses dois segmentos vieram de dois computadores diferentes?
8. Suponha que um servidor da Web seja executado no Computador C na porta 80. Esse servidor utiliza conexões contínuas e, no momento, está recebendo solicitações de dois Computadores diferentes, A e B. Todas as solicitações estão sendo enviadas através do mesmo socket no Computador C? Se eles estão passando por diferentes sockets, dois desses sockets possuem porta 80? Discuta e explique.

Seção 3.4

9. Nos nossos protocolos rdt, por que precisamos introduzir números de sequência?
10. Nos nossos protocolos rdt, por que precisamos introduzir temporizadores?
11. Suponha que o atraso de viagem de ida e volta entre o emissor e o receptor seja constante e conhecido para o emissor. Ainda seria necessário um temporizador no protocolo rdt 3.0, supondo que os pacotes podem ser perdidos? Explique.
12. Visite o applet Go-Back-N Java no Companion siteWeb.

- a. A fonte enviou cinco pacotes e depois interrompeu a animação antes que qualquer um dos cinco pacotes chegassem ao destino. Então, elimine o primeiro pacote e reinicie a animação. Descreva o que acontece.
 - b. Repita o experimento, mas agora deixe o primeiro pacote chegar ao destino e elimine o primeiro reconhecimento. Descreva novamente o que acontece.
 - c. Por fim, tente enviar seis pacotes. O que acontece?
13. Repita a questão 12, mas agora com o applet *Selective Repeat Java*. O que difere o *Selective Repeat* do Go-Back-N?

Seção 3.5

14. Verdadeiro ou falso:

- a. O hospedeiro A está enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Suponha que o hospedeiro B não tenha dados para enviar para o hospedeiro A. O hospedeiro B não enviará reconhecimentos para o hospedeiro A porque ele não pode dar carona aos reconhecimentos dos dados.
- b. O tamanho de *rwnd* do TCP nunca muda enquanto dura a conexão.
- c. Suponha que o hospedeiro A esteja enviando ao hospedeiro B um arquivo grande por uma conexão TCP. O número de bytes não reconhecidos que o hospedeiro A envia não pode exceder o tamanho do buffer de recepção.
- d. Imagine que o hospedeiro A esteja enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Se o número de sequência para um segmento dessa conexão for m , então o número de sequência para o segmento subsequente será necessariamente $m + 1$.
- e. O segmento TCP tem um campo em seu cabeçalho para *Rwnd*.
- f. Suponha que o último *SampleRTT* de uma conexão TCP seja igual a 1 segundo. Então, o

valor corrente de *TimeoutInterval* para a conexão será necessariamente ajustado para um valor ≥ 1 segundo.

- g. Imagine que o hospedeiro A envie ao hospedeiro B, por uma conexão TCP, um segmento com o número de sequência 38 e 4 bytes de dados. Nesse mesmo segmento, o número de reconhecimento será necessariamente 42.
15. Suponha que o hospedeiro A envie dois segmentos TCP um atrás do outro ao hospedeiro B sobre uma conexão TCP. O primeiro segmento tem número de sequência 90 e o segundo, número de sequência 110.
- a. Quantos dados tem o primeiro segmento?
 - b. Suponha que o primeiro segmento seja perdido, mas o segundo chegue a B. No reconhecimento que B envia a A, qual será o número de reconhecimento?
16. Considere o exemplo do Telnet discutido na Seção 3.5. Alguns segundos após o usuário digitar a letra 'C', ele digitará a letra 'R'. Depois disso, quantos segmentos serão enviados e o que será colocado nos campos de número de sequência e de reconhecimento dos segmentos?

Seção 3.7

17. Suponha que duas conexões TCP estejam presentes em algum enlace congestionado de velocidade R bps. Ambas as conexões têm um arquivo imenso para enviar (na mesma direção, pelo enlace congestionado). As transmissões dos arquivos começam exatamente ao mesmo tempo. Qual é a velocidade de transmissão que o TCP gostaria de dar a cada uma das conexões?
18. Verdadeiro ou falso: considere o controle de congestionamento no TCP. Quando um temporizador expira no remetente, o "valor de *ssthresh*" é ajustado para a metade de seu valor anterior.



Problemas

1. Suponha que o cliente A inicie uma sessão Telnet com o servidor S. Quase ao mesmo tempo, o cliente B também inicia uma sessão Telnet com o servidor S. Forneça possíveis números de porta da fonte e do destino para:
 - a. Os segmentos enviados de A para S.
 - b. Os segmentos enviados de B para S.
- c. Os segmentos enviados de S para A.
- d. Os segmentos enviados de S para B.
- e. Se A e B são hospedeiros diferentes, é possível que o número de porta da fonte nos segmentos de A para S seja o mesmo que nos de B para S?
- f. E se forem o mesmo hospedeiro?

2. Considere a Figura 3.5. Quais são os valores da porta de fonte e da porta de destino nos segmentos que fluem do servidor de volta aos processos clientes? Quais são os endereços IP nos datagramas de camada de rede que carregam os segmentos de camada de transporte?
3. O UDP e o TCP usam complementos de 1 para suas somas de verificação. Suponha que você tenha as seguintes três palavras de 8 bits: 01010011, 01010100 e 01110100. Qual é o complemento de 1 para as somas dessas palavras? (Note que, embora o UDP e o TCP usem palavras de 16 bits no cálculo da soma de verificação, nesse problema solicitamos que você considere somas de 8 bits). Mostre todo o trabalho. Por que o UDP toma o complemento de 1 da soma, isto é, por que não toma apenas a soma? Com o esquema de complemento de 1, como o destinatário detecta erros? É possível que um erro de 1 bit passe despercebido? E um erro de 2 bits?
4. a. Suponha que você tenha os seguintes bytes: 01011100 e 01010110. Qual é o complemento de da soma desses 2 bytes?
 b. Suponha que você tenha os seguintes bytes: 11011010 e 00110110. Qual é o complemento de 1 da soma desses 2 bytes?
 c. Para os bytes do item (a), dê um exemplo em que um bit é invertido em cada um dos 2 bytes e, mesmo assim, o complemento de um não muda.
5. Suponha que o receptor UDP calcule a soma de verificação da Internet para o segmento UDP recebido e encontre que essa soma coincide com o valor transportado no campo da soma de verificação. O receptor pode estar absolutamente certo de que não ocorreu nenhum erro de bit? Explique.
6. Considere nosso motivo para corrigir o protocolo rdt2.1. Demonstre que o destinatário apresentado na figura 3.57, quando em operação com o remetente mostrado na Figura 3.11, pode levar remetente e destinatário a entrar em estado de travamento, em que cada um espera por um evento que nunca vai ocorrer.
7. No protocolo rdt3.0, os pacotes ACK que fluem do destinatário ao remetente não têm números de sequência (embora tenham um campo de ACK que contém o número de sequência do pacote que estão reconhecendo). Por que nossos pacotes ACK não requerem números de sequência?
8. Elabore a FSM para o lado destinatário do protocolo rdt3.0.
9. Elabore um diagrama de mensagens para a operação do protocolo rdt3.0 quando pacotes de dados e de reconhecimento estão truncados. Seu diagrama deve ser semelhante ao usado na Figura 3.16.
10. Considere um canal que pode perder pacotes, mas cujo atraso máximo é conhecido. Modifique o protocolo rdt2.1 para incluir esgotamento de temporização do remetente e retransmissão. Informalmente, argumente por que seu protocolo pode se comunicar de modo correto por esse canal.

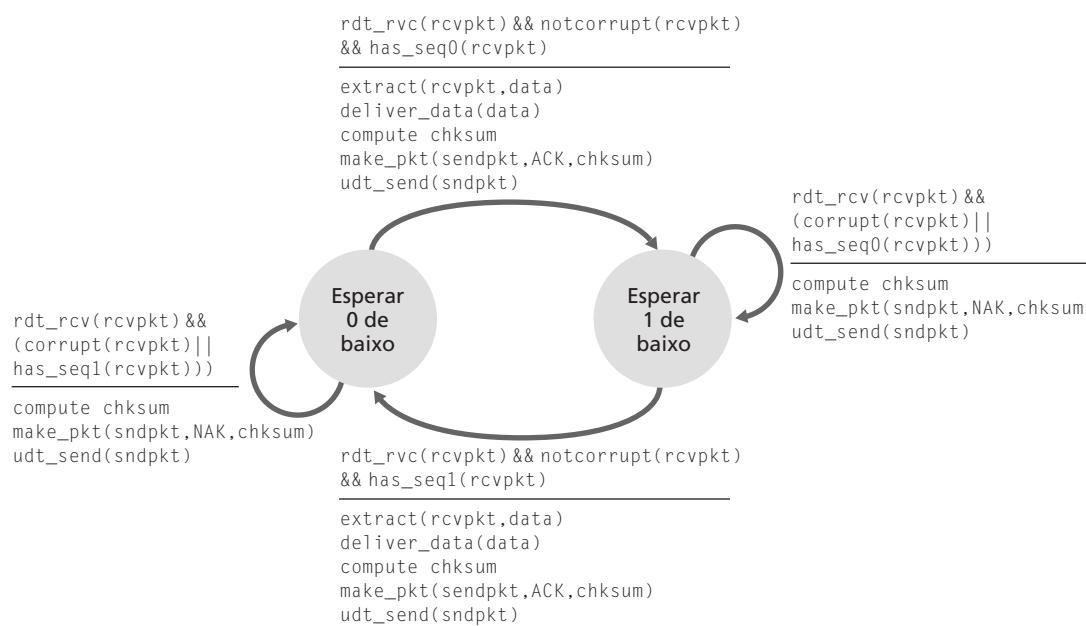


Figura 3.57 Um receptor incorreto para o protocolo rdt 2.1

- 11.** O lado remetente do rdt3.0 simplesmente ignora (isto é, não realiza nenhuma ação) todos os pacotes recebidos que estão errados ou com o valor errado no campo acknum de um pacote de reconhecimento. Suponha que em tais circunstâncias o rdt3.0 devesse apenas retransmitir o pacote de dados corrente. Nesse caso, o protocolo ainda funcionaria? (*Dica:* considere o que aconteceria se houvesse apenas erros de bits; não há perdas de pacotes, mas podem ocorrer esgotamentos de temporização prematuros. Imagine quantas vezes o *enésimo* pacote é enviado, no limite em que n se aproximasse do infinito.)
- 12.** Considere o protocolo rdt3.0. Elabore um diagrama mostrando que, se a conexão de rede entre o remetente e o destinatário puder alterar a ordem de mensagens (isto é, se for possível reordenar duas mensagens que se propagam no meio entre o remetente e o destinatário), então o protocolo bit alternante não funcionará corretamente (lembre-se de identificar claramente o sentido no qual o protocolo não funcionará corretamente). Seu diagrama deve mostrar o remetente à esquerda e o destinatário à direita; o eixo do tempo deverá estar orientado de cima para baixo na página e mostrar a troca de mensagem de dados (D) e de reconhecimento (A). Não esqueça de indicar o número de sequência associado com qualquer segmento de dados ou de reconhecimento.
- 13.** Considere um protocolo de transferência confiável de dados que use somente reconhecimentos negativos. Suponha que o remetente envie dados com pouca frequência. Um protocolo que utiliza somente NAKs seria preferível a um protocolo que utiliza ACKs? Por quê? Agora suponha que o remetente tenha uma grande quantidade de dados para enviar e que a conexão fíme a fíme sofra poucas perdas. Nesse segundo caso, um protocolo que utilize somente NAKs seria preferível a um protocolo que utilize ACKs? Por quê?
- 14.** Considere o exemplo em que se atravessa os Estados Unidos mostrado na Figura 3.17. Que tamanho deveria ter a janela para que a utilização do canal fosse maior do que 95 por cento? Suponha que o tamanho de um pacote seja 1.500 bytes, incluindo os campos do cabeçalho e os dados.
- 15.** Suponha que uma aplicação utilize rdt3.0 como seu protocolo da camada de transporte. Como o protocolo pare e espere possui uma utilização do canal muito baixa (mostrada no exemplo de travessia dos Estados Unidos), os criadores dessa aplicação permitem que o receptor continue enviando de volta um número (mais do que dois) de ACK 0 alternado e ACK 1, mesmo que os dados correspondentes não chegarem ao receptor. O projeto dessa aplicação aumentaria a utilização do canal? Por quê? Há possíveis problemas com esse método? Explique.
- 16.** No protocolo genérico SR que estudamos na Seção 3.4.4, o remetente transmite uma mensagem assim que ela está disponível (se ela estiver na janela), sem esperar por um reconhecimento. Suponha, agora, que queremos um protocolo SR que envie duas mensagens de cada vez. Isto é, o remetente enviará um par de mensagens, e o par de mensagens subsequente somente deverá ser enviado quando o remetente souber que ambas as mensagens do primeiro par foram recebidas corretamente. Suponha que o canal possa perder mensagens, mas que não as corromperá nem as reordenará. Elabore um protocolo de controle de erro para a transferência confiável unidirecional de mensagens. Dê uma descrição FSM do remetente e do destinatário. Descreva o formato dos pacotes enviados entre o remetente e o destinatário e vice-versa. Se você usar quaisquer procedimentos de chamada que não sejam os da Seção 3.4 (por exemplo, `udt_send()`, `start_timer()`, `rdt_rcv()` etc.), esclareça as ações desses procedimentos. Dê um exemplo (um diagrama de mensagens para o remetente e para o destinatário) mostrando como seu protocolo se recupera de uma perda de pacote.
- 17.** Considere um cenário em que o hospedeiro A queira enviar pacotes para os hospedeiros B e C simultaneamente. O hospedeiro A está conectado a B e a C por um canal *broadcast* — um pacote enviado por A é levado pelo canal a B e a C. Suponha que o canal *broadcast* que conecta A, B e C possa, independentemente, perder e corromper mensagens (e assim, por exemplo, uma mensagem enviada de A poderia ser recebida corretamente por B, mas não por C). Projete um protocolo de controle de erro do tipo pare e espere para a transferência confiável de um pacote de A para B e para C, tal que A não receba novos dados da camada superior até que saiba que B e C receberam corretamente o pacote em questão. Dê descrições FSM de A e C. (*Dica:* a FSM para B deve ser essencialmente a mesma que para C.) Também dê uma descrição do(s) formato(s) de pacote usado(s).
- 18.** Considere um cenário em que os hospedeiros A e B queiram enviar mensagens ao Hospedeiro C. Os hospedeiros A e C estão conectados por um canal que pode perder e corromper (e não reordenar) mensagens. Os hospedeiros B e C estão conectados por um outro canal (independente do canal que conecta A e C) com as mesmas propriedades. A camada de transporte no hospedeiro C deve alternar o envio de mensagens de A e B para a camada acima (ou seja, ela deve primeiro transmitir os dados de um pacote de A e depois os dados de um pacote de B, e assim por

- dante.) Elabore um protocolo de controle de erro pare e espere para transferência confiável de pacotes de A e B para C, com envio alternado em C, como descrito acima. Dê descrições FSM de A e C. [Dica: a FSM para B deve ser basicamente a mesma de A.] Dê, também, uma descrição do(s) formato(s) de pacote utilizado(s).
- 19.** Considere o protocolo GBN com um tamanho de janela remetente de 3 e uma faixa de números de sequência de 1.024. Suponha que, no tempo t , o pacote seguinte na ordem, pelo qual o destinatário está esperando, tenha um número de sequência k . Admita que o meio não reordene as mensagens. Responda às seguintes perguntas:
- Quais são os possíveis conjuntos de números de sequência dentro da janela do remetente no tempo t ? Justifique sua resposta.
 - Quais são todos os possíveis valores do campo ACK em todas as mensagens que estão correntemente se propagando de volta ao remetente no tempo t ? Justifique sua resposta.
- 20.** Suponha que haja duas entidades de rede A e B e que B tenha um suprimento de mensagens de dados que será enviado a A de acordo com as seguintes convenções: quando A recebe uma solicitação da camada superior para obter a mensagem de dados (D) seguinte de B, A deve enviar uma mensagem de requisição (R) para B no canal A-a-B; somente quando B receber uma mensagem R, ele poderá enviar uma mensagem de dados (D) de volta a A pelo canal B-a-A; A deve entregar uma cópia de cada mensagem D à camada superior; mensagens R podem ser perdidas (mas não corrompidas) no canal A-a-B; mensagens (D), uma vez enviadas, são sempre entregues corretamente; o atraso entre ambos os canais é desconhecido e variável.
- Elabore um protocolo (dê uma descrição FSM) que incorpore os mecanismos apropriados para compensar a propensão à perda do canal A a B e implemente passagem de mensagem para a camada superior na entidade A, como discutido antes. Utilize apenas os mecanismos absolutamente necessários.
- 21.** Considere os protocolos GBN e SR. Suponha que o espaço de números de sequência seja de tamanho k . Qual será o maior tamanho de janela permitível que evitará que ocorram problemas como os da Figura 3.27 para cada um desses protocolos?
- 22.** Responda verdadeiro ou falso às seguintes perguntas e justifique resumidamente sua resposta:
- Com o protocolo SR, é possível o remetente receber um ACK para um pacote que caia fora de sua janela corrente.
 - Com o GBN, é possível o remetente receber um ACK para um pacote que caia fora de sua janela corrente.
 - O protocolo bit alternante é o mesmo que o protocolo SR com janela do remetente e do destinatário de tamanho 1.
 - O protocolo bit alternante é o mesmo que o protocolo GBN com janela do remetente e do destinatário de tamanho 1.
- 23.** Dissemos que um aplicação pode escolher o UDP para um protocolo de transporte pois ele oferece um controle de aplicações melhor (do que o TCP) de quais dados são enviados em um segmento e quando isso ocorre.
- Por que uma aplicação possui mais controle de quais dados são enviados em um segmento?
 - Por que uma aplicação possui mais controle de quando o segmento é enviado?
- 24.** Considere a transferência de um arquivo enorme de L bytes do hospedeiro A para o hospedeiro B. Suponha um MSS de 536 bytes.
- Qual é o máximo valor de L tal que não sejam exauridos os números de sequência TCP? Lembre-se de que o campo de número de sequência TCP tem quatro bytes.
 - Para o L que obtiver em (a), descubra quanto tempo demora para transmitir o arquivo. Admita que um total de 66 bytes de cabeçalho de transporte, de rede e de enlace de dados seja adicionado a cada segmento antes que o pacote resultante seja enviado por um enlace de 155 Mbps. Ignore controle de fluxo e controle de congestionamento de modo que A possa enviar os segmentos um atrás do outro e continuamente.
- 25.** Os hospedeiros A e B estão se comunicando por meio de uma conexão TCP, e o hospedeiro B já recebeu de A todos os bytes até o byte 126. Suponha que o Hospedeiro A envie, então, dois segmentos para o Hospedeiro B sucessivamente. O primeiro e o segundo segmentos contêm 70 e 50 bytes de dados, respectivamente. No primeiro segmento, o número de sequência é 127, o número de porta de partida é 302, e o número de porta de destino é 80. O hospedeiro B envia um reconhecimento ao receber um segmento do hospedeiro A.
- No segundo segmento enviado do Hospedeiro A para N, quais são o número de sequência, a porta de partida e a porta de destino?
 - Se o primeiro segmento chegar antes do segundo, no reconhecimento do primeiro segmento que chegar, qual é o número do reconhecimento, da porta de partida e da porta de destino?

- c. Se o segundo segmento chegar antes do primeiro, no reconhecimento do primeiro segmento que chegar, qual é o número do reconhecimento?
- d. Suponha que dois segmentos enviados por A cheguem em ordem a B. O primeiro reconhecimento é perdido e o segundo chega após o primeiro intervalo do esgotamento de temporização. Elabore um diagrama de temporização, mostrando esses segmentos, e todos os outros, e os reconhecimentos enviados. (Admita que não haja nenhuma perda de pacote adicional.) Para cada segmento de seu desenho, apresente o número de sequência e o número de bytes de dados; para cada reconhecimento adicionado por você, apresente o número do reconhecimento.
- 26.** Os hospedeiros A e B estão diretamente conectados com um enlace de 100 Mbps. Existe uma conexão TCP entre os dois hospedeiros, e o hospedeiro A está enviando ao B um arquivo enorme por meio dessa conexão. O hospedeiro A pode enviar seus dados da aplicação para o socket TCP a uma taxa que chega a 120 Mbps, mas o Hospedeiro B pode ler o buffer de recebimento TCP a uma taxa de 60 Mbps. Descreva o efeito do controle de fluxo do TCP.
- 27.** Os cookies SYN foram discutidos na Seção 3.5.6.
- Por que é necessário que o servidor use um número de sequência especial no SYNACK?
 - Suponha que um atacante saiba que um hospedeiro-alvo utilize SYN cookies. O atacante consegue criar conexões semiaberta ou completamente abertas simplesmente enviando um pacote ACK para o alvo? Por que sim? Por que não?
 - Suponha que um atacante receba uma grande quantidade de números de sequência enviados pelo servidor. O atacante consegue fazer com que o servidor crie muitas conexões totalmente abertas enviando ACKs com esses números de sequência? Por quê?
- 28.** Considere a rede mostrada no Cenário 2 na Seção 3.6.1. Suponha que os hospedeiros emissores A e B possuam valores de esgotamento de temporização fixos.
- Analise o fato de que aumentar o tamanho do buffer finito do roteador pode possivelmente reduzir a vazão (λ_{out}).
 - Agora suponha que os hospedeiros ajustem dinamicamente seus valores de esgotamento de temporização (como o que o TCP faz) baseado no atraso no buffer no roteador. Aumentar o tamanho do buffer ajudaria a aumentar a vazão? Por quê?
- 29.** Considere o procedimento TCP para estimar RTT. Suponha que $\alpha = 0,1$. Seja SampleRTT₁ a amostra mais recente de RTT, SampleRTT₂ a seguinte amostra mais recente de RTT etc.
- Para uma dada conexão TCP, suponha que quatro reconhecimentos foram devolvidos com as amostras RTT correspondentes SampleRTT₄, SampleRTT₃, SampleRTT₂ e SampleRTT₁. Expressse EstimatedRTT em termos das quatro amostras RTT.
 - Generalize sua fórmula para n amostras de RTTs.
 - Para a fórmula em (b), considere n tendendo ao infinito. Comente por que esse procedimento de média é denominado média móvel exponencial.
- 30.** Na Seção 3.5.3 discutimos estimativa de RTT para o TCP. Na sua opinião, por que o TCP evita medir o SampleRTT para segmentos retransmitidos?
- 31.** Qual é a relação entre a variável SendBase na Seção 3.5.4 e a variável LastByteRcvd na Seção 3.5.5?
- 32.** Qual é a relação entre a variável LastByteRcvd na Seção 3.5.5 e a variável y na seção 3.5.4?
- 33.** Na Seção 3.5.4 vimos que o TCP espera até receber três ACKs duplicados antes de realizar uma retransmissão rápida. Na sua opinião, por que os projetistas do TCP preferiram não realizar uma retransmissão rápida após ser recebido o primeiro ACK duplicado para um segmento?
- 34.** Compare o GBN, SR e o TCP (sem ACK retardado). Admita que os valores do esgotamento de temporização para os três protocolos sejam suficientemente longos de tal modo que cinco segmentos de dados consecutivos e seus ACKs correspondentes possam ser recebidos (se não perdidos no canal) por um hospedeiro receptor (hospedeiro B) e por um hospedeiro emissor (hospedeiro A), respectivamente. Suponha que o hospedeiro A envie cinco segmentos de dados para o hospedeiro B, e que o segundo segmento (enviado de A) esteja perdido. No fim, todos os 5 segmentos de dados foram corretamente recebidos pelo hospedeiro A.
- Quantos segmentos o hospedeiro A enviou no total e quantos ACKs o hospedeiro B enviou no total? Quais são seus números de sequência? Responda essa questão considerando os três protocolos.
 - Se os valores do esgotamento de temporização para os três protocolos forem muito maiores do que 5 RTT, então qual protocolo envia com sucesso todos os cinco segmentos de dados em um menor intervalo de tempo?
- 35.** Em nossa descrição sobre o TCP na Figura 3.53, o valor do limiar, ssthresh, é definido para $ssthresh = cwnd/2$ em diversos lugares e o valor ssthresh é referido como sendo definido para metade do tama-

- nho da janela quando ocorreu um evento de perda. A taxa à qual o emissor está enviando quando ocorreu o evento de perda deve ser aproximadamente igual a segmentos cwnd por RTT? Explique sua resposta. Se sua resposta for negativa, você pode sugerir uma maneira diferente pela qual ssthresh deve ser definido?
36. Considere a Figura 3.46(b). Se λ'_{in} aumentar para mais do que $R/2$, λ'_{out} poderá aumentar para mais do que $R/3$? Explique. Agora considere a Figura 3.46(c). Se λ'_{in} aumentar para mais do que $R/2$, λ'_{out} poderá aumentar para mais de $R/4$ admitindo-se que um pacote será transmitido duas vezes, em média, do roteador para o destinatário? Explique.
37. Considere a Figura 3.58. Admitindo-se que TCP Reno é o protocolo que experimenta o comportamento mostrado no gráfico, responda às seguintes perguntas. Em todos os casos você deverá apresentar uma justificativa resumida para sua resposta.
- Quais os intervalos de tempo em que a partida lenta do TCP está em execução?
 - Quais os intervalos de tempo em que a prevenção de congestionamento do TCP está em execução?
 - Após a 16^a rodada de transmissão, a perda de segmento será detectada por três ACKs duplicados ou por um esgotamento de temporização?
 - Após a 22^a rodada de transmissão, a perda de segmento será detectada por três ACKs duplicados ou por um esgotamento de temporização?
 - Qual é o valor inicial de ssthresh na primeira rodada de transmissão?
 - Qual é o valor inicial de ssthresh na 18^a rodada de transmissão?
38. Consulte a Figura 3.56, que ilustra a convergência do algoritmo AIMD do TCP. Suponha que, em vez de uma diminuição multiplicativa, o TCP reduza o tamanho da janela de uma quantidade constante. O AIMD resultante convergiria a um algoritmo de igual compartilhamento? Justifique sua resposta usando um diagrama semelhante ao da Figura 3.56.
39. Na Seção 3.5.4 discutimos a duplicação do intervalo de temporização após um evento de esgotamento de temporização. Esse mecanismo é uma forma de controle de congestionamento. Por que o TCP precisa de um mecanismo de controle de congestionamento que utiliza janelas (como estudado na Seção 3.7) além desse mecanismo de duplicação do intervalo de esgotamento de temporização?

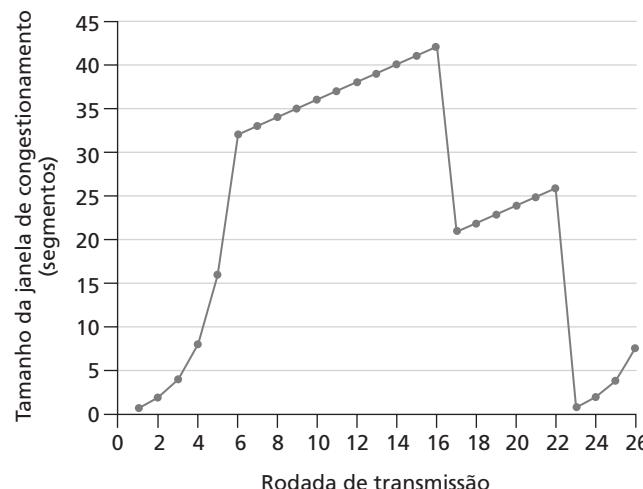


Figura 3.58 Tamanho da janela TCP como uma função de tempo

- 40.** O hospedeiro A está enviando um arquivo enorme ao hospedeiro B por uma conexão TCP. Nessa conexão nunca há perda de pacotes e os temporizadores nunca se esgotam. Seja R bps a taxa de transmissão do enlace que liga o hospedeiro A à Internet. Suponha que o processo no hospedeiro A consiga enviar dados para seu socket TCP a uma taxa de S bps, em que $S = 10 \cdot R$. Suponha ainda que o buffer de recepção do TCP seja grande o suficiente para conter o arquivo inteiro e que o buffer de envio possa conter somente um por cento do arquivo. O que impediria o processo no hospedeiro A de passar dados continuamente para seu socket TCP à taxa de S bps: o controle de fluxo do TCP; o controle de congestionamento do TCP; ou alguma outra coisa? Elabore sua resposta.
- 41.** Considere enviar um arquivo grande de um computador a outro por meio de uma conexão TCP em que não haja perda.
- Suponha que o TCP utilize AIMD para seu controle de congestionamento sem partida lenta. Admitindo que $cwnd$ aumenta 1 MSS sempre que um lote de ACK é recebido e os tempos da viagem de ida e volta constantes, quanto tempo leva para $cwnd$ aumentar de 5 MSS para 11 MSS? (admitindo nenhum evento de perda)?
 - Qual é a vazão média (em termos de MSS e RTT) para essa conexão sendo o tempo = 6 RTT?
- 42.** Relembre a descrição macroscópica da vazão do TCP. No período de tempo transcorrido para a taxa da conexão variar de $W/(2 \cdot RTT)$ a W/RTT , apenas um pacote é perdido (bem ao final do período).
- Mostre que a taxa de perda (fração de pacotes perdidos) é igual a
- $$L = \text{fração de pacotes perdidos} = \frac{1}{\frac{3}{8} W^2 + \frac{3}{4} W}$$
- Use o resultado anterior para mostrar que, se uma conexão tiver taxa de perda L , sua largura de banda média é dada aproximadamente por:
- $$\approx \frac{1.22 \cdot MSS}{RTT \sqrt{L}}$$
- 43.** Considere que somente uma única conexão TCP (Reno) utiliza um enlace de 10 Mbps que não armazena nenhum dado. Suponha que esse enlace seja o único congestionado entre os hospedeiros emissor e receptor. Admita que o emissor TCP tenha um arquivo enorme para enviar ao receptor e o buffer de recebimento do receptor é muito maior do que a janela de congestionamento. Também fazemos as seguintes suposições: o tamanho de cada segmento TCP é 1.500 bytes; o atraso de propagação bidirecional dessa conexão é 10 milissegundos; e essa conexão está sempre na fase de prevenção de congestionamento, ou seja, ignore a partida lenta.
- Qual é o tamanho máximo da janela (em segmentos) que a conexão TCP pode atingir?
 - Qual é o tamanho médio da janela (em segmentos) e a vazão média (em bps) dessa conexão TCP?
 - Quanto tempo essa conexão TCP leva para alcançar sua janela máxima novamente após se recuperar de uma perda de pacote?
- 44.** Considere o cenário descrito na questão anterior. Suponha que o enlace de 10 Mbps possa armazenar um número finito de segmentos. Demonstre que para o enlace sempre enviar dados, teríamos que escolher um tamanho de buffer que é, pelo menos, o produto da velocidade do enlace C e o atraso de propagação bidirecional entre o emissor e o receptor.
- 45.** Repita a questão 43, mas substituindo o enlace de 10 Mbps por um de 10 Gbps. Observe que em sua resposta ao item (c), você verá que o tamanho da janela de congestionamento leva muito tempo para atingir seu tamanho máximo após se recuperar de uma perda de pacote. Elabore uma solução para resolver este problema.
- 46.** Suponha que T (medido por RTT) seja o intervalo de tempo que uma conexão TCP leva para aumentar seu tamanho de janela de congestionamento de $W/2$ para W , sendo W o tamanho máximo da janela de congestionamento. Demonstre que T é uma função da vazão média do TCP.
- 47.** Considere um algoritmo AIMD do TCP simplificado, sendo o tamanho da janela de congestionamento medido em número de segmentos e não em bytes. No aumento aditivo, o tamanho da janela de congestionamento aumenta por um segmento em cada RTT. Na diminuição multiplicativa, o tamanho da janela de congestionamento diminui para metade (se o resultado não for um número inteiro, arredondar para o número inteiro mais próximo). Suponha que duas conexões TCP, C_1 e C_2 compartilhem um único enlace congestionado com 30 segmentos por segundo de velocidade. Admita que C_1 e C_2 estejam na fase de prevenção de congestionamento. GRTT da conexão C_1 é 100ms e o da conexão C_2 é de 200ms. Suponha que quando a taxa de dados no enlace excede a velocidade do enlace, todas as conexões TCP sofrem perda de segmento de dados.
- Se C_1 e C_2 no tempo t_0 possuem uma janela de congestionamento de 10 segmentos, quais são

- seus tamanhos de janela de congestionamento após 2.200 milissegundos?
- b. No final das contas, essas duas conexões obterão a mesma porção da largura de banda do enlace congestionado? Explique.
48. Considere a rede descrita na questão anterior. Agora suponha que as duas conexões TCP, C_1 e C_2 , possuam o mesmo RTT de 100 milissegundos e que no tempo t_0 o tamanho da janela de congestionamento de C_1 seja 15 segmentos, e que de C_2 seja 10 segmentos.
- a. Quais são os tamanhos de suas janelas de congestionamento após 2.200 milissegundos?
- b. No final das contas, essas duas conexões irão obter a mesma porção da largura de banda do enlace congestionado?
- c. Dizemos que duas conexões são sincronizadas se ambas atingirem o tamanho máximo e mínimo de janela ao mesmo tempo. No final das contas, essas duas conexões serão sincronizadas? Se sim, quais os tamanhos máximos de janela?
- d. Essa sincronização ajudará a melhorar a utilização do enlace compartilhado? Por quê? Elabore alguma ideia para romper essa sincronização.
49. Considere uma modificação ao algoritmo de controle de congestionamento do TCP. Em vez do aumento aditivo, podemos utilizar a diminuição multiplicativa. Um emissor TCP aumenta seu tamanho de janela por uma constante positiva pequena ($0 < a < 1$) ao receber um ACK válido. Encontre a relação funcional entre a taxa de perda L e a janela máxima de congestionamento W . Para esse TCP modificado, demonstre, independentemente da vazão média TCP, que uma conexão TCP sempre gasta a mesma quantidade de tempo para aumentar seu tamanho da janela de congestionamento de $W/2$ para W .
50. Quando discutimos TCPs futuros na Seção 3.7, observamos que, para conseguir uma vazão de 10 Gbps, o TCP apenas poderia tolerar uma probabilidade de perda de segmentos de $2 \cdot 10^{-10}$ (ou, equivalentemente, uma perda para cada 5.000.000.000 segmentos). Mostre a derivação dos valores para $2 \cdot 10^{-10}$ ou 1: 5.000.000 a partir dos valores de RTT e do MSS dados na Seção 3.7. Se o TCP precisasse suportar uma conexão de 100 Gbps, qual seria a perda tolerável?
51. Quando discutimos controle de congestionamento em TCP na Seção 3.7, admitimos implicitamente que o remetente TCP sempre tinha dados para enviar. Agora considere o caso em que o remetente TCP envie uma grande quantidade de dados e então fique ocioso em t_1 (já que não há mais dados a enviar). O TCP permanecerá ocioso por um período de tempo relativamente longo e então irá querer enviar mais dados em t_2 . Quais são as vantagens e desvantagens do TCP utilizar os valores cwnd e ssthresh de t_1 quando começar a enviar dados em t_2 ? Que alternativa você recomendaria? Por quê?
52. Neste problema, verificamos se o UDP ou o TCP apresentam um grau de autenticação do ponto de chegada.
- a. Considere um servidor que recebe uma solicitação dentro de um pacote UDP e responde a essa solicitação dentro de um pacote UDP (por exemplo, como feito por um servidor DNS). Se um cliente com endereço IP X o engana com o endereço Y, para onde o servidor enviará sua resposta?
- b. Suponha que um servidor receba um SYN de endereço-fonte IP Y com o número de reconhecimento correto. Admitindo que o servidor escolha um número de sequência aleatório e que não haja um "man-in-the-middle", o servidor pode ter certeza de que o cliente realmente está em Y (e não em outro endereço X que está enganando Y)?
53. Neste problema, consideramos o atraso apresentado pela fase de partida lenta do TCP. Considere um cliente e um servidor da Web diretamente conectados por um enlace de taxa R . Suponha que o cliente queira restaurar um objeto cujo tamanho seja exatamente igual a 15 S, sendo S o tamanho máximo do segmento (MSS). Considere RTT como o tempo de viagem de ida e volta entre o cliente e o servidor (admitindo que seja constante). Ignorando os cabeçalhos do protocolo, determine o tempo para restaurar o objeto (incluindo o estabelecimento da conexão TCP) quando
- a. $4 S/R > S/R + RTT > 2S/R$
- b. $S/R + RTT > 4 S/R$
- c. $S/R > RTT$



Questões dissertativas

1. O que é sequestro de TCP? Como ele pode ser realizado?
2. Na Seção 3.7, observamos que um cliente-servidor pode criar muitas conexões paralelas “de modo injusto”. O que pode ser feito para tornar a Internet verdadeiramente justa?
3. Consulte a literatura de pesquisa para saber o que quer dizer TCP *amigável*. Leia também a entrevista de Sally Floyd ao final deste capítulo. Redija uma página descrevendo a característica *amigável* do TCP.
4. Ao final da Seção 3.7.1, discutimos o fato de uma aplicação querer abrir várias conexões TCP e obter uma vazão mais alta (ou, o que é equivalente, um tempo menor de transferência de dados). O que aconteceria se todas as aplicações tentassem melhorar seus desempenhos utilizando conexões múltiplas? Cite algumas das dificuldades envolvidas na utilização de um elemento da rede para determinar se uma aplicação está ou não usando conexões TCP múltiplas.
5. Além da varredura de portas TCP e UDP, o nmap possui qual funcionalidade? Capture traços de pacote com o Ethereal (ou qualquer outro analisador de pacote) das trocas de pacote nmap. Utilize as trilhas para explicar como funcionam alguns recursos avançados.
6. Em nossa descrição sobre o TCP na Figura 3.53, o valor inicial de cwnd é definido para 1. Consulte a literatura de pesquisa e uma RFC da Internet e discuta métodos alternativos que foram propostos para definir o valor inicial de cwnd.
7. Consulte a literatura a respeito do SCTP [RFC 2960, RFC 3286]. Para quais aplicações os criadores do SCTP consideraram-no ser usado? Quais recursos do SCTP foram adicionados para atender às necessidades dessas aplicações?



Tarefas de programação

Detalhes completos das tarefas de programação podem ser encontrados no site http://www.aw.com/kurose_br.

Implementando um protocolo de transporte confiável

Nesta tarefa de programação de laboratório, você escreverá o código para a camada de transporte do remetente e do destinatário no caso da implementação de um protocolo simples de transferência confiável de dados. Há duas versões deste laboratório: a do protocolo de bit alternante e a do GBN. Essa tarefa será muito divertida, já que a sua implementação não será muito diferente da que seria exigida em uma situação real.

Como você provavelmente não tem máquinas autônomas (com um sistema operacional que possa modificar), seu código terá de rodar em um ambiente hardware/software simulado. Contudo, a interface de programação

fornecida às suas rotinas — isto é, o código que chama suas entidades de cima (da camada 5) e de baixo (da camada 3) — é muito próxima ao que é feito em um ambiente UNIX real. (Na verdade, as interfaces do software descritas nesta tarefa de programação são muito mais realistas do que os remetentes e destinatários de laço infinito descritos em muitos livros.) A parada e o acionamento dos temporizadores também são simulados, e as interrupções do temporizador farão com que sua rotina de tratamento de temporização seja ativada.

A tarefa completa de laboratório, assim como o código necessário para compilar seu próprio código, estão disponíveis no site: http://www.aw.com/kurose_br.



Wireshark Lab: Explorando o TCP

Neste laboratório, você usará seu browser Web para acessar um arquivo de um servidor Web. Como nos Wireshark Labs anteriores, você usará Wireshark para capturar os pacotes que estão chegando ao seu computador. Mas, diferentemente daqueles laboratórios, também

poderá descargar do mesmo servidor Web do qual descarregou o arquivo, um relatório de mensagens (trace) que pode ser lido pelo Wireshark. Nesse relatório de mensagens do servidor, você encontrará os pacotes que foram gerados pelo seu próprio acesso ao servidor Web. Você

analisará os diagramas dos lados do cliente e do servidor de modo a explorar aspectos do TCP e, em especial, fará uma avaliação do desempenho da conexão TCP entre seu computador e o servidor Web. Utilizando o diagrama de mensagens, você analisará o comportamento da janela TCP e inferirá comportamentos de perda de pacote, de

retransmissão, de controle de fluxo e de controle de congestionamento e do tempo de ida e volta estimado.

Como acontece com todos os Wireshark Labs, a descrição completa deste pode ser encontrada, em inglês, no site http://www.aw.com/kurose_br.



Wireshark Lab: explorando o UDP

Neste pequeno laboratório, você realizará uma captura de pacote e uma análise de sua aplicação favorita que utiliza o UDP (por exemplo, o DNS ou uma aplicação multimídia, como o Skype). Como aprendemos na Seção 3.3, o UDP é um protocolo de transporte simples. Neste laboratório,

você examinará os campos do cabeçalho no segmento UDP, assim como o cálculo da soma de verificação.

Como acontece com todos os Wireshark Labs, a descrição completa deste pode ser encontrada, em inglês, no site http://www.aw.com/kurose_br.

Entrevista

Sally Floyd

Sally Floyd é cientista-pesquisadora no ICSI Center for Internet Research, um instituto dedicado a questões da Internet e de redes. Ela é conhecida no setor por seu trabalho em projetos de protocolos de Internet, em especial nas áreas de multicast confiável, controle de congestionamento (TCP), escalonamento de pacotes (RED) e análise de protocolos. Sally é bacharel em Sociologia pela Universidade da Califórnia em Berkeley, e mestre e doutora em ciência da computação por essa mesma universidade.



Como você decidiu estudar ciência da computação?

Após terminar o bacharelado em sociologia, comecei a me preocupar em obter meu próprio sustento. Acabei fazendo um curso de eletrônica de dois anos na faculdade local e, em seguida, passei dez anos trabalhando com eletrônica e ciência da computação. Nesses dez anos, estão incluídos os oito que trabalhei como engenheira de sistemas de computação, cuidando dos computadores dos trens da Bay Area Rapid Transit (BART). Mais tarde, decidi estudar ciência da computação mais formalmente e iniciei a pós-graduação no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley.

O que a fez se decidir pela especialização em redes?

Na pós-graduação, comecei a me interessar pela parte teórica da ciência da computação. Primeiramente, dediquei-me à análise probabilística de algoritmos e, mais tarde, à teoria do aprendizado da computação. Nessa época, trabalhava também no Lawrence Berkeley Laboratory (LBL) uma vez por mês, e meu escritório ficava em frente ao escritório de Van Jacobson, que pesquisava algoritmos de controle de congestionamento no TCP. Certo dia, Van me perguntou se eu gostaria de trabalhar na análise de alguns algoritmos para um problema relacionado a redes, que envolvia a indesejada sincronização de mensagens periódicas de roteamento. Isso me pareceu interessante, e foi o que fiz durante o verão.

Após eu ter concluído minha tese, Van me ofereceu um emprego de tempo integral para continuar trabalhando com redes. Eu não tinha planejado ficar trabalhando com redes durante dez anos, mas, para mim, a pesquisa é mais gratificante do que a ciência teórica da computação. Sinto mais satisfação em trabalhos aplicados, pois as consequências do que faço são mais tangíveis.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Meu primeiro emprego na área de computação foi na BART, onde trabalhei de 1975 a 1982 com os computadores que controlavam os trens. Comecei como técnica em manutenção e conserto dos vários sistemas distribuídos de computadores envolvidos na operação do sistema da BART.

Entre esses sistemas estavam: um sistema central de computadores e um sistema distribuído de minicomputadores, que controlavam a movimentação dos trens; um sistema de computadores da DEC, que exibia avisos e destinos de trens nos painéis, e um sistema de computadores Modcomp, que coletava informações das bilheterias. Meus últimos anos na BART foram dedicados a um projeto conjunto BART/LBL para estruturar um sistema substituto para o sistema de computadores de controle de trens da BART, que já estava ficando ultrapassado.



Qual parte de seu trabalho lhe apresenta mais desafios?

A pesquisa é a parte mais desafiadora. O primeiro tópico de pesquisa inclui explorar as questões referentes ao controle de congestionamento para aplicações, como o fluxo de mídia. Um segundo tópico é abordar obstáculos da rede para uma comunicação mais explícita entre os roteadores e os nós finais. Esses obstáculos podem incluir túneis IP e caminhos MPLS, roteadores ou dispositivos intermediários que soltam os pacotes contendo opções IP, redes complexas da camada 2 e potenciais para ataques na rede. Um terceiro tópico existente é explorar como nossa escolha de modelos de cenários para uso em análise, simulação e experimentos afeta nossa avaliação do desempenho dos mecanismos de controle de congestionamento. Mais informações sobre esses assuntos podem ser encontradas nas páginas Web do DCCP, Quick-Start e TMRG em <http://www.icir.org/floyd>.

Em sua opinião, qual é o futuro das redes e da Internet?

Uma possibilidade é que o congestionamento típico encontrado pelo tráfego da Internet se torne menos severo à medida que a largura de banda disponível aumentar mais rapidamente do que a demanda. Considero que a tendência está se dirigindo para um congestionamento menos intenso, embora não pareça de todo impossível contemplar um futuro de crescente congestionamento, a médio prazo, pontuado por colapso ocasional.

O futuro da Internet em si ou da arquitetura da Internet ainda não está claro para mim. Há muitos fatores que estão contribuindo para rápidas mudanças. Portanto, é difícil predizer como a Internet ou a arquitetura da Internet evoluirão ou mesmo prever o grau de sucesso que essa evolução será capaz de alcançar na prevenção das muitas armadilhas potenciais ao longo do caminho.

Uma tendência negativa bem conhecida é a dificuldade crescente de realizar mudanças na arquitetura da Internet, que não é mais um conjunto coerente, e os vários componentes como protocolos de transporte, mecanismos do roteador, firewalls,平衡adores de carga, mecanismos de segurança e seus semelhantes às vezes trabalham com objetivos contrários.

Quais pessoas a inspiraram profissionalmente?

Richard Karp, o orientador de minha tese, me ensinou a fazer, de fato, pesquisa. Van Jacobson, meu “chefe de grupo” no LBL, foi o responsável pelo meu interesse no estudo de redes e por grande parte do meu aprendizado sobre a infraestrutura da Internet. Dave Clark me inspirou por sua visão clara da arquitetura da Internet e por seu papel no desenvolvimento dessa arquitetura por meio de pesquisa, de artigos e da participação na IETF e em outros fóruns públicos. Deborah Estrin me inspirou com sua capacidade de concentração e efetividade, e sua habilidade em tomar decisões lúcidas sobre o que investigar e por quê.

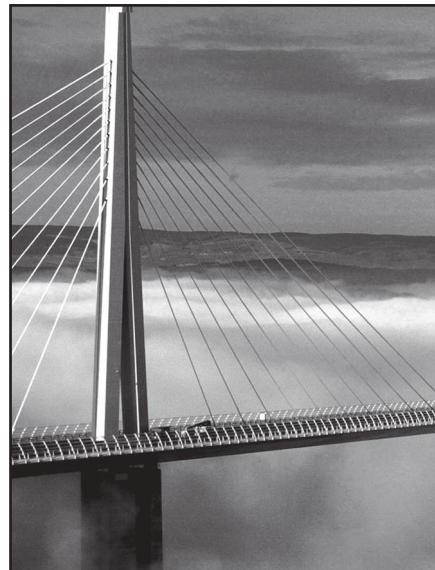
Uma das razões por que aprecio trabalhar na pesquisa de rede é que encontrei nesse campo muitas pessoas de que gosto, a quem respeito e admiro. São pessoas inteligentes, batalhadoras e que estão profundamente comprometidas com o desenvolvimento da Internet, divergem (ou concordam) de modo amigável e são bons parceiros para uma cerveja após um dia de reuniões.





Capítulo 4

A camada de rede



Vimos no capítulo anterior que a camada de transporte provê várias formas de comunicação processo a processo com base no serviço de comunicação hospedeiro a hospedeiro da camada de rede. Vimos também que a camada de transporte faz isso sem saber como a camada de rede implementa esse serviço. Portanto, é bem possível que agora você esteja imaginando o que está por baixo do serviço de comunicação hospedeiro a hospedeiro; o que o faz funcionar.

Neste capítulo estudaremos exatamente como a camada de rede implementa o serviço de comunicação hospedeiro a hospedeiro. Veremos que há um pedaço da camada de rede em cada um dos hospedeiros e roteadores na rede, o que não acontece com a camada de transporte. Por causa disso, os protocolos de camada de rede estão entre os mais desafiadores (e, portanto, os mais interessantes!) da pilha de protocolos.

A camada de rede é, também, uma das camadas mais complexas da pilha de protocolos e, assim, temos um longo caminho a percorrer. Iniciaremos nosso estudo com uma visão geral da camada de rede e dos serviços que ela pode prover. Em seguida, examinaremos novamente as duas abordagens da estruturação da entrega de pacotes de camada de rede — o modelo de datagramas e o modelo de circuitos virtuais — que vimos pela primeira vez no Capítulo 1 e veremos o papel fundamental que o endereçamento desempenha na entrega de um pacote a seu hospedeiro de destino.

Faremos, neste capítulo, uma distinção importante entre as funções de **repasse** e **roteamento** da camada de rede. Repasse envolve a transferência de um pacote de um enlace de entrada para um enlace de saída dentro de um único roteador. Roteamento envolve *todos* os roteadores de uma rede, cujas interações coletivas por meio do protocolo de roteamento determinam os caminhos que os pacotes percorrem em suas viagens do nó de origem ao nó de destino. Essa será uma distinção importante para manter em mente ao ler este capítulo.

Para aprofundar nosso conhecimento do repasse de pacotes, examinaremos o “interior” de um roteador — a organização e a arquitetura de seu hardware. Então, examinaremos o repasse de pacotes na Internet, juntamente com o famoso Protocolo da Internet (IP). Investigaremos o endereçamento na camada de rede e o formato de datagrama IPv4. Em seguida, estudaremos a tradução de endereço de rede (*network address translation* — NAT), a fragmentação de datagrama, o Protocolo de Mensagem de Controle da Internet (*Internet Control Message Protocol* — ICMP) e IPv6.

Então voltaremos nossa atenção à função de roteamento da camada de rede. Veremos que o trabalho de um algoritmo de roteamento é determinar bons caminhos (ou rotas) entre remetentes e destinatários. Em primeiro lugar, estudaremos a teoria de algoritmos de roteamento concentrando nossa atenção nas duas classes de algoritmos mais predominantes: algoritmos de estado de enlace e de vetor de distâncias. Visto que a complexidade

dos algoritmos de roteamento aumenta consideravelmente com o aumento do número de roteadores na rede, também será interessante abordar o roteamento hierárquico. Em seguida veremos como a teoria é posta em prática quando falarmos sobre os protocolos de roteamento de intrassistemas autônomos da Internet (RIP, OSPF e IS-IS) e seu protocolo de roteamento de intersistemas autônomos, o BGP. Encerraremos este capítulo com uma discussão sobre roteamento *broadcast* e *multicast*.

Em resumo, este capítulo tem três partes importantes. A primeira, seções 4.1 e 4.2, aborda funções e serviços de camada de rede. A segunda, seções 4.3 e 4.4, examina o repasse. Finalmente, a terceira parte, seções 4.5 a 4.7, estuda roteamento.

4.1 Introdução

A Figura 4.1 mostra uma rede simples com dois hospedeiros, H1 e H2, e diversos roteadores no caminho entre H1 e H2. Suponha que H1 esteja enviando informações a H2 e considere o papel da camada de rede nesses hospedeiros e nos roteadores intervenientes. A camada de rede em H1 pegará segmentos da camada de transporte em H1, encapsulará cada segmento em um datagrama (isto é, em um pacote de camada de rede) e então dará início à jornada dos datagramas até seu destino, isto é, enviará os datagramas para seu roteador vizinho, R1. No hospedeiro receptor (H2), a camada de rede receberá os datagramas de seu roteador vizinho R2, extrairá os segmentos de camada de transporte e os entregará à camada de transporte em H2. O papel primordial dos roteadores é repassar datagramas de enlaces de entrada para enlaces de saída. Note que os roteadores da Figura 4.1 são mostrados com a pilha de protocolos truncada, isto é, sem as camadas superiores acima da camada de rede, porque (exceto para finalidades de controle) roteadores não rodam protocolos de camada de transporte e de aplicação como os que examinamos nos capítulos 2 e 3.

4.1.1 Repasse e roteamento

Assim, o papel da camada de rede é aparentemente simples — transportar pacotes de um hospedeiro remetente a um hospedeiro destinatário. Para fazê-lo, duas importantes funções da camada de rede podem ser identificadas:

Repasse. Quando um pacote chega ao enlace de entrada de um roteador, este deve conduzi-lo até o enlace de saída apropriado. Por exemplo, um pacote proveniente do hospedeiro H1 que chega ao roteador R1 deve ser repassado ao roteador seguinte por um caminho até H2. Na Seção 4.3 examinaremos o interior de um roteador e investigaremos como um pacote é realmente repassado de um enlace de entrada de um roteador até um enlace de saída.

Roteamento. A camada de rede deve determinar a rota ou caminho tomado pelos pacotes ao fluírem de um remetente a um destinatário. Os algoritmos que calculam esses caminhos são denominados **algoritmos de roteamento**. Um algoritmo de roteamento determinaria, por exemplo, o caminho ao longo do qual os pacotes fluiriam de H1 para H2.

Os termos *repasse* e *roteamento* são usados indistintamente por autores que discutem a camada de rede. Neste livro, usaremos esses termos com maior exatidão. *Repasse* refere-se à ação local realizada por um roteador para transferir um pacote da interface de um enlace de entrada para a interface de enlace de saída apropriada. *Roteamento* refere-se ao processo de âmbito geral da rede que determina os caminhos fim a fim que os pacotes percorrem desde a fonte até o destino. Para usar uma viagem como analogia, voltemos àquele nosso viajante da Seção 1.3.2, que vai da Pensilvânia à Flórida. Durante a viagem, nosso motorista passa por muitos cruzamentos de rodovias em sua rota até a Flórida. Podemos imaginar o repasse como o processo de passar por um único cruzamento: um carro chega ao cruzamento vindo de uma rodovia e determina qual rodovia ele deve pegar para sair do cruzamento. Podemos imaginar o roteamento como o processo de planejamento da viagem da Pensilvânia até a Flórida: antes de partir, o motorista consultou um mapa e escolheu um dos muitos caminhos possíveis. Cada um desses caminhos consiste em uma série de trechos de rodovias conectados por cruzamentos.

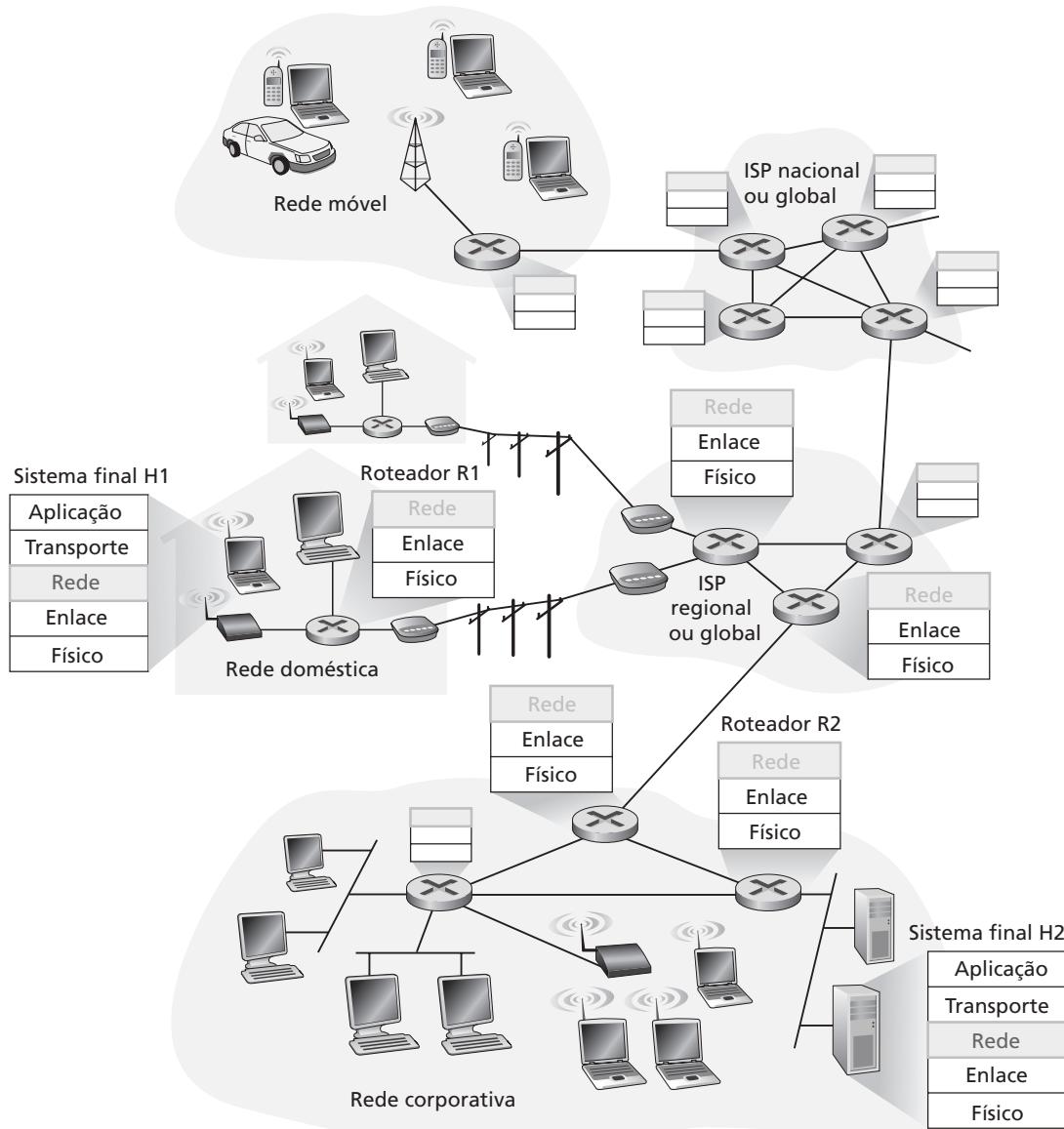


Figura 4.1 A camada de rede

Cada roteador tem uma **tabela de repasse**. Um roteador repassa um pacote examinando o valor de um campo no cabeçalho do pacote que está chegando e então utiliza esse valor para indexar sua tabela de repasse. O resultado da tabela de repasse indica para qual das interfaces de enlace do roteador o pacote deve ser repassado. Dependendo do protocolo de camada de rede, esse valor no cabeçalho do pacote pode ser o endereço de destino do pacote ou uma indicação da conexão à qual ele pertence. A Figura 4.2 dá um exemplo.

Na Figura 4.2, um pacote cujo valor no campo de cabeçalho é 0111 chega a um roteador. O roteador o indexa em sua tabela de repasse, determina que a interface de enlace de saída para esse pacote é a interface 2 e, então, repassa o pacote internamente à interface 2. Na Seção 4.3 examinaremos o interior de um roteador e estudaremos a função de repasse muito mais detalhadamente.

Agora você deve estar imaginando como são configuradas as tabelas de repasse nos roteadores. Essa é uma questão crucial, que expõe a importante interação entre roteamento e repasse. Como ilustrado na Figura 4.2, o algoritmo de roteamento determina os valores que são inseridos nas tabelas de repasse dos roteadores.

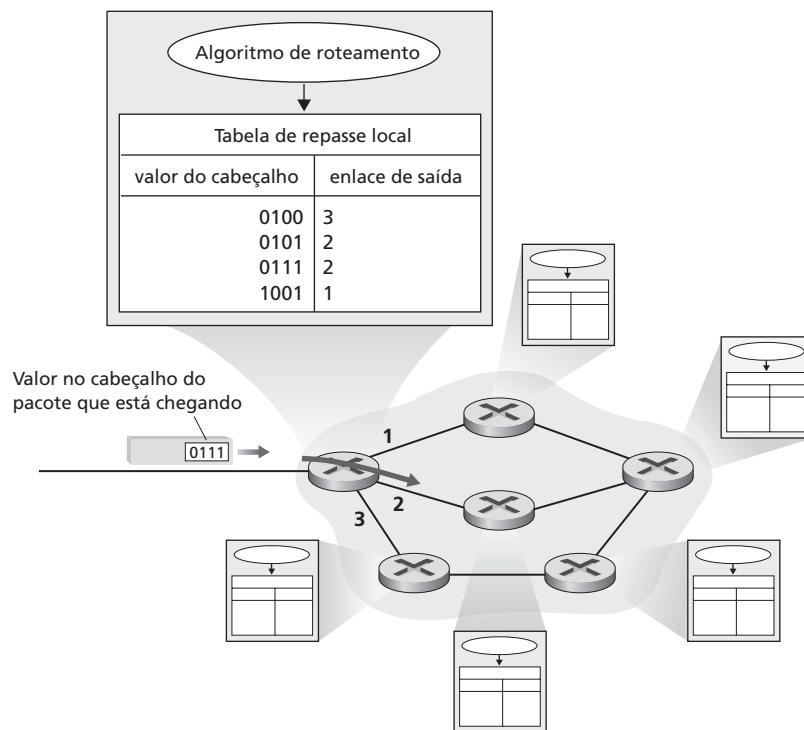


Figura 4.2 Algoritmos de roteamento determinam valores em tabelas de repasse

O algoritmo de roteamento pode ser centralizado (por exemplo, com um algoritmo que roda em um local central e descarrega informações de roteamento a cada um dos roteadores) ou descentralizado (isto é, com um pedaço do algoritmo de roteamento distribuído funcionando em cada roteador). Em qualquer dos casos, um roteador recebe mensagens de protocolo de roteamento que são utilizadas para configurar sua tabela de repasse. As finalidades distintas e diferentes das funções de repasse e roteamento podem ser mais bem esclarecidas considerando o caso hipotético (e não realista, mas tecnicamente viável) de uma rede na qual todas as tabelas de repasse são configuradas diretamente por operadores de rede humanos, fisicamente presentes nos roteadores. Nesse caso, não seria preciso *nenhum* protocolo de roteamento! É claro que os operadores humanos precisariam interagir uns com os outros para garantir que as tabelas fossem configuradas de tal modo que os pacotes chegassem a seus destinos pretendidos. Também é provável que uma configuração humana seria mais propensa a erro e muito mais lenta do que um protocolo de roteamento para reagir a mudanças na topologia da rede. Portanto, sorte nossa que todas as redes têm uma função de repasse e também uma função de roteamento!

Enquanto estamos no tópico da terminologia, é interessante mencionar dois outros termos que também são utilizados indistintamente, mas que usaremos com maior cuidado. Reservaremos o termo *comutador de pacotes* para designar um dispositivo geral de comutação de pacotes que transfere um pacote de interface de enlace de entrada para interface de enlace de saída conforme o valor que está em um campo no cabeçalho do pacote. Alguns comutadores de pacotes, denominados **comutadores de camada de enlace** (que veremos no Capítulo 5), baseiam a decisão de repasse no valor que está no campo da camada de enlace. Outros, denominados roteadores, baseiam sua decisão de repasse no valor que está no campo de camada de rede. (Para dar real valor a essa importante distinção, seria interessante você ler novamente a Seção 1.5.2, em que discutimos datagramas de camada de rede e quadros de camada de enlace e as relações entre eles.) Visto que o foco deste capítulo é a camada de rede, usaremos o termo *roteador* no lugar de *comutador de pacotes*. Usaremos o termo *roteador* até mesmo quando falarmos sobre comutadores de pacotes em redes de circuitos virtuais (que discutiremos em breve).

Estabelecimento de conexão

Acabamos de dizer que a camada de rede tem duas funções importantes, repasse e roteamento. Mas logo veremos que em algumas redes de computadores há uma terceira função importante de camada de rede, a saber, o **estabelecimento de conexão**. Lembre-se de que, quando estudamos o TCP, verificamos que é necessária uma apresentação de três vias antes de os dados realmente poderem fluir do remetente ao destinatário. Isso permite que o remetente e o destinatário estabeleçam a informação de estado necessária (por exemplo, número de sequência e tamanho inicial da janela de controle de fluxo). De maneira análoga, algumas arquiteturas de camada de rede — por exemplo, ATM, frame-relay, mas não a Internet — exigem que roteadores ao longo do caminho escolhido desde a fonte até o destino troquem mensagens entre si com a finalidade de estabelecer estado antes que pacotes de dados de camada de rede dentro de uma dada conexão fonte-destino possam começar a fluir. Na camada de rede, esse processo é denominado estabelecimento de conexão. Examinaremos estabelecimento de conexão na Seção 4.2.

4.1.2 Modelos de serviço de rede

Antes de examinar a camada de rede, vamos tomar uma perspectiva mais ampla e considerar os diferentes tipos de serviço que poderiam ser oferecidos pela camada de rede. Quando a camada de transporte em um hospedeiro remetente transmite um pacote para dentro da rede (isto é, passa o pacote para a camada de rede do hospedeiro remetente), ela pode contar com a camada de rede para entregar o pacote no destino? Quando são enviados vários pacotes, eles serão entregues à camada de transporte no hospedeiro destinatário na ordem em que foram enviados? A quantidade de tempo decorrido entre duas transmissões de pacotes sequenciais será a mesma quantidade de tempo decorrido entre suas recepções? A rede fornecerá algum tipo de informação sobre congestionamento na rede? Qual é o modelo (propriedades) abstrato do canal que conecta a camada de transporte nos hospedeiros remetente e destinatário? As respostas a essas e a outras perguntas são determinadas pelo modelo de serviço oferecido pela camada de rede. O **modelo de serviço de rede** define as características do transporte de dados fim a fim entre uma borda da rede e a outra, isto é, entre sistemas finais remetente e destinatário.

Vamos considerar agora alguns serviços possíveis que a camada de rede poderia prover. Quando a camada de transporte em um hospedeiro remetente passa um pacote para a camada de rede, alguns serviços específicos que poderiam ser oferecidos pela camada de rede são:

- *Entrega garantida.* Esse serviço assegura que o pacote mais cedo ou mais tarde chegará a seu destino.
- *Entrega garantida com atraso limitado.* Esse serviço não somente assegura a entrega de um pacote, mas também a entrega com um atraso hospedeiro a hospedeiro limitado e especificado (por exemplo, dentro de 100 milissegundos).

Além disso, há outros serviços que podem ser providos a um *fluxo de pacotes* entre uma fonte e um destino determinados, como os seguintes:

- *Entrega de pacotes na ordem.* Este serviço garante que pacotes chegarão ao destino na ordem em que foram enviados.
- *Largura de banda mínima garantida.* Esse serviço de camada de rede emula o comportamento de um enlace de transmissão com uma taxa de bits especificada (por exemplo, 1 Mbps) entre hospedeiros remetentes e destinatários (mesmo que o caminho fim a fim real passe por vários enlaces físicos). Contanto que o hospedeiro remetente transmita bits (como parte de pacotes) a uma taxa abaixo da taxa de bits especificada, nenhum pacote será perdido e cada um chegará dentro de um atraso hospedeiro a hospedeiro previamente especificado (por exemplo, dentro de 40 milissegundos).
- *Jitter máximo garantido.* Este serviço garante que a quantidade de tempo entre a transmissão de dois pacotes sucessivos no remetente seja igual à quantidade de tempo entre o recebimento dos dois pacotes no destino (ou que esse espaçamento não mude mais do que algum valor especificado).
- *Serviços de segurança.* Utilizando uma chave de sessão secreta conhecida somente por um computador-fonte e um computador-alvo, a camada de rede no computador-fonte pode codificar a carga útil de todos os datagramas que estão sendo enviados ao computador-alvo. A camada de rede no computador-

-alvo, então, seria responsável por decodificar as cargas úteis. Com esse serviço, o sigilo seria fornecido para todos os segmentos da camada de transporte (TCP e UDP) entre os computadores-fonte e os computadores-alvo. Além do sigilo, a camada de rede poderia prover integridade dos dados e serviços de autenticação de fonte.

Essa é uma lista apenas parcial de serviços que uma camada de rede pode prover — há incontáveis variações possíveis.

A camada de rede da Internet fornece um único modelo de serviço, conhecido como **serviço de melhor esforço**. Consultando a Tabela 4.1, pode parecer que *serviço de melhor esforço* seja um eufemismo para *absolutamente nenhum serviço*. Com o serviço de melhor esforço, não há garantia de que a temporização entre pacotes seja preservada, não há garantia de que os pacotes sejam recebidos na ordem em que foram enviados e não há garantia da entrega final dos pacotes transmitidos. Dada essa definição, uma rede que não entregasse *nenhum* pacote ao destinatário satisfaria a definição de serviço de entrega de melhor esforço. Contudo, como discutiremos adiante, há razões sólidas para esse modelo minimalista de serviço de camada de rede. Examinaremos outros modelos de serviço de Internet ainda em evolução no Capítulo 7.

Outras arquiteturas de rede definiram e implementaram modelos de serviço que vão além do serviço de melhor esforço da Internet. Por exemplo, a arquitetura de rede ATM [MFA Forum 2009; Black, 1995] habilita vários modelos de serviço, o que significa que, dentro da mesma rede, podem ser oferecidas conexões diferentes com classes de serviço diferentes. Discutir o modo como uma rede ATM provê esses serviços vai muito além do escopo deste livro; nossa meta aqui é apenas salientar que existem alternativas ao modelo de melhor esforço da Internet. Dois dos modelos mais importantes de serviço ATM são os serviços de taxa constante e de taxa disponível.

Serviço de rede de taxa constante de bits (*constant bit rate* — CBR). Esse foi o primeiro modelo de serviço ATM a ser padronizado, refletindo o interesse imediato das empresas de telefonia por esse serviço e a adequabilidade do serviço CBR para transmitir tráfego de áudio e vídeo de taxa constante de bits. O objetivo do serviço CBR é conceitualmente simples: prover um fluxo de pacotes (conhecidos como células na terminologia ATM) com uma tubulação virtual cujas propriedades são iguais às de um hipotético enlace de transmissão dedicado de largura de banda fixa entre os hospedeiros remetente e destinatário. Com serviço CBR, um fluxo de células ATM é carregado através da rede de modo tal que garanta que o atraso fim a fim de uma célula, a variabilidade desse atraso (isto é, o *jitter*) e a fração de células perdidas ou entregues atrasadas sejam menores do que valores especificados. Esses valores são acertados entre o hospedeiro remetente e a rede ATM quando a conexão CBR é estabelecida pela primeira vez.

Serviço de rede de taxa de bits disponível (*available bit rate* — ABR). Como o serviço oferecido pela Internet é “de melhor esforço”, o ATM ABR pode ser caracterizado como um “serviço de melhor esforço ligeiramente melhorado”. Como acontece com o modelo de serviço da Internet, também com o serviço ABR pode haver perda de células. Mas, diferentemente da Internet, as células não podem ser reordenadas (embora possam ser perdidas) e é garantida uma taxa mínima de transmissão de células (*minimum cell transmission rate* — MCR) para uma conexão que está usando o serviço ABR. Se, contudo, a rede dispuser de suficientes recursos livres em um dado momento, um remetente também poderá enviar células com sucesso a uma taxa mais alta do que a MCR.

Arquitetura da rede	Modelo de serviço	Garantia de largura de banda	Garantia contra perda	Ordenamento	Temporização	Indicação de congestionamento
Internet	Melhor esforço	Nenhuma	Nenhuma	Qualquer ordem possível	Não mantida	Nenhuma
ATM	CBR	Taxa constante garantida	Sim	Na ordem	Mantida	Não ocorrerá congestionamento
ATM	ABR	Mínima garantida	Nenhuma	Na ordem	Não mantida	Indicação de congestionamento

Tabela 4.1 Modelos de serviço das redes Internet, ATM CBR e ATM ABR

Além disso, como vimos na Seção 3.6, o serviço ATM ABR pode prover realimentação ao remetente (em termos de um bit de notificação de congestionamento ou de uma taxa de envio explícita), que controla o modo como o remetente ajusta sua taxa entre a MCR e uma taxa de pico admissível.

4.2 Redes de circuitos virtuais e de datagramas

Lembre-se de que, no Capítulo 3, dissemos que a camada de transporte pode oferecer às aplicações serviço não orientado para conexão ou serviço orientado para conexão. Por exemplo, a camada de transporte da Internet oferece a cada aplicação uma alternativa entre dois serviços: UDP, um serviço não orientado para conexão; ou TCP, um serviço orientado para conexão. De modo semelhante, uma camada de rede também pode oferecer qualquer desses dois serviços. Serviços de camada de rede orientados para conexão e não orientados para conexão são, sob muitos aspectos, semelhantes a esses mesmos serviços providos pela camada de transporte. Por exemplo, um serviço de camada de rede orientado para conexão começa com uma apresentação entre os hospedeiros de origem e de destino; e um serviço de camada de rede não orientado para conexão não tem nenhuma apresentação preliminar.

Embora os serviços de camada de rede orientados para conexão e não orientados para conexão tenham algumas semelhanças com os mesmos serviços oferecidos pela camada de transporte, há diferenças cruciais:

- Na camada de rede, são serviços de hospedeiro a hospedeiro provados pela camada de rede à camada de transporte. Na camada de transporte, são serviços de processo a processo fornecidos pela camada de transporte à camada de aplicação.
- Em todas as arquiteturas importantes de redes de computadores existentes até agora (Internet, ATM, frame-relay e assim por diante), a camada de rede oferece um serviço entre hospedeiros não orientado para conexão, ou um serviço entre hospedeiros orientado para conexão, mas não ambos. Redes de computadores que oferecem apenas um serviço orientado para conexão na camada de rede são denominadas **redes de circuitos virtuais (redes CV)**; redes de computadores que oferecem apenas um serviço não orientado para conexão na camada de rede são denominadas **redes de datagramas**.
- As implementações de serviço orientado para conexão na camada de transporte e de serviço de conexão na camada de rede são fundamentalmente diferentes. No capítulo anterior vimos que o serviço de camada de transporte orientado para conexão é implementado na borda da rede nos sistemas finais; em breve veremos que o serviço da camada de rede orientado para conexão é implementado nos roteadores no núcleo da rede, bem como nos sistemas finais.

Redes de circuitos virtuais e redes de datagramas são duas classes fundamentais de redes de computadores. Elas utilizam informações muito diferentes para tomar suas decisões de repasse. Vamos agora examinar suas implementações mais de perto.

4.2.1 Redes de circuitos virtuais

Já vimos que a Internet é uma rede de datagramas. Entretanto, muitas arquiteturas de rede alternativas — entre elas as das redes ATM e frame relay — são redes de circuitos virtuais e, portanto, usam conexões na camada de rede. Essas conexões de camada de rede são denominadas **circuitos virtuais (CVs)**. Vamos considerar agora como um serviço de CVs pode ser implantado em uma rede de computadores.

Um circuito virtual (CV) consiste em (1) um caminho (isto é, uma série de enlaces e roteadores) entre hospedeiros de origem e de destino, (2) números de CVs, um número para cada enlace ao longo do caminho e (3) registros na tabela de repasse em cada roteador ao longo do caminho. Um pacote que pertence a um circuito virtual portará um número de CV em seu cabeçalho. Como um circuito virtual pode ter um número de CV diferente em cada enlace, cada roteador interveniente deve substituir o número de CV de cada pacote em trânsito por um novo número. Esse número novo do CV é obtido da tabela de repasse.

Para ilustrar o conceito, considere a rede mostrada na Figura 4.3. Os números ao lado dos enlaces de R1 na Figura 4.3 são os números das interfaces de enlaces. Suponha agora que o hospedeiro A solicite à rede que

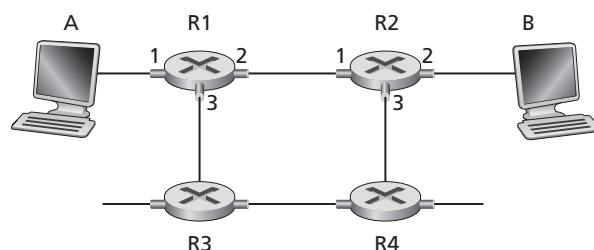


Figura 4.3 Uma rede de circuitos virtuais simples

estabeleça um CV entre ele e o hospedeiro B. Suponha ainda que a rede escolha o caminho A–R1–R2–B e atribua números de CV 12, 22, 32 aos três enlaces nesse caminho para esse circuito virtual. Nesse caso, quando um pacote nesse CV sai do hospedeiro A, o valor no campo do número de CV é 12; quando sai de R1, o valor é 22, e, quando sai de R2, o valor é 32.

Como o roteador determina o novo número de CV para um pacote que passa por ele? Para uma rede de CV, a tabela de repasse de cada roteador inclui a tradução de número de CV; por exemplo, a tabela de repasse de R1 pode ser algo parecido com o seguinte:

Interface de entrada	Nº do CV de entrada	Interface de saída	Nº do CV de saída
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...

Sempre que um novo CV é estabelecido através de um roteador, um registro é adicionado à tabela de repasse. De maneira semelhante, sempre que um CV termina, são removidos os registros apropriados em cada tabela ao longo de seu caminho.

É bem possível que você esteja pensando por que um pacote não conserva o mesmo número de CV em cada um dos enlaces ao longo de sua rota. Isso ocorre por dois motivos: primeiro, substituir o número de enlace em enlace reduz o comprimento do campo do CV no cabeçalho do pacote. Segundo, e mais importante, o estabelecimento de um CV é consideravelmente simplificado se for permitido um número diferente de CV em cada enlace ao longo do caminho do circuito virtual. Especificamente, com vários números de CV, cada enlace do caminho pode escolher um número de CV independentemente dos números de CV escolhidos em outros enlaces ao longo do caminho. Se fosse exigido um mesmo número de CV para todos os enlaces ao longo do caminho, os roteadores teriam de trocar e processar um número substancial de mensagens para escolher o número de CV a ser usado para uma conexão (por exemplo, um número que não está sendo usado por nenhum outro CV nesses roteadores).

Em uma rede de circuitos virtuais, os roteadores da rede devem manter **informação de estado de conexão** para as conexões em curso. Especificamente, cada vez que uma nova conexão for estabelecida através de um roteador, um novo registro de conexão deve ser adicionado à tabela de repasse do roteador. E, sempre que uma conexão for desativada, um registro deve ser removido da tabela. Observe que, mesmo que não haja tradução de números de CVs, ainda assim é necessário manter informação de estado de conexão que associe números de CVs com números das interfaces de saída. A questão de um roteador manter ou não informação de estado de conexão para cada conexão em curso é crucial — e retornaremos a ela várias vezes neste livro.

Há três fases que podem ser identificadas em um circuito virtual:

Estabelecimento de CV. Durante a fase de estabelecimento, a camada de transporte remetente contata a camada de rede, especifica o endereço do receptor e espera até a rede estabelecer o CV. A camada de

rede determina o caminho entre remetente e destinatário, ou seja, a série de enlaces e roteadores pelos quais todos os pacotes do CV trafegarão. A camada de rede também determina o número de CV para cada enlace ao longo do caminho e, finalmente, adiciona um registro na tabela de repasse em cada roteador ao longo do caminho. Durante o estabelecimento do CV, a camada de rede pode também reservar recursos (por exemplo, largura de banda) ao longo do caminho do CV.

Transferência de dados. Como mostra a Figura 4.4, tão logo estabelecido o CV, pacotes podem começar a fluir ao longo do CV.

Encerramento do CV. O encerramento começa quando o remetente (ou o destinatário) informa à camada de rede seu desejo de desativar o CV. A camada de rede então tipicamente informará o sistema final do outro lado da rede de término de conexão e atualizará as tabelas de repasse em cada um dos roteadores de pacotes no caminho para indicar que o CV não existe mais.

Há uma distinção sutil, mas importante, entre estabelecimento de CV na camada de rede e estabelecimento de conexão na camada de transporte (por exemplo, a apresentação TCP de três vias que estudamos no Capítulo 3). Estabelecer conexão na camada de transporte envolve apenas os dois sistemas finais. Durante o estabelecimento da conexão na camada de transporte, os dois sistemas finais determinam os parâmetros (por exemplo, número de sequência inicial e tamanho da janela de controle de fluxo) de sua conexão de camada de transporte. Embora os dois sistemas finais fiquem cientes da conexão de camada de transporte, os roteadores dentro da rede ficam completamente alheios a ela. Por outro lado, com uma camada de rede de CV, os roteadores *ao longo do caminho entre os dois sistemas finais estão envolvidos no estabelecimento de CV e cada roteador fica totalmente ciente de todos os CVs que passam por ele*.

As mensagens que os sistemas finais enviam à rede para iniciar ou encerrar um CV e as mensagens passadas entre os roteadores para estabelecer o CV (isto é, modificar estado de conexão em tabelas de roteadores) são conhecidas como **mensagens de sinalização** e os protocolos usados para trocar essas mensagens frequentemente são denominados **protocolos de sinalização**. O estabelecimento de CV está ilustrado na Figura 4.4. Não abordaremos protocolos de sinalização de CVs neste livro; [Black, 1997] apresenta uma discussão geral sobre sinalização em redes orientadas para conexão e [ITU-T Q.2931, 1994] mostra a especificação do protocolo de sinalização Q.2931 do ATM.

4.2.2 Redes de datagramas

Em uma **rede de datagramas**, toda vez que um sistema final quer enviar um pacote, ele marca o pacote com o endereço do sistema final de destino e então o envia para dentro da rede. Como mostra a Figura 4.5, isso é feito sem o estabelecimento de nenhum CV. Roteadores em uma rede de datagramas não mantêm nenhuma informação de estado sobre CVs (porque não há nenhum CV!).

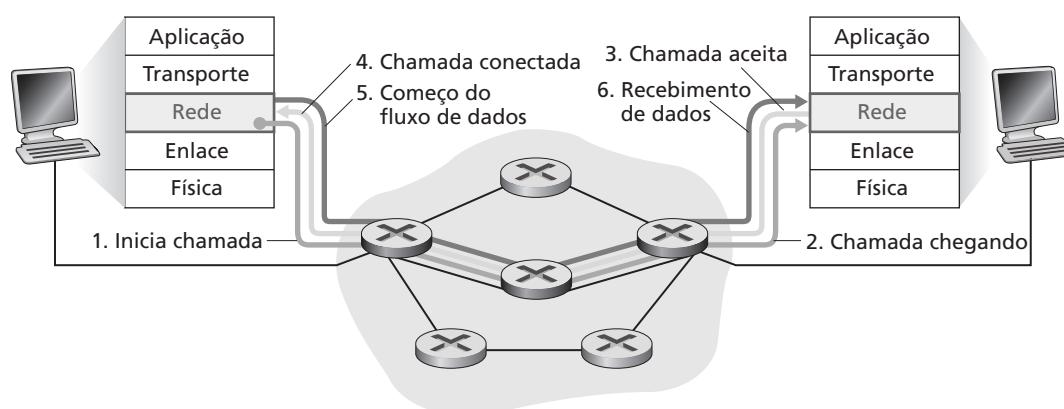


Figura 4.4 Estabelecimento de circuito virtual

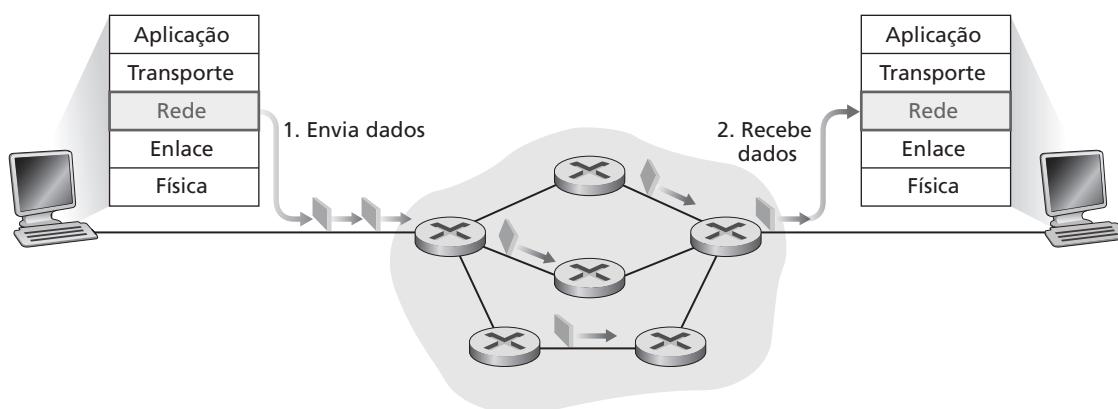


Figura 4.5 Rede de datagramas

Ao ser transmitido da fonte ao destino, um pacote passa por uma série de roteadores. Cada um desses roteadores usa o endereço de destino do pacote para repassá-lo. Especificamente, cada roteador tem uma tabela de repasse que mapeia endereços de destino para interfaces de enlaces; quando um pacote chega ao roteador, este usa o endereço de destino do pacote para procurar a interface de enlace de saída apropriada na tabela de repasse. Então, o roteador transmite o pacote para aquela interface de enlace de saída.

Para entender melhor a operação de consulta, vamos examinar um exemplo específico. Suponha que todos os endereços de destino tenham 32 bits (que, por acaso, é exatamente o comprimento do endereço de destino em um datagrama IP). Uma implementação de força bruta da tabela de repasse teria um registro para cada endereço de destino possível. Porém, como há mais de quatro bilhões de endereços possíveis, essa opção está totalmente fora de questão — exigiria uma tabela de repasse imensa.

Vamos supor ainda mais que nosso roteador tenha quatro enlaces numerados de 0 a 3, e que os pacotes devem ser repassados para as interfaces de enlace como mostrado a seguir:

Faixa de Endereços de Destino	Interface de enlace
11001000 00010111 00010000 00000000	
até	0
11001000 00010111 00010111 11111111	
11001000 00010111 00011000 00000000	
até	1
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000	
até	2
11001000 00010111 00011111 11111111	
senão	3

Fica claro, por este exemplo, que não é necessário ter quatro bilhões de registros na tabela de repasse do roteador. Poderíamos, por exemplo, ter a seguinte tabela de repasse com apenas quatro registros:

Prefixo do endereço	Interface de enlace
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
senão	3

Com esse tipo de tabela de repasse, o roteador compara um **prefixo** do endereço de destino do pacote com os registros na tabela; se houver uma concordância de prefixos, o roteador transmite o pacote para o enlace associado àquele prefixo compatível. Por exemplo, suponha que o endereço de destino do pacote seja 11001000 00010111 00010110 10100001; como o prefixo de 21 bits desse endereço é igual ao primeiro registro na tabela, o roteador transmite o pacote para a interface de enlace 0. Se o prefixo do pacote não combinar com nenhum dos três primeiros registros, o roteador envia o pacote para a interface 3. Embora isso pareça bastante simples, há aqui uma sutileza importante. Você talvez tenha notado que existe a possibilidade de um endereço de destino combinar com mais de um registro. Por exemplo, os primeiros 24 bits do endereço 11001000 00010111 00011000 10101010 combinam com o segundo registro na tabela e os primeiros 21 bits do endereço combinam com o terceiro registro da tabela. Quando há várias concordâncias de prefixos, o roteador usa a **regra da concordância do prefixo mais longo**, isto é, encontra o registro cujo prefixo tem mais bits compatíveis com os bits do endereço do pacote e envia o pacote à interface de enlace associada com esse prefixo mais longo compatível. Veremos exatamente por que essa regra de prefixo mais longo compatível é utilizada quando estudarmos endereçamento da Internet em mais detalhes da Seção 4.4.

Embora em redes de datagramas os roteadores não mantenham nenhuma informação de estado de conexão, ainda assim mantêm informação de estado de repasse em suas tabelas de repasse. Todavia, a escala temporal da mudança dessas informações de estado é relativamente lenta. Na verdade, as tabelas de repasse em uma rede de datagramas são modificadas pelos algoritmos de roteamento que normalmente atualizam uma tabela de repasse em intervalos de um a cinco minutos, aproximadamente. A tabela de repasse de um roteador em uma rede de CVs é modificada sempre que é estabelecida uma nova conexão através do roteador ou sempre que uma conexão existente é desativada. Em um roteador de backbone de nível 1, isso poderia acontecer facilmente em uma escala temporal de microssegundos.

Como em redes de datagramas as tabelas de repasse podem ser modificadas a qualquer momento, uma série de pacotes enviados de um sistema final para outro pode seguir caminhos diferentes através da rede e muitos podem chegar fora da ordem. [Paxson, 1997] e [Jaiswal, 2003] apresentam interessantes estudos de medição da reordenação de pacotes e de outros fenômenos na Internet pública.

4.2.3 Origens das redes de circuitos virtuais e de datagramas

A evolução das redes de datagramas e de circuitos virtuais reflete as origens dessas redes. A ideia de um circuito virtual como princípio fundamental de organização tem suas raízes no mundo da telefonia (que usa circuitos reais). Como em redes de CV os roteadores mantêm estado de chamada e estado por chamada, esse tipo de rede é consideravelmente mais complexo do que uma rede de datagramas. ([Molinero-Fernandez, 2002] faz uma comparação interessante entre as complexidades de redes de comutação de circuitos e de comutação de pacotes.) Isso também está de acordo com a herança da telefonia. A complexidade das redes telefônicas estava, necessariamente, dentro da própria rede, já que elas conectavam sistemas finais não inteligentes tais como telefones de disco. (Para os mais jovens que não o conhecem, um telefone de disco é um telefone analógico [isto é, não digital] sem teclas —, tem somente um mostrador com números e um dispositivo mecânico denominado disco.)

Por sua vez, a Internet como uma rede de datagramas surgiu da necessidade de conectar computadores. Como esses sistemas finais são mais sofisticados, os arquitetos da Internet preferiram construir um modelo de serviço de camada de rede o mais simples possível. Como já vimos nos Capítulos 2 e 3, funcionalidades adicionais (por exemplo, entrega na ordem, transferência confiável de dados, controle de congestionamento e resolução de nomes DNS) são implementadas em uma camada mais alta, nos sistemas finais. Isso inverte o modelo da rede de telefonia, com algumas consequências interessantes:

Visto que o modelo de serviço de camada de rede resultante da Internet, que oferece garantias mínimas (nenhuma!) de serviço, impõe exigências mínimas sobre a camada de rede. Isso facilita a interconexão de redes que usam tecnologias de camada de enlace muito diferentes (por exemplo, satélite, Ethernet, fibra ou rádio) e cujas taxas de transmissão e características de perda também são muito diferentes. Vamos abordar detalhadamente a interconexão de redes IP na Seção 4.4.

Como vimos no Capítulo 2, aplicações como e-mail, a Web e até mesmo um serviço centrado na camada de rede como o DNS são implementadas em hospedeiros (servidores) na borda da rede. A capacidade de adicionar um novo serviço simplesmente ligando um hospedeiro à rede e definindo um novo protocolo de camada de aplicação (como o HTTP) permitiu que novas aplicações como a Web fossem distribuídas pela Internet em um período notavelmente curto.

Como veremos no Capítulo 7, há uma considerável discussão na comunidade da Internet sobre como a arquitetura da camada de rede da Internet deve evoluir para suportar serviços de tempo real como multimídia. Uma comparação interessante entre a arquitetura de rede ATM orientada para CVs e a próxima geração proposta para a arquitetura da Internet é dada em [Crowcroft, 1995].

4.3 O que há dentro de um roteador?

Agora que já tivemos uma visão geral das funções e serviços da camada de rede, voltaremos nossa atenção para a função de repasse — a transferência, propriamente dita, de pacotes dos enlaces de entrada até os enlaces de saída adequados de um roteador. Já estudamos algumas questões de repasse na Seção 4.2, a saber, endereçamento e compatibilização com o prefixo mais longo. Nesta seção examinaremos arquiteturas específicas de roteadores para transferir pacotes de enlaces de entrada para enlaces de saída. Neste caso, nossa análise é necessariamente breve, pois seria necessário um curso completo para tratar do projeto de roteadores com profundidade. Consequentemente, faremos um esforço especial nesta seção para indicar material que apresente esse tópico com mais profundidade. Mencionamos, de passagem, que os pesquisadores e profissionais de redes de computadores usam as palavras *repasse* e *comutação* indistintamente; nós usaremos ambos os termos neste livro.

Uma visão de alto nível da arquitetura de um roteador genérico é mostrada na Figura 4.6. Quatro componentes de um roteador podem ser identificados:

Portas de entrada. A porta de entrada tem diversas funções. Ela realiza as funções de camada física (a caixa mais à esquerda da porta de entrada e a caixa mais à direita da porta de saída na Figura 4.6) de terminar um enlace físico de entrada em um roteador. Realiza também as funções de camada de enlace (representadas pelas caixas do meio nas portas de entrada e de saída) necessárias para interoperar com as funções da camada de enlace do outro lado do enlace de entrada. Realiza ainda uma função de exame e de repasse (a caixa mais à direita da porta de entrada e a caixa mais à esquerda da porta de saída), de modo que um pacote repassado ao elemento de comutação do roteador surja na porta de saída apropriada. Pacotes de controle (por exemplo, pacotes carregando informações de protocolo de roteamento) são repassados de uma porta de entrada até o processador de roteamento. Na prática, várias portas são frequentemente reunidas em uma única **placa de linha** no interior de um roteador.

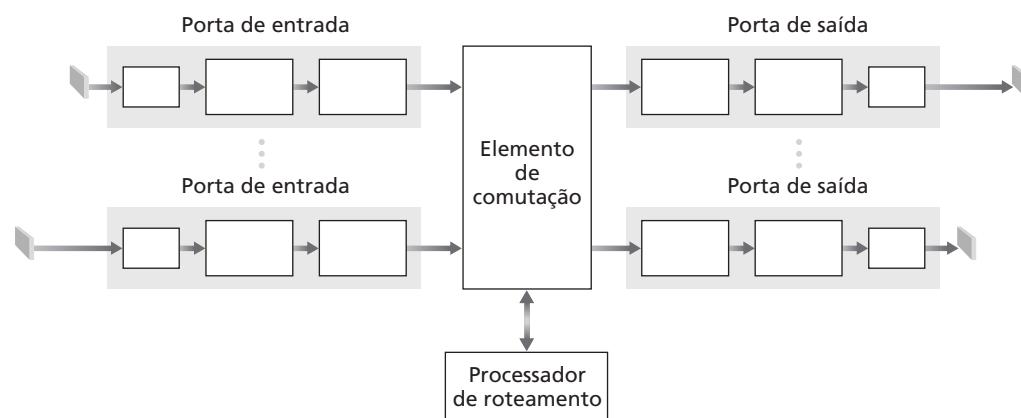


Figura 4.6 Arquitetura de roteador

Elemento de comutação. O elemento de comutação conecta as portas de entrada do roteador às suas portas de saída. Ele está integralmente contido no interior do roteador — uma rede dentro de um roteador de rede!

Portas de saída. Uma porta de saída armazena os pacotes que foram repassados a ela através do elemento de comutação e, então, os transmite até o enlace de saída. Assim, a porta de saída realiza o inverso da funcionalidade da camada de enlace e da camada física da porta de entrada. Quando um enlace é bidirecional (isto é, carrega um tráfego em ambas as direções), uma porta de saída para o enlace será tipicamente emparelhada com a porta de entrada para aquele enlace na mesma placa de linha.

Processador de roteamento. O processador de roteamento roda os protocolos de roteamento (por exemplo, os protocolos que estudamos na Seção 4.6), mantém as informações de roteamento e tabelas de repasse e executa funções de gerenciamento de rede (veja o Capítulo 9) dentro do roteador.

Nas subseções seguintes, examinaremos com maiores detalhes portas de entrada, o elemento de comutação e portas de saída. [Chuang, 2005; Keslasy, 2003; Chao, 2001; Turner, 1988; Giacopelli, 1990; McKeown, 1997a; Partridge, 1998] fornecem uma análise de algumas arquiteturas específicas de roteadores. [McKeown, 1997b] apresenta uma visão geral particularmente fácil de ler de modernas arquiteturas de roteadores usando o roteador Cisco 12000 como exemplo. Para manter a discussão em terreno concreto, admitimos que a rede de computadores é uma rede de pacotes e que decisões de repasse se baseiam nos endereços de destino dos pacotes (e não em um número de CV em uma rede de circuitos virtuais). Todavia, os conceitos e técnicas são similares aos de uma rede de circuitos virtuais.

4.3.1 Portas de entrada

Uma visão mais detalhada da funcionalidade de porta de entrada é apresentada na Figura 4.7. Como discutido anteriormente, as funções de terminação de linha e de processamento de enlace realizadas pela porta de entrada implementam as camadas física e de enlace associadas a um enlace de entrada individual do roteador. O módulo examinar/repassar da porta de entrada é fundamental para a função de repasse do roteador. Em muitos roteadores, é aqui que o roteador determina a porta de saída para a qual o pacote que está chegando será repassado pelo elemento de comutação. A escolha da porta de saída é feita usando a informação contida na tabela de repasse. Embora a tabela de repasse seja calculada pelo processador de roteamento, uma cópia da tabela é comumente armazenada em cada porta de entrada e atualizada, quando necessário, pelo processador de roteamento. Com cópias locais da tabela de repasse, as decisões de repasse podem ser tomadas localmente, em cada porta de entrada, sem chamar o processador de roteamento centralizado. Esse repasse *descentralizado* evita a criação de um gargalo de processamento de repasse em um único ponto no interior do roteador.

Em roteadores com capacidade limitada de processamento na porta de entrada, a porta pode simplesmente repassar o pacote para o processador de roteamento centralizado, que, então, realizará o exame da tabela de repasse e transmitirá o pacote à porta de saída apropriada. Esta é a abordagem adotada quando uma estação de trabalho ou um servidor funciona como um roteador. Aqui, o processador de roteamento é, na realidade, apenas a CPU da estação de trabalho, enquanto a porta de entrada é, na verdade, apenas uma placa de interface de rede (por exemplo, uma placa Ethernet).

Dada a existência de uma tabela de repasse, o exame é conceitualmente simples — basta procurar o registro mais longo compatível com o endereço de destino, como descrito na Seção 4.2.2. Porém, na prática, as coisas não

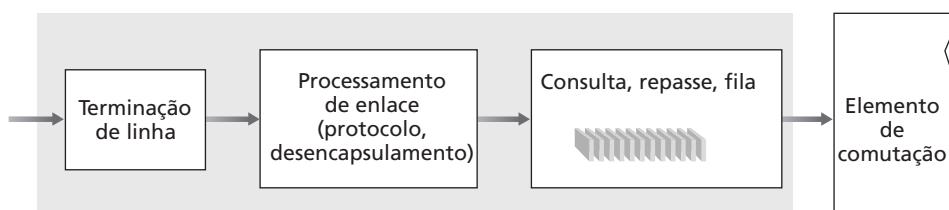


Figura 4.7 Processamento na porta de entrada



História

Cisco Systems: dominando o núcleo da rede

Até outubro de 2008, a Cisco empregava mais de 65 mil pessoas. A empresa dominava o mercado de roteadores da Internet e, nos últimos anos, voltou-se para o mercado de telefonia por Internet, em que concorre de igual para igual com empresas fabricantes de equipamentos de telefonia como Lucent, Alcatel, Nortel e Siemens. Como surgiu essa gigantesca empresa de rede? Tudo começou em 1984 na sala de estar de um apartamento no Vale do Silício. Len Bosak e sua esposa Sandy Lerner trabalhavam na universidade de Stanford quando tiveram a ideia de construir e vender roteadores de Internet a instituições acadêmicas e de pesquisa. Sandy Lerner teve a ideia do nome "Cisco" (uma abreviação de San Francisco) e também criou o logotipo da empresa, que é uma ponte. A sede da empresa era originalmente a sala de estar do casal que, no início, financiou o projeto com cartões de crédito e trabalhos extras. No final de 1986, a Cisco conseguiu atrair capital de risco — dois milhões de dólares da Sequoia Capital em troca de um terço da empresa. Nos anos seguintes, a Cisco continuou a crescer e a conquistar cada vez mais participação de mercado. Ao mesmo tempo, o relacionamento entre Bosak/Lerner e a administração da empresa começou a ficar tenso. A Cisco abriu seu capital em 1990 e, nesse mesmo ano, Lerner e Bosak saíram da empresa.

Com o passar dos anos, a Cisco expandiu amplamente seu mercado de roteadores, vendendo produtos voltados à segurança, redes sem fio, com tecnologia Voz sobre IP e serviços. Entretanto, a empresa está enfrentando concorrência internacional crescente, incluindo a empresa chinesa em crescimento Huawei, que fornece equipamentos de rede. Outras fontes de concorrência para a Cisco no mercado de roteadores e Ethernet comutada incluem a Alcatel-Lucent e a Juniper.

são assim tão simples. Talvez o principal fator de complicaçāo seja o fato de os roteadores de backbone terem de operar em altas velocidades, rodando milhões de exames por segundo. Na verdade, é desejável que o processamento da porta de entrada tenha a capacidade de operar à **velocidade da linha**, isto é, que o exame possa ser feito em tempo menor do que o necessário para receber um pacote na porta de entrada. Nesse caso, o processamento de entrada de um pacote recebido pode ser concluído antes do término da operação de recebimento subsequente. Para ter uma ideia das exigências de desempenho para uma consulta, considere que um enlace do tipo OC48 opere à velocidade de 2,5 Gbps. Com pacotes de comprimento de 256 bytes, isso implica uma taxa de aproximadamente um milhão de consultas por segundo.

Devido à necessidade de operar nas altas velocidades de enlace atuais, é impossível uma busca linear em uma tabela de repasse extensa. Uma técnica mais razoável é armazenar os registros da tabela de repasse em uma estrutura de árvore de dados. Cada nível da árvore pode ser imaginado como um bit no endereço de destino. Para examinar um endereço, basta simplesmente começar no nó da raiz da árvore. Se o primeiro bit do endereço for zero, então a subárvore da esquerda conterá o registro da tabela de repasse para o endereço de destino; caso contrário, ele estará na subárvore da direita. A subárvore apropriada é, então, percorrida usando os bits de endereço remanescentes — se o próximo bit do endereço for zero, será escolhida a subárvore da esquerda da subárvore inicial; caso contrário, será escolhida a subárvore da direita. Dessa maneira, podemos examinar o registro da tabela de repasse em N etapas, em que N é o número de bits do endereço. (Note que isso é, essencialmente, uma busca binária em um espaço de endereço de tamanho 2^N .) Um melhoramento das técnicas de busca binária é descrito por [Srinivasan, 1999], e um levantamento geral de algoritmos de classificação de pacotes pode ser encontrado em [Gupta, 2001].

Mas, mesmo quando $N = 32$ etapas (por exemplo, um endereço IP de 32 bits), a velocidade de consulta pela busca binária não é suficientemente rápida para os requisitos atuais de roteamento de backbone. Por exemplo, admitindo um acesso à memória a cada etapa, menos de um milhão de consultas de endereços por segundo podem ser realizadas com tempos de acesso à memória de 40 nanosegundos. Assim, diversas técnicas têm sido pesquisadas para aumentar as velocidades de consulta. **Memórias de conteúdo endereçável** (*content addressable*

memories — CAMs) permitem que um endereço IP de 32 bits seja apresentado à CAM, que então devolve o conteúdo do registro da tabela de repasse para o endereço em tempo essencialmente constante. O roteador da série 8500 da Cisco tem uma CAM de 64K para cada porta de entrada.

Outra técnica para acelerar a consulta é manter registros de tabela de repasse acessados recentemente em armazenamento intermediário (cache) [Feldmeier, 1988]. Nesse caso, a preocupação é o tamanho potencial do cache. Foram propostas estruturas de dados velozes, que permitem que os registros da tabela de repasse sejam localizados em $\log(N)$ etapas [Waldvogel, 1997] ou que comprimem tabelas de repasse de maneiras originais [Brodnik, 1997]. Uma abordagem da consulta baseada em hardware e otimizada para o caso corriqueiro em que o endereço que está sendo examinado tem 24 bits significativos ou menos é discutida por [Gupta, 1998]. Para uma análise e taxonomia de algoritmos de busca de endereço IP de alta velocidade, consulte [Ruiz-Sánchez, 2001].

Assim que a porta de saída para um pacote é determinada por meio da consulta, o pacote pode ser repassado para o elemento de comutação. Contudo, um pacote pode ser temporariamente impedido (bloqueado) de entrar no elemento de comutação (por haver pacotes vindos de outras portas de entrada usando o elemento naquele momento). Assim, um pacote bloqueado deve entrar na fila da porta de entrada e então ser programado para atravessar o elemento de comutação mais tarde. Vamos examinar com mais detalhes o bloqueio, a formação de fila e o escalonamento de pacotes (nas portas de entrada e de saída) dentro de um roteador na Seção 4.3.4.

4.3.2 Elemento de comutação

O elemento de comutação está no coração de um roteador. É por meio do elemento de comutação que os pacotes são comutados (isto é, repassados) de uma porta de entrada para uma porta de saída. A comutação pode ser realizada de inúmeras maneiras, como mostra a Figura 4.8.

Comutação por memória. Os primeiros e mais simples roteadores quase sempre eram computadores tradicionais nos quais a comutação entre as portas de entrada e de saída era realizada sob o controle direto da CPU (processador de roteamento). As portas de entrada e de saída funcionavam como dispositivos tradicionais de entrada/saída de um sistema operacional tradicional. Uma porta de entrada na

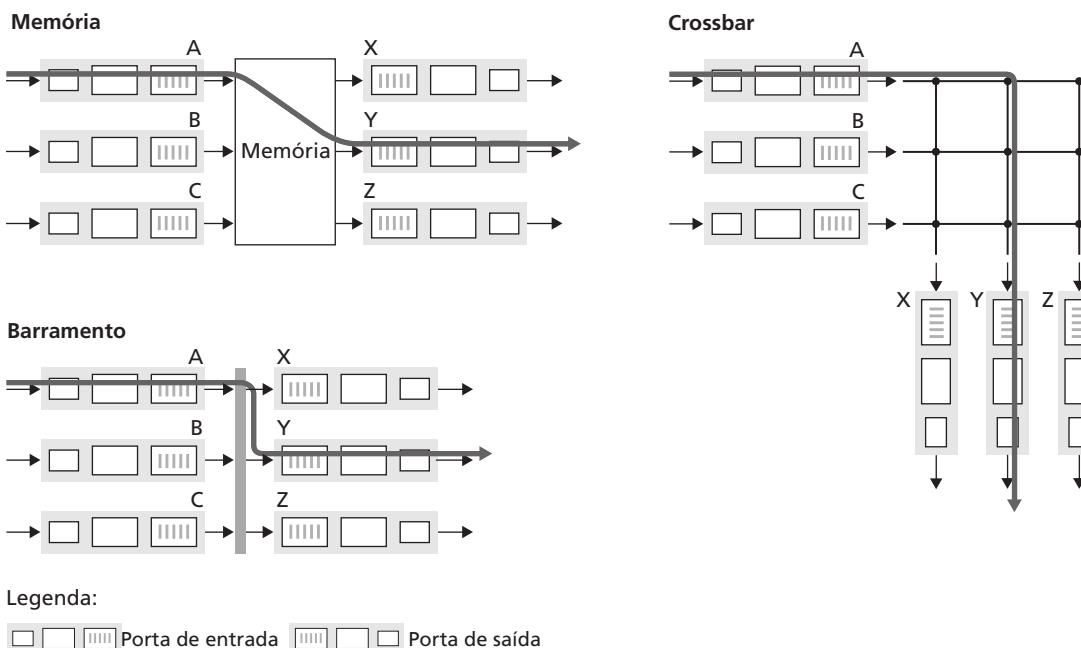


Figura 4.8 Três técnicas de comutação

qual um pacote estivesse entrando primeiramente sinalizaria ao processador de roteamento por meio de uma interrupção. O pacote era então copiado da porta de entrada para a memória do processador. O processador de roteamento então extraía o endereço de destino do cabeçalho, consultava a porta de saída apropriada na tabela de repasse e copiava o pacote para os buffers da porta de saída. Note que, se a largura de banda da memória for tal que B pacotes/segundo possam ser escritos ou lidos na memória, então a vazão total de repasse (a velocidade total com que os pacotes são transferidos de portas de entrada para portas de saída) deverá ser menor do que $B/2$.

Muitos roteadores modernos também comutam por memória. Contudo, uma importante diferença entre esses roteadores e os antigos é que a consulta do endereço de destino e o armazenamento do pacote na localização adequada da memória são realizados por processadores nas placas de linha de entrada. Em certos aspectos, roteadores que comutam por memória se parecem muito com multiprocessadores de memória compartilhada, nos quais os processadores de uma placa de linha comutam pacotes para a memória da porta de saída adequada. Os comutadores série 8500 Catalyst da Cisco [Cisco 8500, 2009] comutam pacotes por uma memória compartilhada. Um modelo abstrato para estudar as propriedades da comutação por memória e uma comparação com outras formas de comutação podem ser encontrados em [Iyer, 2002].

Comutação por um barramento. Nessa abordagem, as portas de entrada transferem um pacote diretamente para a porta de saída por um barramento compartilhado sem a intervenção do processador de roteamento (note que na comutação por memória o pacote também deve atravessar o barramento do sistema para vir da memória e ir para ela.) Embora o processador de roteamento não esteja envolvido na transferência por barramento, como o barramento é compartilhado, somente um pacote por vez pode ser transferido por meio do barramento. Um pacote que chega a uma porta de entrada e encontra o barramento ocupado com a transferência de outro pacote tem bloqueada sua passagem pelo elemento de comutação e é colocado na fila da porta de entrada. Como cada pacote deve atravessar o único barramento, a largura de banda de comutação do roteador fica limitada à velocidade do barramento.

Dado que larguras de banda de barramento acima de 1 Gbps são possíveis com a tecnologia atual, a comutação por barramento muitas vezes é suficiente para roteadores que operam em redes de acesso e redes de empresas (por exemplo, redes locais [LANs] e redes corporativas). A comutação por barramento foi adotada em uma série de produtos de roteamento atuais, entre eles o Cisco 5600 [Cisco Switches, 2009], que comuta pacotes por um barramento da placa-mãe (*backplane bus*) de 32 Gbps.

Comutação por uma rede de interconexão. Um modo de vencer a limitação da largura de banda de um barramento único compartilhado é usar uma rede de interconexão mais sofisticada, tal como as que eram utilizadas no passado para interconectar processadores em uma arquitetura de computadores multiprocessadores. Um comutador do tipo crossbar é uma rede de interconexão que consiste em $2n$ barramentos. Esses barramentos conectam n portas de entrada com n portas de saída, como mostra a Figura 4.8. Um pacote que chega a uma porta de entrada percorre o barramento horizontal ligado à porta de entrada até interceptar o barramento vertical que leva à porta de saída desejada. Se o barramento vertical que leva à porta de saída estiver livre, o pacote será transferido para a porta de saída. Se o barramento vertical estiver sendo usado para transferir um pacote de outra porta de entrada para essa mesma porta de saída, o pacote que chegou ficará bloqueado e deverá entrar na fila da porta de entrada.

Elementos de comutação Delta e Omega também foram propostos como rede de interconexão entre portas de entrada e de saída. Consulte [Tobagi, 1990] para um levantamento de arquiteturas de comutação. Os comutadores da Família 12000 da Cisco [Cisco 12000, 2009] usam uma rede de interconexão que fornece até 60 Gbps pelo elemento de comutação. Uma tendência no projeto de rede de interconexão [Keshav, 1998] é fragmentar um datagrama IP de comprimento variável em células de comprimento fixo e, então, marcar e comutar as células de comprimento fixo por meio da rede de interconexão. As células são então montadas novamente reproduzindo o pacote original na porta de saída. A célula de comprimento fixo e a tag de controle interno podem simplificar e acelerar consideravelmente a comutação de pacotes pela rede de interconexão.

4.3.3 Portas de saída

O processamento de portas de saída mostrado na Figura 4.9 toma os pacotes que foram armazenados na memória da porta de saída e os transmite pelo enlace de saída. O processamento do protocolo de enlace e a terminação da linha são as funcionalidades de camada de enlace e de camada física do lado remetente que interagem com a porta de entrada do outro lado do enlace de saída, como discutido na Seção 4.3.1. As funcionalidades de fila e de gerenciamento de buffer são necessárias quando o elemento de comutação entrega pacotes à porta de saída a uma taxa que excede a taxa do enlace de saída. Vamos examinar a seguir o enfileiramento na porta de saída.

4.3.4 Onde ocorre formação de fila?

Se examinarmos a funcionalidade da porta de entrada e da porta de saída e as configurações mostradas na Figura 4.8, veremos que filas de pacotes podem se formar tanto nas portas de entrada como nas portas de saída. É importante considerar essas filas com um pouco mais de detalhes, já que, à medida que elas ficam maiores, o espaço de buffer do roteador será finalmente exaurido e ocorrerá **perda de pacote**. Lembre-se de que, em nossas discussões anteriores, dissemos que pacotes eram perdidos dentro da rede ou descartados em um roteador. E é aí, nessas filas dentro de um roteador, que esses pacotes são descartados ou perdidos. O local real da perda do pacote (seja nas filas da porta de entrada, seja nas filas da porta de saída) dependerá da carga de tráfego, da velocidade relativa do elemento de comutação e da taxa da linha, como discutiremos a seguir.

Suponha que as taxas da linha de entrada e as taxas da linha de saída sejam idênticas e que haja n portas de entrada e n portas de saída. Defina a **taxa do elemento de comutação** como a taxa na qual o elemento de comutação pode movimentar pacotes de portas de entrada à portas de saída. Se essa taxa for no mínimo n vezes a taxa da linha de entrada, então não ocorrerá formação de fila nas portas de entrada. Isso porque, mesmo no pior caso em que todas as n linhas de entrada estiverem recebendo pacotes, o comutador poderá transferir n pacotes da porta de entrada para a porta de saída no tempo que levar para cada uma das n portas de entrada receber (simultaneamente) um único pacote. Mas o que pode acontecer nas portas de saída? Vamos ainda supor que a taxa do elemento de comutação seja no mínimo n vezes as taxas das linhas. No pior caso, os pacotes que chegarem a cada uma das n portas de entrada serão destinados à *mesma* porta de saída. Nesse caso, no tempo que leva para receber (ou enviar) um único pacote, n pacotes chegarão a essa porta de saída. Uma vez que a porta de saída pode transmitir somente um único pacote em cada unidade de tempo (o tempo de transmissão do pacote), os n pacotes que chegarem terão de entrar na fila (esperar) para transmissão pelo enlace de saída. Então, mais n pacotes poderão chegar durante o tempo que leva para transmitir apenas um dos n pacotes que estavam na fila anteriormente. E assim por diante. Finalmente, o número de pacotes na fila pode ficar grande o bastante para exaurir o espaço de memória na porta de saída, caso em que os pacotes são descartados.

A formação de fila na porta de saída está ilustrada na Figura 4.10. No tempo t , um pacote chegou a cada uma das portas de entrada, cada um deles destinado à porta de saída que está mais acima na figura. Admitindo taxas da linha idênticas e um comutador operando a uma taxa três vezes maior do que a da linha, uma unidade de tempo mais tarde (isto é, no tempo necessário para receber ou enviar um pacote), todos os três pacotes originais foram transferidos para a porta de saída e estão em fila aguardando transmissão. Na unidade de tempo seguinte, um desses três pacotes terá sido transmitido pelo enlace de saída. Em nosso exemplo, dois novos pacotes chegaram do lado de entrada do comutador; um desses pacotes é destinado àquela mesma porta de saída que está mais acima na figura.

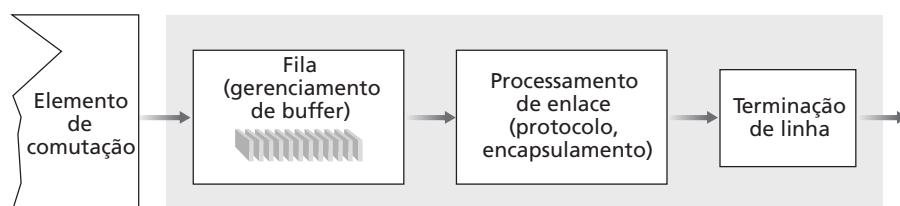


Figura 4.9 Processamento de porta de saída

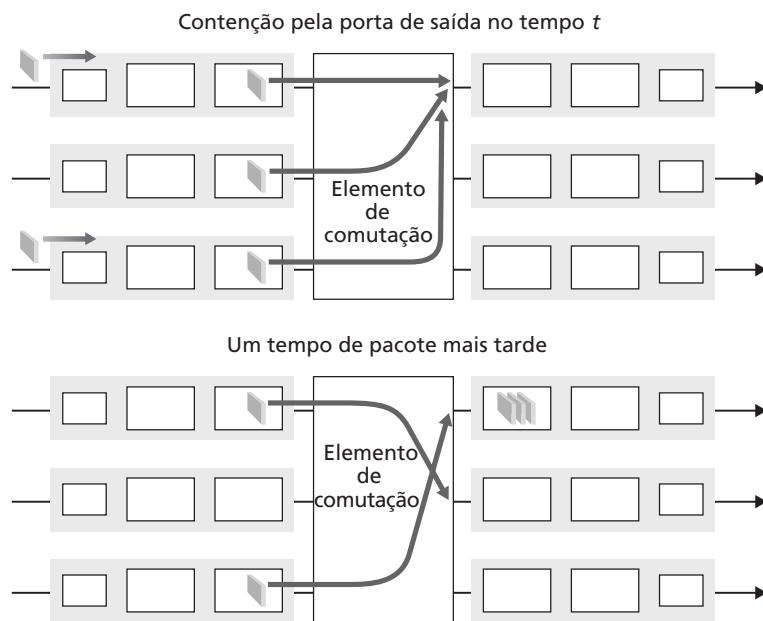


Figura 4.10 Formação de fila na porta de saída

Dado que os buffers do roteador são necessários para absorver as oscilações da carga de tráfego, deve-se perguntar quanto de armazenamento em buffer é necessário. Durante muitos anos, a regra prática [RFC 3439] para dimensionamento de buffer foi que a quantidade de armazenamento em buffer deveria ser igual a um tempo de viagem de ida e volta (RTT, digamos, 250 milissegundos) vezes a capacidade do enlace (C). Esse resultado é baseado em uma análise da dinâmica de filas com um número relativamente pequeno dos fluxos do TCP [Villamizar, 1994]. Assim, um enlace de 10 Gbps com um RTT de 250 milissegundos precisaria de uma quantidade de armazenamento em buffer igual a $B = RTT \cdot C = 2,5$ Gbps de buffers. Esforços teóricos e experimentais recentes [Appenzeller, 2004], entretanto, sugerem que quando há um grande número de fluxos do TCP (N) passando por um enlace, a quantidade de armazenamento em buffer necessária é $B = RTT \cdot C/\sqrt{N}$. Com um grande número de fluxos passando normalmente por grandes enlaces dos roteadores da fonte principal (consulte, por exemplo, [Fraleigh, 2003]), o valor de N pode ser grande, e a diminuição do tamanho do buffer necessário se torna bastante significativa. [Appenzellar, 2004; Wischik, 2005; Beheshti, 2008] apresentam discussões esclarecedoras em relação ao problema do dimensionamento de buffer a partir de um ponto de vista teórico, de aplicação e operacional.

Uma consequência da fila na porta de saída é que um **escalonador de pacotes** na porta de saída deve escolher para transmissão um pacote dentre os que estão na fila. Essa seleção pode ser feita com base em uma regra simples baseada na própria ordem da fila (FCFS) ou por uma regra de escalonamento mais sofisticada, tal como a fila ponderada justa (*weighted fair queuing* — WFQ), que compartilha o enlace de saída com justiça entre as diferentes conexões fim a fim que têm pacotes na fila para transmissão. O escalonamento de pacotes desempenha um papel crucial no fornecimento de **garantia de qualidade de serviço**. Examinaremos o escalonamento de pacotes extensivamente no Capítulo 7. Uma discussão sobre disciplinas de escalonamento de pacotes na porta de saída pode ser encontrada em [Cisco Queue, 2009].

De modo semelhante, se não houver memória suficiente para armazenar um pacote que está chegando, será preciso tomar a decisão de descartar esse pacote (política conhecida como **descarte do final da fila**) ou remover um ou mais pacotes já enfileirados para liberar lugar para o pacote recém-chegado. Em alguns casos pode ser vantajoso descartar um pacote (ou marcar o seu cabeçalho) antes de o buffer ficar cheio, para dar um sinal de congestionamento ao remetente. Várias políticas de descarte e marcação de pacotes (conhecidas coletivamente como algoritmos de **gerenciamento ativo de fila** [*active queue management* — AQM]) foram propostas e analisadas [Labrador, 1999; Hollot, 2002]. Um dos algoritmos AQM mais estudados e implementados é o algoritmo de **detectção aleatória rápida** (*random early detection* — RED). Esse algoritmo mantém uma média ponderada do

comprimento da fila de saída. Se o comprimento médio da fila for menor do que um valor limite mínimo, min_{th} , quando um pacote chegar, será admitido na fila. Inversamente, se a fila estiver cheia ou se o comprimento médio da fila for maior do que um valor limite máximo, max_{th} , quando um pacote chegar, será marcado ou descartado. Finalmente, se o pacote chegar e encontrar uma fila de comprimento médio no intervalo $[min_{th}, max_{th}]$, o pacote será marcado ou descartado com uma probabilidade que normalmente é alguma função do comprimento médio da fila, de min_{th} e de max_{th} . Foram propostas inúmeras funções probabilísticas para atuação nos pacotes e várias versões do RED foram modeladas, simuladas e/ou implementadas analiticamente. [Christiansen, 2001] e [Floyd, 2009] oferecem visões gerais e indicações de leituras adicionais.

Se o elemento de comutação não for suficientemente veloz (em relação às taxas da linha de entrada) para transmitir sem atraso *todos* os pacotes que chegam através dele, então poderá haver formação de fila também nas portas de entrada, pois os pacotes devem se juntar às filas nas portas de entrada para esperar sua vez de ser transferidos através do elemento de comutação até a porta de saída. Para ilustrar uma importante consequência dessa fila, considere um elemento de comutação do tipo ‘crossbar’ e suponha que (1) todas as velocidades de enlace sejam idênticas, (2) um pacote possa ser transferido de qualquer uma das portas de entrada até uma dada porta de saída no mesmo tempo que leva para um pacote ser recebido em um enlace de entrada e (3) pacotes sejam movimentados de uma dada fila de entrada até sua fila de saída desejada no modo FCFS. Vários pacotes podem ser transferidos em paralelo, contanto que suas portas de saída sejam diferentes. Entretanto, se dois pacotes que estão à frente das duas filas de entrada forem destinados à mesma fila de saída, então um dos pacotes ficará bloqueado e terá de esperar na fila de entrada — o elemento comutador só pode transferir um pacote por vez até uma porta de saída.

A parte superior da Figura 4.11 apresenta um exemplo em que dois pacotes (mais escuros) à frente de suas filas de entrada são destinados à mesma porta de saída mais alta à direita. Suponha que o elemento de comutação escolha transferir o pacote que está à frente da fila mais alta à esquerda. Nesse caso, o pacote mais escuro na fila mais baixa à esquerda tem de esperar. Mas não é apenas este último que tem de esperar; também tem de esperar o pacote claro que está na fila atrás dele (no retângulo inferior à esquerda), mesmo que não haja *nenhuma* contenção pela porta de saída do meio à direita (que é o destino do pacote claro). Esse fenômeno é conhecido como

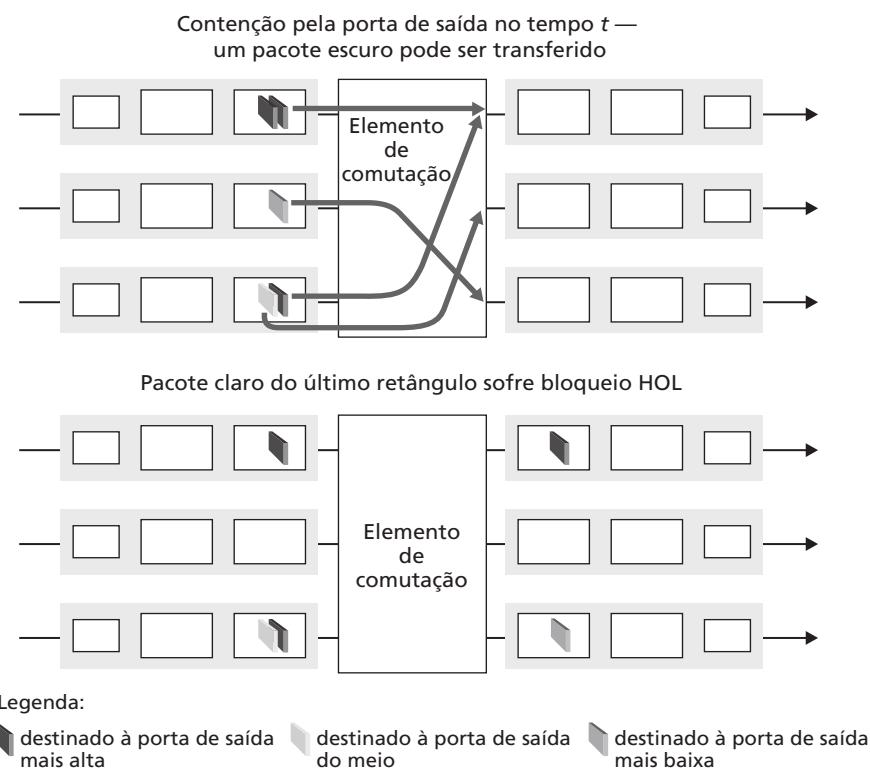


Figura 4.11 Bloqueio de cabeça de fila em um comutador com fila de entrada

bloqueio de cabeça de fila (*head-of-the-line — HOL blocking*) em um comutador com fila de entrada — um pacote que está na fila de entrada deve esperar pela transferência através do elemento de comutação (mesmo que sua porta de saída esteja livre) porque ele está bloqueado por um outro pacote na cabeça da fila. [Karol, 1987] demonstra que, devido ao bloqueio HOL, o comprimento da fila de entrada cresce sem limites (informalmente, isso equivale a dizer que haverá significativas perdas de pacotes) em determinadas circunstâncias assim que a taxa de chegada de pacotes no enlace de entrada alcançar apenas 58 por cento de sua capacidade. Uma série de soluções para o bloqueio HOL é discutida por [McKeown, 1997b].

4.4 O Protocolo da Internet (IP): repasse e endereçamento na Internet

Até agora discutimos endereçamento e repasse na camada de rede, sem referências a nenhuma rede de computadores específica. Nesta seção, voltaremos nossa atenção a como são feitos o endereçamento e o repasse na Internet. Veremos que o endereçamento e o repasse na Internet são componentes importantes do Protocolo da Internet (IP). Há duas versões do protocolo IP em uso hoje. Examinaremos primeiramente a versão mais utilizada do IP, a versão 4, que normalmente é denominada simplesmente IPv4 [RFC 791]. Examinaremos a versão 6 do IP [RFC 2460; RFC 4291], que foi proposta para substituir o IPv4, no final desta seção.

Mas, antes de iniciar nosso ataque ao IP, vamos voltar um pouco atrás e considerar os componentes que formam a camada de rede da Internet. Como mostra a Figura 4.12, a camada de rede da Internet tem três componentes mais importantes. O primeiro componente é o protocolo IP, que é o tópico desta seção. O segundo componente mais importante é o componente de roteamento, que determina o caminho que um datagrama segue desde a origem até o destino. Mencionamos anteriormente que protocolos de roteamento calculam as tabelas de repasse que são usadas para transmitir pacotes pela rede. Estudaremos os protocolos de roteamento da Internet na Seção 4.6. O componente final da camada de rede é um dispositivo para comunicação de erros em datagramas e para atender requisições de certas informações de camada de rede. Examinaremos o protocolo de comunicação de erro e de informações da Internet, ICMP (*Internet Control Message Protocol*), na Seção 4.4.3.

4.4.1 Formato do datagrama

Lembre-se de que um pacote de camada de rede é denominado um *datagrama*. Iniciamos nosso estudo do IP com uma visão geral da sintaxe e da semântica do datagrama IPv4. Você talvez esteja pensando que nada poderia ser mais desinteressante do que a sintaxe e a semântica dos bits de um pacote. Mesmo assim, o datagrama desempenha

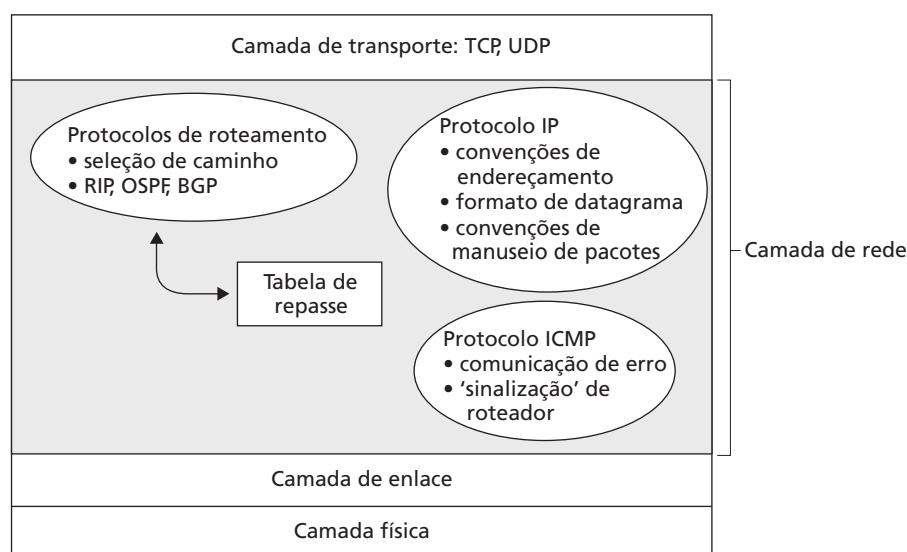


Figura 4.12 Contemplando o interior da camada de rede da Internet

um papel central na Internet — todos os estudantes e profissionais de rede precisam vê-lo, absorvê-lo e dominá-lo. O formato do datagrama IPv4 é mostrado na Figura 4.13. Os principais campos desse datagrama são os seguintes:

Número da versão. Esses quatro bits especificam a versão do protocolo IP do datagrama. Examinando o número da versão, o roteador pode determinar como interpretar o restante do datagrama IP. Diferentes versões de IP usam diferentes formatos de datagramas. O formato do datagrama para a versão atual do IP (IPv4) é mostrado na Figura 4.13. O formato do datagrama para a nova versão do IP (IPv6) será discutido no final desta seção.

Comprimento do cabeçalho. Como um datagrama IPv4 pode conter um número variável de opções (incluídas no cabeçalho do datagrama IPv4), esses quatro bits são necessários para determinar onde, no datagrama IP, os dados realmente começam. A maior parte dos datagramas IP não contém opções; portanto, o datagrama IP típico tem um cabeçalho de 20 bytes.

Tipo de serviço. Os bits de tipo de serviço (*type of service* — TOS) foram incluídos no cabeçalho do IPv4 para poder diferenciar os diferentes tipos de datagramas IP (por exemplo, datagramas que requerem, particularmente, baixo atraso, alta vazão ou confiabilidade) que devem ser distinguidos uns dos outros. Por exemplo, poderia ser útil distinguir datagramas de tempo real (como os usados por uma aplicação de telefonia IP) de tráfego que não é de tempo real (por exemplo, FTP). O nível específico de serviço a ser fornecido é uma questão de política determinada pelo administrador do roteador. Examinaremos detalhadamente a questão do serviço diferenciado no Capítulo 7.

Comprimento do datagrama. É o comprimento total do datagrama IP (cabeçalho mais dados) medido em bytes. Uma vez que esse campo tem 16 bits de comprimento, o tamanho máximo teórico do datagrama IP é 65.535 bytes. Contudo, datagramas raramente são maiores do que 1.500 bytes.

Identificador, flags, deslocamento de fragmentação. Esses três campos têm a ver com a fragmentação do IP, um tópico que em breve vamos considerar detalhadamente. O interessante é que a nova versão do IP, o IPv6, não permite fragmentação em roteadores.

Tempo de vida. O campo de tempo de vida (*time-to-live* — TTL) é incluído para garantir que datagramas não fiquem circulando para sempre na rede (devido a, por exemplo, um laço de roteamento de longa duração). Esse campo é decrementado de uma unidade cada vez que o datagrama é processado por um roteador. Se o campo TTL chegar a 0, o datagrama deve ser descartado.

Protocolo. Esse campo é usado somente quando um datagrama IP chega a seu destino final. O valor desse campo indica o protocolo de camada de transporte específico ao qual a porção de dados desse

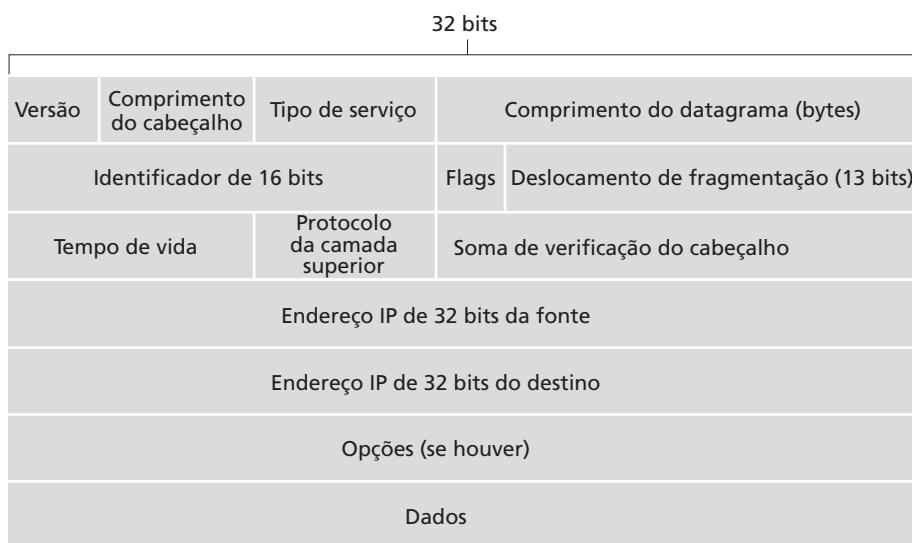


Figura 4.13 Formato do datagrama IPv4



datagrama IP deverá ser passada. Por exemplo, um valor 6 indica que a porção de dados será passada ao TCP, enquanto um valor 17 indica que os dados serão passados ao UDP. Consulte [IANA Protocol Number, 2009] para ver uma lista de todos os valores possíveis. Note que o número do protocolo no datagrama IP tem um papel que é análogo ao do campo de número de porta no segmento da camada de transporte. O número do protocolo é o adesivo que liga as camadas de rede e de transporte, ao passo que o número de porta é o adesivo que liga as camadas de transporte e de aplicação. Veremos no Capítulo 5 que o quadro de camada de enlace também tem um campo especial que liga a camada de enlace à camada de rede.

Soma de verificação do cabeçalho. A soma de verificação do cabeçalho auxilia um roteador na detecção de erros de bits em um datagrama IP recebido. A soma é calculada tratando cada 2 bytes do cabeçalho como se fossem um número e somando esses números usando complementos aritméticos de 1. Como discutimos na Seção 3.3, o complemento de 1 dessa soma, conhecida como soma de verificação da Internet, é armazenado no campo de soma de verificação. Um roteador calculará o valor da soma de verificação para cada datagrama IP recebido e detectará uma condição de erro se o valor da soma de verificação carregado no cabeçalho do datagrama não for igual à soma de verificação calculada. Roteadores normalmente descartam datagramas quando um erro é detectado. Note que a soma de verificação deve ser recalculada e armazenada novamente em cada roteador, pois o campo TTL e, possivelmente, também os campos de opções podem mudar. Uma discussão interessante sobre algoritmos rápidos para calcular a soma de verificação da Internet é encontrada em [RFC 1071]. Uma pergunta que sempre é feita nesse ponto é: por que o TCP/IP faz verificação de erro nas camadas de transporte e de rede? Há várias razões para essa repetição. Em primeiro lugar, note que, na camada IP, a soma de verificação é calculada somente para o cabeçalho IP, enquanto no TCP/UDP a soma de verificação é calculada para todo o segmento TCP/IP. Em segundo lugar, o TCP/UDP e o IP não precisam necessariamente pertencer à mesma pilha de protocolos. O TCP pode, em princípio, rodar sobre um protocolo diferente (por exemplo, ATM) e o IP pode carregar dados que não serão passados ao TCP/UDP.

Endereços IP de fonte e de destino. Quando uma fonte cria um datagrama, insere seu endereço IP no campo de endereço de fonte IP e insere o endereço do destino final no campo de endereço de destinatário IP. Muitas vezes o hospedeiro da fonte determina o endereço do destinatário por meio de uma consulta ao DNS, como discutimos no Capítulo 2. Discutiremos endereçamento IP detalhadamente na Seção 4.4.2.

Opções. O campo de opções permite que um cabeçalho IP seja ampliado. A intenção é que as opções de cabeçalho sejam usadas raramente — por isso a decisão de poupar sobrecarga não inclui a informação em campos de opções em todos os cabeçalhos de datagrama. Contudo, a mera existência de opções, na realidade, complica as coisas — uma vez que cabeçalhos de datagramas podem ter comprimentos variáveis, não é possível determinar *a priori* onde começará o campo de dados. Além disso, uma vez que alguns datagramas podem requerer processamento de opções e outros não, a quantidade de tempo necessária para processar um datagrama IP em um roteador pode variar bastante. Essas considerações se tornam particularmente importantes para o processamento do IP em roteadores e hospedeiros de alto desempenho. Por essas e outras razões, as opções IP foram descartadas no cabeçalho da versão IPv6, como discutimos na Seção 4.4.4.

Dados (carga útil). Finalmente, chegamos ao último e mais importante campo — a razão de ser do datagrama! Em muitas circunstâncias, o campo de dados do datagrama IP contém o segmento da camada de transporte (TCP ou UDP) a ser entregue ao destino. Contudo, o campo de dados pode carregar outros tipos de dados, como mensagens ICMP (discutidas na Seção 4.4.3).

Note que um datagrama IP tem um total de 20 bytes de cabeçalho (admitindo-se que não haja opções). Se o datagrama carregar um segmento TCP, então cada datagrama (não fragmentado) carrega um total de 40 bytes de cabeçalho (20 bytes de cabeçalho IP, mais 20 bytes de cabeçalho TCP) juntamente com a mensagem de camada de aplicação.

Fragmentação do datagrama IP

Veremos no Capítulo 5 que nem todos os protocolos de camada de enlace podem transportar pacotes do mesmo tamanho. Alguns protocolos podem transportar datagramas grandes, ao passo que outros podem transportar apenas pacotes pequenos. Por exemplo, quadros Ethernet não podem conter mais do que 1.500 bytes de dados, enquanto quadros para alguns enlaces de longa distância não podem conter mais do que 576 bytes. A quantidade máxima de dados que um quadro de camada de enlace pode carregar é denominada unidade máxima de transmissão (*maximum transmission unit* — MTU). Como cada datagrama IP é encapsulado dentro do quadro de camada de enlace para ser transportado de um roteador até o roteador seguinte, a MTU do protocolo de camada de enlace estabelece um limite estrito para o comprimento de um datagrama IP. Ter uma limitação estrita para o tamanho de um datagrama IP não é grande problema. O problema é que cada um dos enlaces ao longo da rota entre remetente e destinatário pode usar diferentes protocolos de camada de enlace, e cada um desses protocolos pode ter diferentes MTUs.

Para entender melhor a questão do repasse, imagine que você é um roteador que está interligando diversos enlaces, cada um rodando diferentes protocolos de camada de enlace com diferentes MTUs. Suponha que você receba um datagrama IP de um enlace, verifique sua tabela de repasse para determinar o enlace de saída e perceba que esse enlace de saída tem uma MTU que é menor do que o comprimento do datagrama IP. É hora de entrar em pânico — como você vai comprimir esse datagrama IP de tamanho excessivo no campo de carga útil do quadro da camada de enlace? A solução para esse problema é fragmentar os dados do datagrama IP em dois ou mais datagramas IP menores e, então, enviar esses datagramas menores pelo enlace de saída. Cada um desses datagramas menores é denominado um **fragmento**.

Fragmentos precisam ser reconstruídos antes que cheguem à camada de transporte no destino. Na verdade, tanto o TCP quanto o UDP esperam receber da camada de rede segmentos completos, não fragmentados. Os projetistas do IPv4 perceberam que a reconstrução de datagramas nos roteadores introduziria uma complicação significativa no protocolo e colocaria um freio no desempenho do roteador. (Se você fosse um roteador, iria querer reconstruir fragmentos além de tudo mais que tem de fazer?) Segundo o princípio de manter a simplicidade do núcleo da rede, os projetistas do IPv4 decidiram alocar a tarefa de reconstrução de datagramas aos sistemas finais, e não aos roteadores da rede.

Quando um hospedeiro destinatário recebe uma série de datagramas da mesma fonte, ele precisa determinar se alguns desses datagramas são fragmentos de um datagrama original de maior tamanho. Se alguns desses datagramas forem fragmentos, o hospedeiro ainda deverá determinar quando recebeu o último fragmento e como os fragmentos recebidos devem ser reconstruídos para voltar à forma do datagrama original. Para permitir que o hospedeiro destinatário realize essas tarefas de reconstrução, os projetistas do IP (versão 4) criaram campos de *identificação*, *flag* e *deslocamento* de fragmentação no cabeçalho do datagrama IP. Quando um datagrama é criado, o hospedeiro remetente marca o datagrama com um número de identificação, bem como com os endereços da fonte e do destino. O hospedeiro remetente incrementa o número de identificação para cada datagrama que envia. Quando um roteador precisa fragmentar um datagrama, cada datagrama resultante (isto é, cada fragmento) é marcado com o endereço da fonte, o endereço do destino e o número de identificação do datagrama original. Quando o destinatário recebe uma série de datagramas do mesmo hospedeiro remetente, pode examinar os números de identificação dos datagramas para determinar quais deles são, na verdade, fragmentos de um mesmo datagrama de tamanho maior. Como o IP é um serviço não confiável, é possível que um ou mais desses fragmentos jamais cheguem ao destino. Por essa razão, para que o hospedeiro de destino fique absolutamente seguro de que recebeu o último fragmento do datagrama original, o último datagrama tem um bit de flag ajustado para 0, ao passo que todos os outros fragmentos têm um bit de flag ajustado para 1. Além disso, para que o hospedeiro destinatário possa determinar se está faltando algum fragmento (e possa reconstruir os fragmentos na ordem correta), o campo de deslocamento é usado para especificar a localização exata do fragmento no datagrama IP original.

A Figura 4.14 apresenta um exemplo. Um datagrama de 4 mil bytes (20 bytes de cabeçalho IP, mais 3.980 bytes de carga útil IP) chega a um roteador e deve ser repassado a um enlace com MTU de 1.500 bytes. Isso implica que os 3.980 bytes de dados do datagrama original devem ser alocados a três fragmentos separados (cada qual é também um datagrama IP). Suponha que o datagrama original esteja marcado com o número de

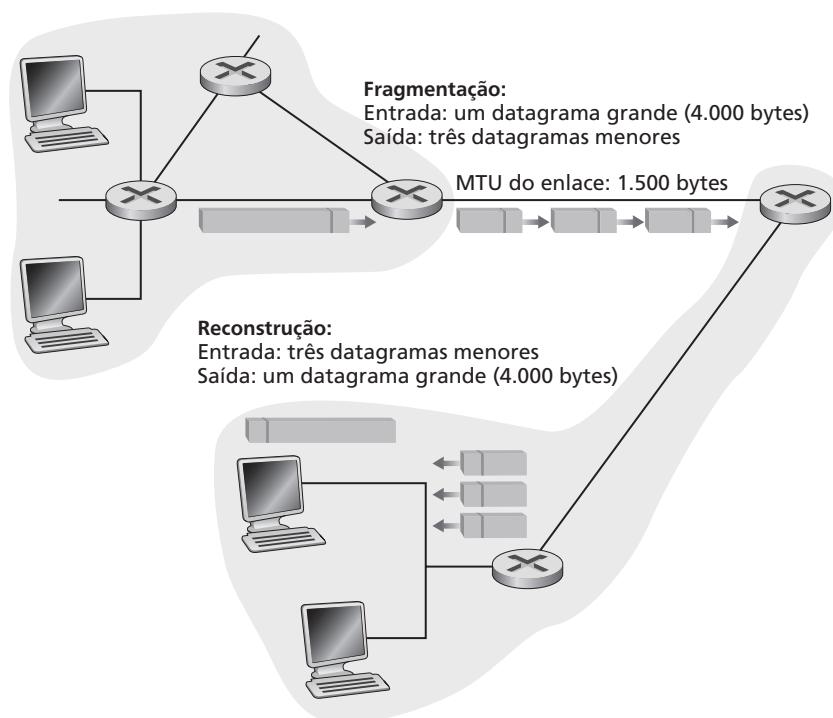


Figura 4.14 Fragmentação e reconstrução IP

identificação 777. As características dos três fragmentos são mostradas na Tabela 4.2. Os valores da Tabela 4.2 refletem a exigência de que a quantidade de dados da carga útil original em todos os fragmentos, exceto o último, seja um múltiplo de 8 bytes, e que o valor de deslocamento seja especificado em unidades de porções de 8 bytes.

No destino, a carga útil do datagrama é passada para a camada de transporte somente após a camada IP ter reconstruído totalmente o datagrama IP original. Se um ou mais fragmentos não chegarem ao destino, o datagrama incompleto será descartado e não será passado à camada de transporte. Mas, como aprendemos no capítulo anterior, se o TCP estiver sendo usado na camada de transporte, ele recuperará essa perda fazendo com que a fonte retransmita os dados do datagrama original.

Vimos que a fragmentação do IP tem papel importante na união das diversas tecnologias distintas da camada de enlace. Mas a fragmentação também tem seu preço. Em primeiro lugar, ela dificulta roteadores e sistemas finais, que precisam ser projetados para acomodar a fragmentação do datagrama e o reagrupamento. Em segundo lugar, a fragmentação pode ser usada para criar ataques DoS fatais, através do qual o atacante envia uma série de fragmentos

Fragmento	Bytes	ID	Deslocamento	Flag
1º fragmento	1.480 bytes no campo de dados do datagrama IP	identificação = 777	deslocamento = 0 (o que significa que os dados devem ser inseridos a partir do byte 0)	flag = 1 (o que significa que há mais)
2º fragmento	1.480 bytes de dados	identificação = 777	deslocamento = 185 (o que significa que os dados devem ser inseridos a partir do byte 1.480. Note que $185 \times 8 = 1.480$)	flag = 1 (o que significa que há mais)
3º fragmento	1.020 bytes de dados ($= 3.980 - 1.480 - 1.480$)	identificação = 777	deslocamento = 370 (o que significa que os dados devem ser inseridos a partir do byte 2.960. Note que $370 \times 8 = 2.960$)	flag = 0 (o que significa que esse é o último fragmento)

Tabela 4.2 Fragmentos IP

estranhos e inesperados. Um exemplo clássico é o ataque Jolt2, no qual o atacante envia uma cadeia de pequenos fragmentos para o computador-alvo, nenhum dos quais tem um deslocamento de zero. O alvo pode se recolher ao tentar reconstruir datagramas pelos pacotes degenerados. Uma outra classe de exploits envia sobreposição de fragmentos IP, ou seja, fragmentos cujos valores de deslocamento são definidos para que os fragmentos não se alinhem corretamente. Sistemas operacionais vulneráveis, sem saber o que fazer com a sobreposição de fragmentos, podem pifar [Skoudis, 2006]. Como veremos no final desta seção, uma nova versão do protocolo IP, o IPv6, põe fim por completo à fragmentação, o processamento do pacote IP e tornando o IP menos vulnerável a ataques.

No site Web deste livro apresentamos um applet Java que gera fragmentos. Basta informar o tamanho do datagrama que está chegando, a MTU e a identificação do datagrama, e o applet gera automaticamente os fragmentos para você. Veja http://www.aw.com/kurose_br.

4.4.2 Endereçamento IPv4

Agora voltaremos nossa atenção ao endereçamento IPv4. Embora você talvez esteja pensando que o endereçamento seja um tópico descomplicado, esperamos que, no final deste capítulo, você se convença de que o endereçamento na Internet não é apenas um tópico rico, cheio de sutilezas e interessante, mas também de crucial importância para a Internet. Tratamentos excelentes do endereçamento IPv4 podem ser encontrados em [3Com Addressing, 2009] e no primeiro capítulo de [Stewart, 1999].

Antes de discutirmos o endereçamento IP, contudo, temos de falar um pouco sobre como hospedeiros e roteadores estão interconectados na rede. Um hospedeiro normalmente tem apenas um único enlace com a rede. Quando o IP no hospedeiro quer enviar um datagrama, ele o faz por meio desse enlace. A fronteira entre o hospedeiro e o enlace físico é denominada **interface**. Agora considere um roteador e suas interfaces. Como a tarefa de um roteador é receber um datagrama em um enlace e repassá-lo a algum outro enlace, ele necessariamente estará ligado a dois ou mais enlaces. A fronteira entre o roteador e qualquer um desses enlaces também é denominada uma interface. Assim, um roteador tem múltiplas interfaces, uma para cada um de seus enlaces. Como todos os hospedeiros e roteadores podem enviar e receber datagramas IP, o IP exige que cada interface tenha seu próprio endereço IP. Desse modo, um endereço IP está tecnicamente associado com uma interface, e não com um hospedeiro ou um roteador que contém aquela interface.

Cada endereço IP tem comprimento de 32 bits (equivalente a 4 bytes). Portanto, há um total de 2^{32} endereços IP possíveis. Fazendo uma aproximação de 2^{10} por 10^3 , é fácil ver que há cerca de 4 bilhões de endereços IP possíveis. Esses endereços são escritos em **notação decimal separada por pontos** (*dotted-decimal notation*), na qual cada byte do endereço é escrito em sua forma decimal e separado dos outros bytes do endereço por um ponto. Por exemplo, considere o endereço IP 193.32.216.9. O 193 é o número decimal equivalente aos primeiros 8 bits do endereço; o 32 é o decimal equivalente ao segundo conjunto de 8 bits do endereço e assim por diante. Por conseguinte, o endereço 193.32.216.9, em notação binária, é:

11000001 00100000 11011000 00001001

Cada interface em cada hospedeiro e roteador da Internet global tem de ter um endereço IP globalmente exclusivo (exceto as interfaces por trás de NATs, como discutiremos no final desta seção). Contudo, esses endereços não podem ser escolhidos de qualquer maneira. Uma parte do endereço IP de uma interface será determinada pela sub-rede à qual ela está conectada.

A Figura 4.15 fornece um exemplo de endereçamento IP e interfaces. Nessa figura, um roteador (com três interfaces) é usado para interconectar sete hospedeiros. Examine os endereços IP atribuídos às interfaces de hospedeiros e roteadores; há diversas particularidades a notar. Todos os três hospedeiros da parte superior esquerda da Figura 4.15 — e também a interface do roteador ao qual estão ligados — têm um endereço IP na forma 223.1.1.xxx, ou seja, todos eles têm os mesmos 24 bits mais à esquerda em seus endereços IP. As quatro interfaces também estão interconectadas por uma rede que não contém nenhum roteador. (Essa rede poderia ser, por exemplo, uma LAN Ethernet, caso em que as interfaces seriam conectadas por um hub Ethernet ou um comutador Ethernet; veja o Capítulo 5). Na terminologia IP, essa rede que interconecta três interfaces de hospedeiros e uma interface de roteador forma uma **sub-rede** [RFC 950]. (Na literatura da Internet, uma sub-rede também é

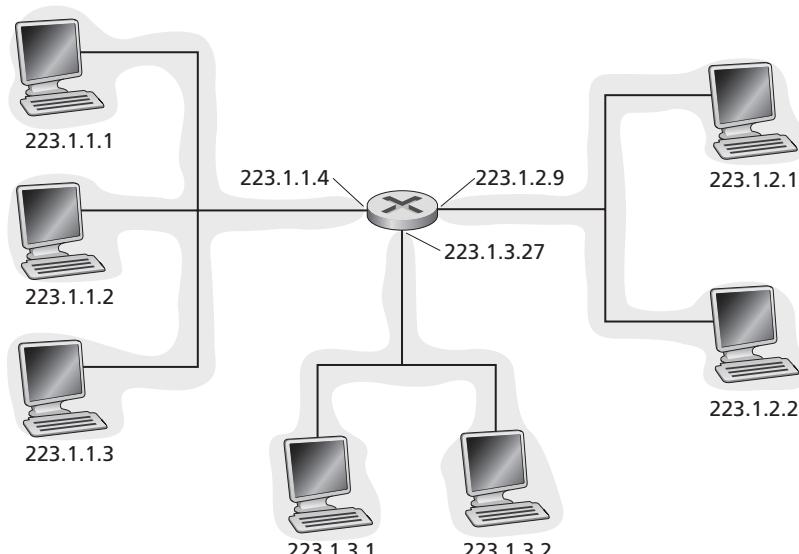


Figura 4.15 Endereços de interfaces e sub-redes

denominada uma *rede IP* ou simplesmente uma *rede*). O endereçamento IP designa um endereço a essa sub-rede: 223.1.1.0/24, no qual a notação /24, às vezes conhecida como uma **máscara de rede**, indica que os 24 bits mais à esquerda do conjunto de 32 bits definem o endereço da sub-rede. Assim, a sub-rede 223.1.1.0/24 consiste em três interfaces de hospedeiros (223.1.1.1, 223.1.1.2 e 223.1.1.3) e uma interface de roteador (223.1.1.4). Quaisquer hospedeiros adicionais ligados à sub-rede 223.1.1.0/24 seriam obrigados a ter um endereço na forma 223.1.1.xxx. Há duas sub-redes adicionais mostradas na Figura 4.15: a sub-rede 223.1.2.0/24 e a sub-rede 223.1.3.0/24. A Figura 4.16 ilustra as três sub-redes IP presentes na Figura 4.15.

A definição IP de uma sub-rede não está restrita a segmentos Ethernet que conectam múltiplos hospedeiros a uma interface de roteador. Para entender melhor isso, considere a Figura 4.17, que mostra três roteadores interconectados por enlaces ponto a ponto. Cada roteador tem três interfaces, uma para cada enlace ponto a ponto e uma para o enlace broadcast que conecta diretamente o roteador a um par de hospedeiros. Que sub-redes IP estão presentes aqui? Três sub-redes, 223.1.1.0/24, 223.1.2.0/24 e 223.1.3.0/24, semelhantes às sub-redes que

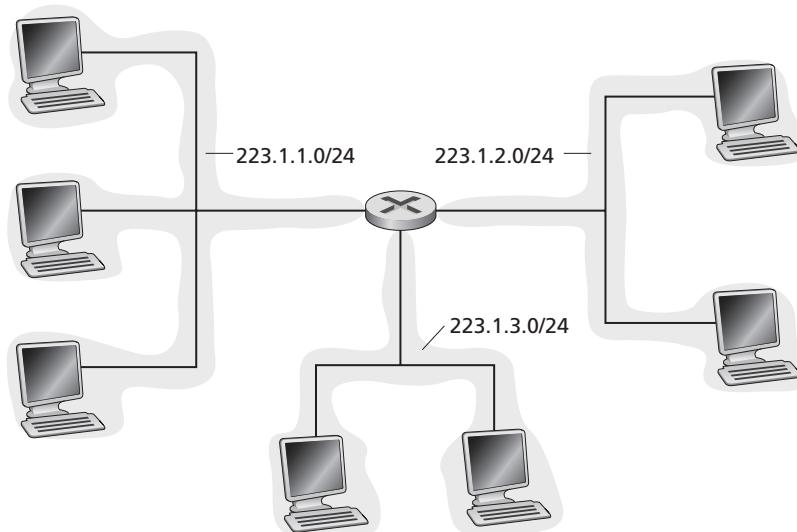


Figura 4.16 Endereços de sub-redes

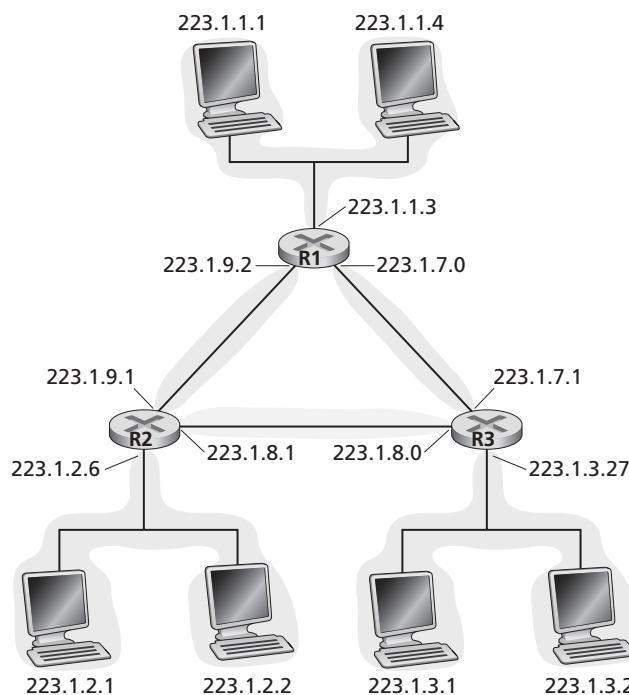


Figura 4.17 Três roteadores interconectando seis sub-redes

encontramos na Figura 4.15. Mas note que também há três sub-redes adicionais nesse exemplo: uma sub-rede, 223.1.9.0/24, para as interfaces que conectam os roteadores R1 e R2; outra sub-rede, 223.1.8.0/24, para as interfaces que conectam os roteadores R2 e R3, e uma terceira sub-rede, 223.1.7.0/24, para as interfaces que conectam os roteadores R3 e R1. Para um sistema geral interconectado de roteadores e hospedeiros, podemos usar a seguinte receita para definir as sub-redes no sistema.

*Para determinar as sub-redes, destaque cada interface de seu hospedeiro ou roteador, criando ilhas de redes isoladas com interfaces fechando as terminações das redes isoladas. Cada uma dessas redes isoladas é denominada **sub-rede**.*

Se aplicarmos esse procedimento ao sistema interconectado da Figura 4.17, teremos seis ilhas ou sub-redes.

Fica claro, com essa discussão, que uma organização (tal como uma empresa ou instituição acadêmica) que tenha vários segmentos Ethernet e enlaces ponto a ponto terá várias sub-redes, e todos os equipamentos e dispositivos em uma dada sub-rede terão o mesmo endereço de sub-rede. Em princípio, as diferentes sub-redes poderiam ter endereços bem diferentes. Entretanto, na prática, os endereços de sub-rede frequentemente têm muito em comum. Para entender por que, voltemos nossa atenção ao modo como o endereçamento é manipulado na Internet global.

A estratégia de atribuição de endereços da Internet é conhecida como **roteamento interdomínio sem classes** (*classless interdomain routing* — CIDR), que se pronuncia como a palavra *cider* (cidra) em inglês [RFC 4632]. O CIDR generaliza a noção de endereçamento de sub-rede. Como acontece com o endereçamento de sub-redes, o endereço IP de 32 bits é dividido em duas partes e, mais uma vez, tem a forma decimal com pontos de separação *a.b.c.d/x*, em que *x* indica o número de bits existentes na primeira parte do endereço.

Os *x* bits mais significativos de um endereço na forma *a.b.c.d/x* constituem a parcela da rede do endereço IP e normalmente são denominados **prefixo** (ou *prefixo de rede*). Uma organização normalmente recebe um bloco de endereços contíguos, isto é, uma faixa de endereços com um prefixo comum (ver quadro Princípios na prática). Nesse caso, os endereços IP de equipamentos e dispositivos dentro da organização compartilharão o prefixo comum. Quando estudarmos, na Seção 4.6, o protocolo de roteamento BGP da Internet, veremos que somente esses *x* bits indicativos do prefixo são considerados por roteadores que estão fora da rede da organização. Isto é,



quando um roteador de fora da organização repassar um datagrama cujo endereço de destino esteja dentro da organização terá de considerar somente os x bits indicativos do endereço. Isso reduz consideravelmente o tamanho da tabela de repasse nesses roteadores, visto que um único registro da forma $a.b.c.d/x$ será suficiente para transmitir pacotes para *qualquer* destino dentro da organização.

Os restantes ($32 - x$) bits de um endereço podem ser considerados como os bits que distinguem os equipamentos e dispositivos *dentro* da organização e todos eles têm o mesmo prefixo de rede. Esses serão os bits considerados no repasse de pacotes em roteadores *dentro* da organização. Esses bits de ordem mais baixa podem (ou não) ter uma estrutura adicional de sub-rede tal como aquela discutida anteriormente. Por exemplo, suponha que os primeiros 21 bits do endereço $a.b.c.d/21$, por assim dizer, ‘ciderizado’, especificam o prefixo da rede da organização e são comuns aos endereços IP de todos os hospedeiros da organização. Os 11 bits restantes então identificam os hospedeiros específicos da organização. A estrutura interna da organização poderia ser tal que esses 11 bits mais à direita seriam usados para criar uma sub-rede dentro da organização, como discutido anteriormente. Por exemplo, $a.b.c.d/24$ poderia se referir a uma sub-rede específica dentro da organização.

Antes da adoção do CIDR, os tamanhos das parcelas de um endereço IP estavam limitados a 8, 16 ou 24 bits, um esquema de endereçamento conhecido como **endereçamento de classes cheias**, já que sub-redes com endereços de sub-rede de 8, 16 e 24 eram conhecidas como redes de classe A, B e C, respectivamente. A exigência de que a parcela da sub-rede de um endereço IP tenha exatamente 1, 2 ou 3 bytes há muito tempo se mostrou problemática para suportar o rápido crescimento do número de organizações com sub-redes de pequeno e médio portes. Uma sub-rede de classe C (/24) poderia acomodar apenas $2^8 - 2 = 254$ hospedeiros (dois dos $2^8 = 256$ endereços são reservados para uso especial) — muito pequena para inúmeras organizações. Contudo, uma sub-rede de classe B (/16), que suporta até 65.634 hospedeiros, seria demasiadamente grande. Com o endereçamento de classes cheias, uma organização com, digamos, dois mil hospedeiros, recebia um endereço de sub-rede de classe B (/16), o que resultava no rápido esgotamento do espaço de endereços de classe B e na má utilização do espaço de endereço alocado. Por exemplo, uma organização que usasse um endereço de classe B para seus dois mil hospedeiros, recebia espaço de endereços suficiente para até 65.534 interfaces — o que deixava mais de 63 mil endereços sem uso e que não poderiam ser utilizados por outras organizações.

Seríamos omissos se não mencionássemos ainda um outro tipo de endereço IP, o endereço de *broadcast* 255.255.255.255. Quando um hospedeiro emite um datagrama com endereço de destino 255.255.255.255, a mensagem é entregue a todos os hospedeiros na mesma sub-rede. Os roteadores também têm a opção de repassar a mensagem para suas sub-redes vizinhas (embora usualmente não o façam.)

Agora que já estudamos o endereçamento IP detalhadamente, precisamos saber, antes de qualquer coisa, como hospedeiros e sub-redes obtêm seus endereços. Vamos começar examinando como uma organização obtém um bloco de endereços para seus equipamentos e, então, veremos como um equipamento (tal como um hospedeiro) recebe um endereço do bloco de endereços da organização.

Obtenção de um bloco de endereços

Para obter um bloco de endereços IP para utilizar dentro da sub-rede de uma organização, um administrador de rede poderia, em primeiro lugar, contatar seu ISP, que forneceria endereços a partir de um bloco maior de endereços que já estão alocados ao ISP. Por exemplo, o próprio ISP pode ter recebido o bloco de endereços 200.23.16.0/20. O ISP, por sua vez, pode dividir seu bloco de endereços em oito blocos de endereços contíguos, do mesmo tamanho, e dar um desses blocos de endereços a cada uma de um conjunto de oito organizações suportadas por ele, como demonstrado a seguir. (Sublinhamos a parte da sub-rede desses endereços para melhor visualização.)

Bloco do ISP	200.23.16.0/20	<u>11001000 00010111 00010000</u>	00000000
Organização 0	200.23.16.0/23	<u>11001000 00010111 00010000</u>	00000000
Organização 1	200.23.18.0/23	<u>11001000 00010111 00010010</u>	00000000
Organização 2	200.23.20.0/23	<u>11001000 00010111 00010100</u>	00000000
...	...		
Organização 7	200.23.30.0/23	<u>11001000 00010111 00011110</u>	00000000

Embora a obtenção de um conjunto de endereços de um ISP seja um modo de conseguir um bloco de endereços, não é o único. Evidentemente, também deve haver um modo de o próprio ISP obter um bloco de endereços. Há uma autoridade global que tenha a responsabilidade final de gerenciar o espaço de endereços IP e alocar blocos de endereços a ISPs e outras organizações? É claro que há! Endereços IP são administrados sob a autoridade da Internet Corporation for Assigned Names and Numbers (ICANN) [ICANN, 2009], com base em diretrizes estabelecidas no RFC 2050. O papel da ICANN, uma organização sem fins lucrativos [NTIA, 1998], não é somente alocar endereços IP, mas também administrar os servidores de nome raiz (DNS). Essa organização também tem a controvertida tarefa de atribuir nomes de domínio e resolver disputas de nomes de domínio. A ICANN aloca endereços a serviços regionais de registro da Internet (por exemplo, ARIN, RIPE, APNIC e LACNIC), que, juntas, formam a Address Supporting Organization da ICANN [ASO-ICANN, 2001], e é responsável pela alocação/administração de endereços dentro de suas regiões.

Princípios na prática

Esse exemplo de um ISP que conecta oito organizações com a Internet também ilustra de maneira elegante como endereços ‘ciderizados’ cuidadosamente alocados facilitam o roteamento. Suponha, como mostra a Figura 4.18, que o ISP (que chamaremos de ISP Fly-By-Night) anuncie ao mundo exterior que devem ser enviados a ele quaisquer datagramas cujos primeiros 20 bits de endereço sejam iguais a 200.23.16.0/20. O resto do mundo não precisa saber que dentro do bloco de endereços 200.23.16.0/20 existem, na verdade, oito outras organizações, cada qual com suas próprias sub-redes. Essa capacidade de usar um único prefixo de rede para anunciar várias redes é frequentemente denominada **agregação de endereços** (e também **agregação de rotas** ou **resumo de rotas**).

A agregação de endereços funciona muito bem quando os endereços são alocados em blocos ao ISP e, então, pelo ISP às organizações clientes. Mas o que acontece quando os endereços não são alocados dessa maneira hierárquica? O que aconteceria, por exemplo, se o ISP Fly-By-Night adquirisse o ISPs-R-U's e então ligasse a Organização 1 com a Internet por meio de sua subsidiária ISPs-R-U's? Como mostra a Figura 4.18, a subsidiária ISPs-R-U's é dona do bloco de endereços 199.31.0.0/16, mas os endereços IP da Organização 1 infelizmente estão fora desse bloco de endereços. O que deveria ser feito nesse caso? Com certeza, a Organização 1 poderia renumerar todos os seus roteadores e hospedeiros para que seus endereços ficassem dentro do bloco de endereços

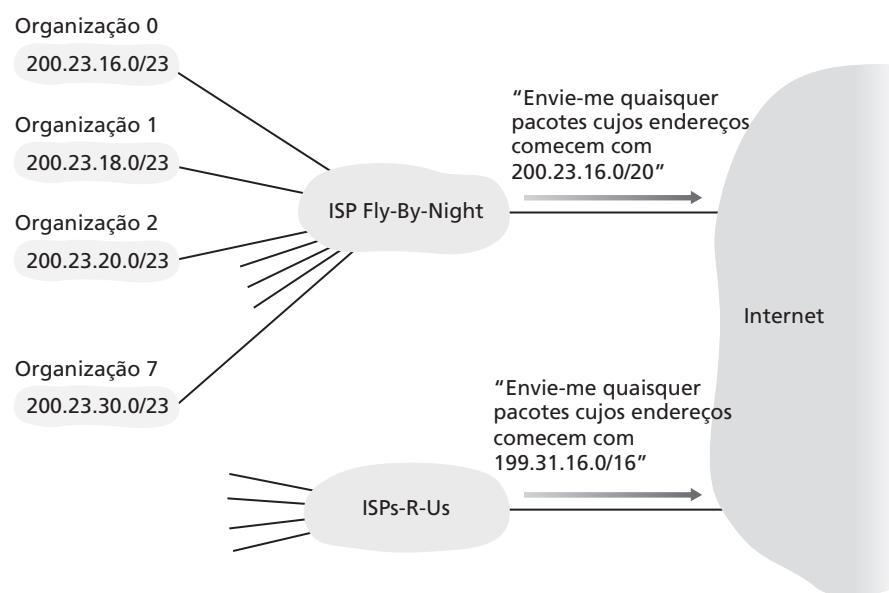


Figura 4.18 Endereçamento hierárquico e agregação de rotas

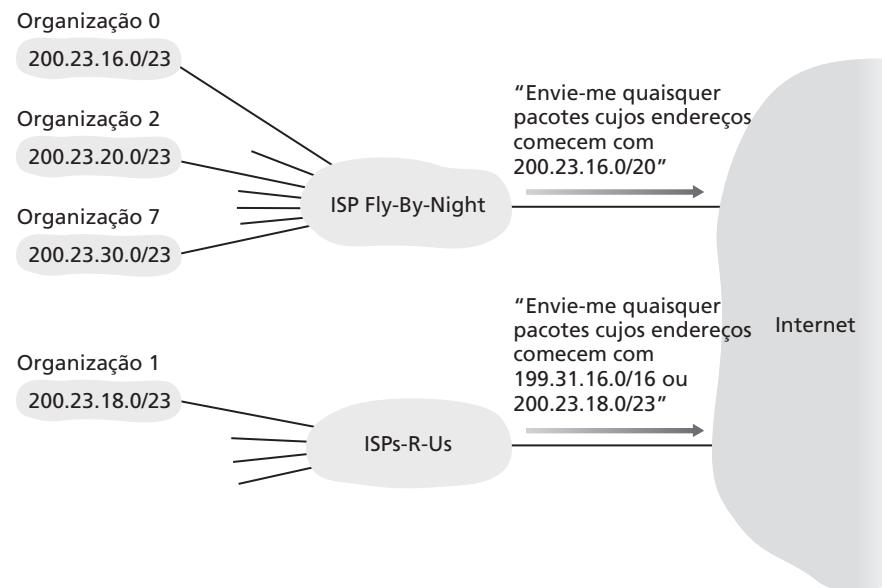


Figura 4.19 ISPs-R-U_s tem uma rota mais específica para a Organização 1

do ISPs-R-U_s. Mas essa é uma solução dispendiosa, e a Organização 1 poderia muito bem preferir mudar para uma outra subsidiária no futuro. A solução normalmente adotada é a Organização 1 manter seus endereços IP em 200.23.18.0/23. Nesse caso, como mostra a Figura 4.19, o ISP Fly-By-Night continua a anunciar o bloco de endereços 200.23.16.0/20 e o ISPs-R-U_s continua a anunciar 199.31.16.0/16. Contudo, o ISPs-R-U_s agora anuncia também o bloco de endereços para a Organização 1, 200.23.18.0/23. Quando outros roteadores da Internet virem os blocos de endereços 200.23.16.0/20 (do ISP Fly-By-Night) e 200.23.18.0/23 (do ISPs-R-U_s) e quiserem rotear para um endereço no bloco 200.23.18.0/23, usarão a regra de compatibilização com o prefixo mais longo (veja Seção 4.2.2) e rotearão para o ISP-R-U_s, já que ele anuncia o prefixo de endereço mais longo (mais específico) que combina com o endereço de destino.

Obtenção de um endereço de hospedeiro: o Protocolo de Configuração Dinâmica de Hospedeiros (DHCP)

Tão logo tenha obtido um bloco de endereços, uma organização pode atribuir endereços IP individuais às interfaces de hospedeiros e roteadores em sua organização. Normalmente um administrador de sistemas configurará manualmente os endereços IP no roteador (muitas vezes remotamente, com uma ferramenta de gerenciamento de rede). Os endereços dos hospedeiros podem também ser configurados manualmente, mas essa tarefa é feita geralmente usando o Protocolo de Configuração Dinâmica de Hospedeiros (DHCP) [RFC 2131]. O DHCP permite que um hospedeiro obtenha (ser alocado) um endereço IP automaticamente. Um administrador de rede pode configurar o DHCP de modo que um determinado hospedeiro receba o mesmo endereço IP toda vez que se conectar à rede, ou um hospedeiro pode receber um endereço IP temporário sempre que se conectar à rede. Além de receber um endereço IP temporário, o DHCP também permite que o hospedeiro descubra informações adicionais, como a máscara de sub-rede, o endereço do primeiro roteador (comumente chamado de porta de comunicação padrão — *default gateway*) e o endereço de seu servidor DNS local.

Por causa de sua capacidade de automatizar os aspectos relativos à rede da conexão de um hospedeiro à rede, o DHCP é comumente denominado um **protocolo plug and play**. Essa capacidade o torna *muito* atraente para o administrador de rede que, caso contrário, teria de executar essas tarefas manualmente! O DHCP também está conquistando ampla utilização em redes residenciais de acesso à Internet e em LANs sem fio, nas quais hospedeiros entram e saem da rede com frequência. Considere, por exemplo, um estudante que leva seu laptop do seu quarto para a biblioteca, para a sala de aula. É provável que ele se conectará a uma nova sub-rede em cada

um desses lugares e, por conseguinte, precisará de um novo endereço IP em cada um deles. O DHCP é ideal para essa situação, pois há muitos usuários em trânsito e os endereços são utilizados apenas por um tempo limitado.

O DHCP é, de maneira semelhante, útil em redes domésticas de acesso. Como exemplo, considere um ISP residencial que tem dois mil clientes, porém, nunca mais de 400 deles estão on-line ao mesmo tempo. O ISP não precisa de um bloco de dois mil endereços para administrar todos os seus dois mil clientes — usando um servidor DHCP para designar endereços dinamicamente, ele precisa de um bloco de apenas 512 endereços (por exemplo, um bloco da forma a.b.c.d/23). À medida que hospedeiros entram e saem da rede, o servidor DHCP precisa atualizar sua lista de endereços IP disponíveis. Toda vez que um hospedeiro se conectar à rede, o servidor DHCP designa a ele um endereço arbitrário do seu reservatório corrente de endereços disponíveis; toda vez que um hospedeiro sair, o endereço é devolvido ao reservatório.

O DHCP é um protocolo cliente-servidor. Normalmente o cliente é um hospedeiro recém-chegado que quer obter informações sobre configuração da rede, incluindo endereço IP, para si mesmo. Em um caso mais simples, cada sub-rede (no sentido de endereçamento na Figura 4.17) terá um servidor DHCP. Se nenhum servidor estiver presente na sub-rede, é necessário um agente relé DHCP (normalmente um roteador) que sabe o endereço de um servidor DHCP para tal rede. A Figura 4.20 ilustra um servidor DHCP conectado à rede 223.1.2/24, servindo o roteador de agente relé para clientes que chegam conectados às sub-redes 223.1.1/24 e 223.1.3/24. Em nossa discussão abaixo, admitiremos que um servidor DHCP está disponível na sub-rede.

Para um hospedeiro recém-chegado, o protocolo DHCP é um processo de quatro etapas, como mostrado na Figura 4.21 para a configuração de rede mostrada na Figura 4.20. Nesta figura, *y iaddr* (como em “seu endereço da Internet) indica que o endereço está sendo alocado para um cliente recém-chegado. As quatro etapas são:

Descoberta do servidor DHCP. A primeira tarefa de um hospedeiro recém-chegado é encontrar um servidor DHCP com quem interagir. Isso é feito utilizando uma mensagem de descoberta DHCP, a qual o cliente envia dentro de um pacote UDP para a porta 67. O pacote UDP é envolvido em um datagrama IP. Mas para quem esse datagrama deve ser enviado? O hospedeiro não sabe o endereço IP da rede à qual está se conectando, muito menos o endereço de um servidor DHCP para essa rede. Desse modo, o

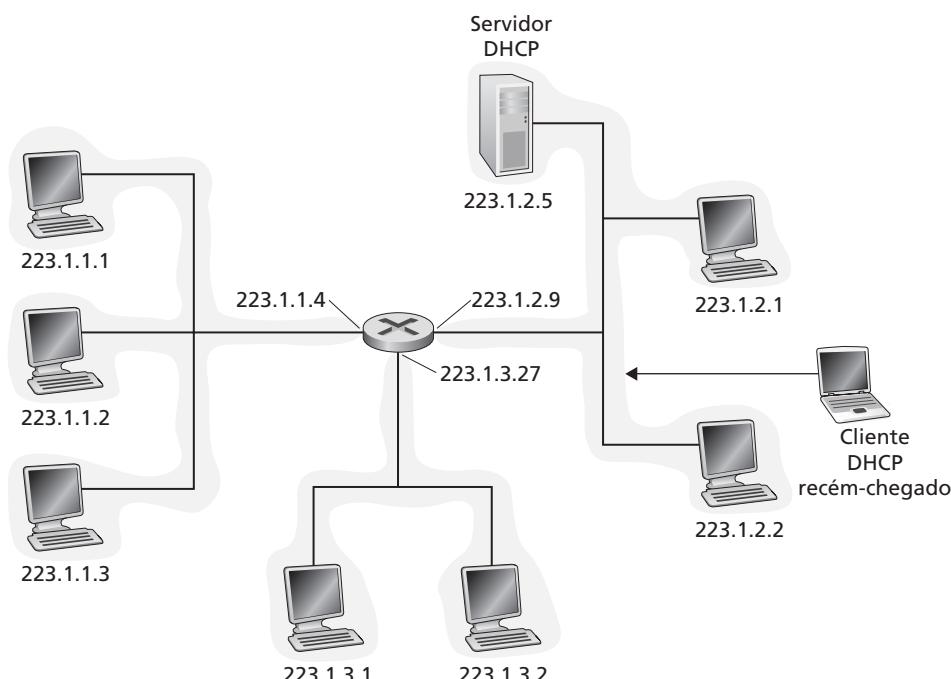


Figura 4.20 Tradução de endereço de rede

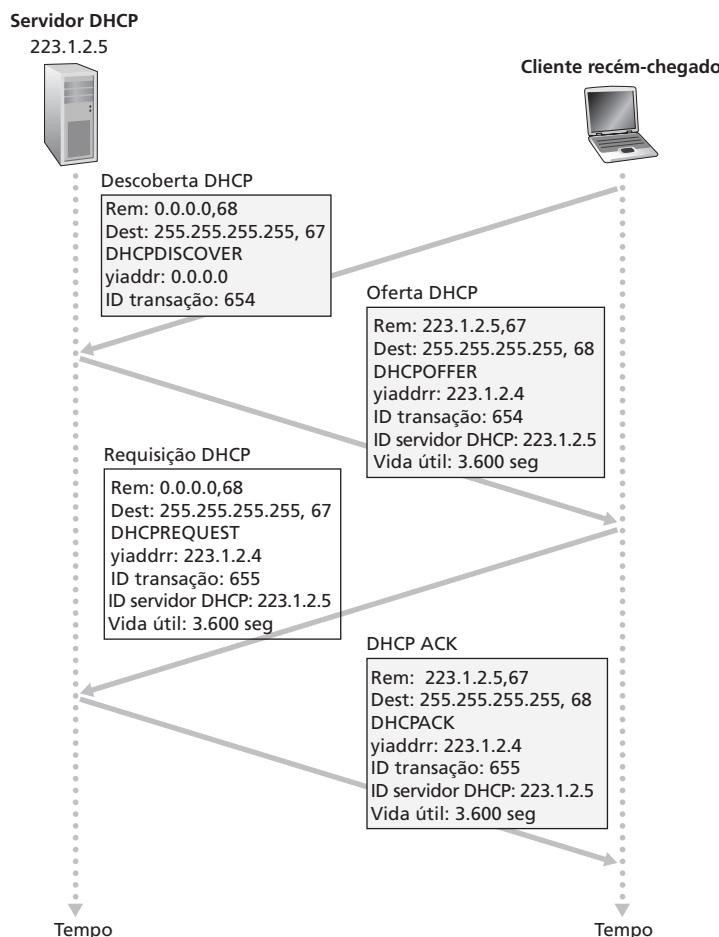


Figura 4.21 Intereração cliente-servidor DHCP

cliente DHCP cria um datagrama IP contendo sua mensagem de descoberta DHCP com o endereço IP de destino de transmissão 255.255.255.255 e um endereço IP destinatário “desse hospedeiro” 0.0.0.0. O cliente DHCP transmite o datagrama IP à camada de enlace que, então, transmite esse quadro para todos os nós conectados à sub-rede (discutiremos os detalhes sobre transmissão da camada de enlace na Seção 5.4).

Oferta(s) dos servidores DHCP. Um servidor DHCP que recebe uma mensagem de descoberta DHCP responde ao cliente com uma mensagem de oferta DHCP, transmitida a todos os nós presentes na sub-rede, novamente utilizando o endereço IP de transmissão 255.255.255.255. (Você pode pensar por que essa resposta do servidor pode ser transmitida). Como diversos servidores DHCP podem estar presentes na sub-rede, o cliente pode se dar ao luxo de escolher dentre as muitas ofertas. Cada mensagem de oferta do servidor contém o ID de transação da mensagem de descoberta recebida, o endereço IP proposto para o cliente, a máscara da rede e o tempo de concessão do endereço IP — o tempo pelo qual o endereço IP será válido. É comum o servidor definir o tempo de concessão para várias horas ou dias [Droms, 2002].

Solicitação DHCP. O cliente recém-chegado escolherá dentre uma ou mais ofertas do servidor e responderá à sua oferta selecionada com uma mensagem de solicitação DHCP, repetindo os parâmetros de configuração.

DHCP ACK. O servidor responde a mensagem de requisição DHCP com uma mensagem DHCP ACK, confirmando os parâmetros requisitados.

Uma vez que o cliente recebe o DHCP ACK, a interação é concluída e ele pode usar o endereço IP alocado pelo DHCP pelo tempo de concessão. Caso o cliente queira usar seu endereço após a expiração da concessão, o DHCP também fornece um mecanismo que permite que o cliente renove sua concessão sobre um endereço IP.

O valor da capacidade *plug and play* do DHCP é clara, considerando o fato de que a alternativa é configurar manualmente um endereço IP do hospedeiro. Considere o aluno que se locomove da sala de aula para a biblioteca até seu dormitório com um laptop, entra em uma nova sub-rede, obtendo, assim, um novo endereço IP em cada local. É inimaginável que um administrador de sistema tivesse de reconfigurar laptops em cada local, e poucos alunos (exceto os que estão tendo aula de rede de computadores!) teriam a habilidade de configurar seus laptops manualmente. Entretanto, partindo de um aspecto de mobilidade, o DHCP possui desvantagens. Uma vez que um novo endereço IP é obtido de um novo DHCP toda vez que um nó se conecta a uma nova sub-rede, uma conexão TCP para uma aplicação remota não pode ser mantida pois o nó móvel se locomove entre as sub-redes. No Capítulo 6, analisaremos o IP móvel — uma extensão recente para a infraestrutura IP que permite que um nó móvel utilize um único endereço permanente à medida que se locomove entre as sub-redes. Mais detalhes sobre o DHCP podem ser encontrados em [Droms, 2002] e [dhc, 2009]. Uma implementação de referência de código-fonte aberto do DHCP está disponível em Internet System Consortium [ISC, 2009].

Tradução de endereços na rede (NAT)

Dada a nossa discussão sobre endereços de Internet e o formato do datagrama IPv4, agora estamos bem cientes de que todo equipamento que utiliza IP precisa de um endereço IP. Isso aparentemente significa que, com a proliferação de sub-redes de pequenos escritórios e de escritórios residenciais (*small office home office* — SOHO), sempre que um desses escritórios quiser instalar uma LAN para conectar várias máquinas, o ISP precisará alocar uma faixa de endereços para atender a todas as máquinas que usam IP do escritório em questão. Se a sub-rede ficasse maior (por exemplo, as crianças da casa não somente têm seus próprios computadores, mas também compraram PDAs, telefones com IP e jogos Game Boys em rede), seria preciso alocar um bloco de endereços maior. Mas, e se o ISP já tiver alocado as porções contíguas da faixa de endereços utilizada atualmente por esse escritório residencial? E, antes de mais nada, qual é o proprietário típico de uma residência que quer (ou precisaria) saber como administrar endereços IP? Felizmente, há uma abordagem mais simples da alocação de endereços que vem conquistando uma utilização cada vez mais ampla nesses tipos de cenários: a **tradução de endereços de rede (network address translation — NAT)** [RFC 2663; RFC 3022].

A Figura 4.22 mostra a operação de um roteador que utiliza NAT. Esse roteador, que fica na residência do cidadão, tem uma interface que faz parte da rede residencial do lado direito da Figura 4.22. O endereçamento dentro da rede residencial é exatamente como vimos anteriormente — todas as quatro interfaces da rede têm o mesmo endereço de sub-rede, 10.0.0.0/24. O espaço de endereço 10.0.0.0/8 é uma das três porções do espaço de endereço IP reservado pelo [RFC 1918] para uma rede privada, ou um **domínio** com endereços privados, tal como a rede residencial da Figura 4.22. Um domínio com endereços privados refere-se a uma rede cujos endereços somente têm significado para equipamentos pertencentes àquela rede. Para ver por que isso é importante, considere o fato de haver centenas de milhares de redes residenciais, muitas delas utilizando o mesmo espaço de endereço 10.0.0.0/24. Equipamentos que pertencem a uma determinada rede residencial podem enviar pacotes entre si utilizando o endereçamento 10.0.0.0/24. Contudo, é claro que pacotes repassados da rede residencial para a Internet global não podem usar esses endereços (nem como endereço de fonte, nem como endereço de destino) porque há centenas de milhares de redes utilizando esse bloco de endereços. Isto é, os endereços 10.0.0.0/24 somente podem ter significado dentro daquela rede residencial. Mas, se endereços privados têm significado apenas dentro de uma dada rede, como o endereçamento é administrado quando pacotes são recebidos ou enviados para a Internet global, onde os endereços são necessariamente exclusivos? A resposta será dada pelo estudo da NAT. O roteador que usa NAT *não parece* um roteador para o mundo externo, pois se comporta como um equipamento único com um único endereço IP. Na Figura 4.22, todo o tráfego que sai do roteador residencial para a Internet tem um endereço IP de fonte 138.76.29.7, e todo o tráfego que entra nessa rede tem de ter endereço de destino 138.76.29.7. Em essência, o roteador que usa NAT está ocultando do mundo exterior os detalhes da rede residencial. (A propósito, você talvez esteja imaginando onde os computadores de redes residenciais obtêm seus endereços e onde o roteador obtém seu endereço IP exclusivo. A resposta é quase sempre a mesma — DHCP!)

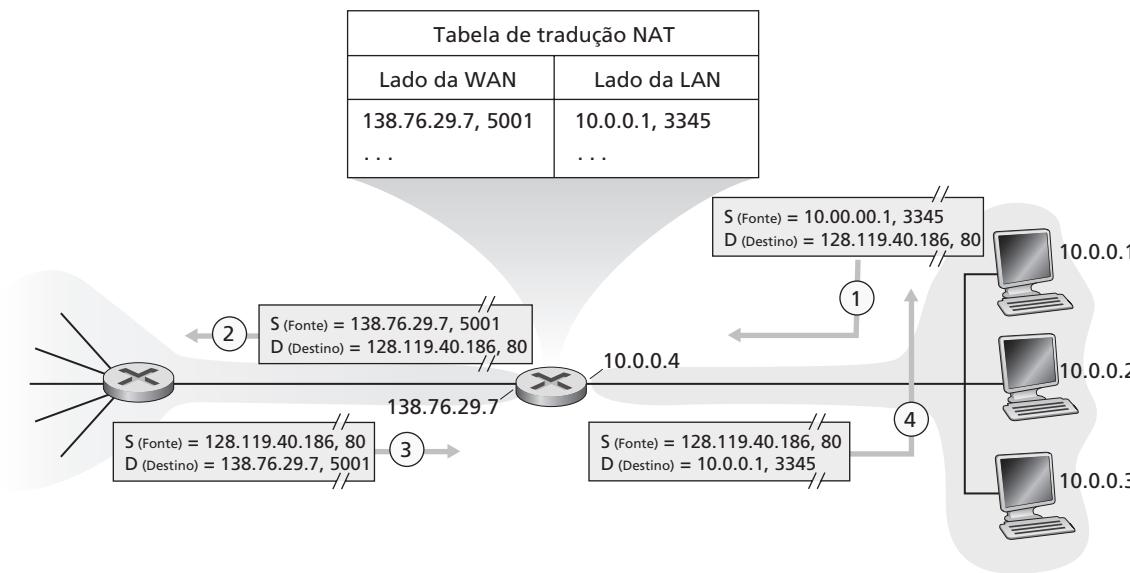


Figura 4.22 Tradução de endereços de rede

O roteador obtém seu endereço do servidor DHCP do ISP e roda um servidor DHCP para fornecer endereços a computadores que estão dentro do espaço de endereços NAT da rede residencial controlado pelo DHCP).

Se todos os datagramas que chegam ao roteador NAT provenientes da WAN tiverem o mesmo endereço IP de destino (especificamente, o endereço da interface do roteador NAT do lado da WAN), então como o roteador sabe para qual hospedeiro interno deve repassar um dado datagrama? O truque é utilizar uma **tabela de tradução NAT** no roteador NAT e incluir nos registros da tabela números de portas, bem como endereços IP.

Considere o exemplo da Figura 4.22. Suponha que um usuário que está utilizando o hospedeiro 10.0.0.1 da rede residencial requisite uma página Web de algum servidor Web (porta 80) cujo endereço IP é 128.119.40.186. O hospedeiro 10.0.0.1 escolhe o número de porta de fonte (arbitrário) 3345 e envia os datagramas para dentro da LAN. O roteador NAT recebe o datagrama, gera um novo número de porta de fonte, 5001, para o datagrama, substitui o endereço IP da fonte por seu endereço IP do lado da WAN, 138.76.29.7, e substitui o número de porta de fonte original, 3345, pelo novo número de porta de fonte, 5001. Ao gerar um novo número de porta de fonte, o roteador NAT pode selecionar qualquer número de porta de fonte que não esteja correntemente na tabela de tradução NAT. (Note que, como o comprimento de um campo de número de porta é 16 bits, o protocolo NAT pode suportar mais de 60 mil conexões simultâneas com um único endereço IP do lado da WAN para o roteador!). A NAT no roteador também adiciona um registro à sua tabela de tradução NAT. O servidor Web, totalmente alheio ao fato de que o datagrama que está chegando com uma requisição HTTP foi manipulado pelo roteador NAT, responde com um datagrama cujo endereço de destino é o endereço IP do roteador NAT, e cujo número de porta de destino é 5001. Quando esse datagrama chega ao roteador NAT, o roteador indexa a tabela de tradução NAT usando o endereço IP de destino e o número de porta de destino para obter o endereço IP (10.0.0.1) e o número de porta de destino (3345) adequados para o browser na rede residencial. O roteador então reescreve o endereço de destino e o número de porta de destino do datagrama e o repassa para a rede residencial.

A NAT conquistou ampla aceitação nos últimos anos. Mas devemos mencionar que muitos puristas da comunidade da IETF têm grandes restrições à NAT. Em primeiro lugar, argumentam, a finalidade dos números de portas é endereçar processos, e não hospedeiros. (Realmente, a violação dessa regra pode causar problemas para servidores que rodam em redes residenciais, pois, como vimos no Capítulo 2, processos servidores esperam pela chegada de requisições em números de portas bem conhecidos.) Em segundo lugar, argumentam que roteadores devem processar pacotes somente até a camada 3. Discutem, em terceiro lugar, que o protocolo NAT viola o argumento denominado fim a fim; isto é, hospedeiros devem falar diretamente uns com os outros, sem a interferência

de nós que modifiquem endereços IP e números de portas. E, em quarto lugar, argumentam que deveríamos usar o IPv6 (veja Seção 4.4.4) para resolver a escassez de endereços IP, e não tentar resolver o problema imprudentemente com uma solução temporária como a NAT. Mas, gostemos ou não, a NAT tornou-se um componente importante da Internet. Outro problema importante da NAT é que ela interfere com aplicações P2P, incluindo aplicações de compartilhamento de arquivos P2P e aplicações de voz sobre IP. Lembre-se de que, no Capítulo 2, dissemos que, em uma aplicação P2P, qualquer par A participante deve poder iniciar uma conexão TCP com qualquer outro par B participante. A essência do problema é que, se o par B estiver por trás de uma NAT, não poderá agir como um servidor e aceitar conexões TCP. Como veremos nos problemas de fixação, essa questão da NAT pode ser contornada se o par A se conectar primeiramente com o par B através de um par C intermediário que não está por trás de uma NAT e com o qual o par B tenha estabelecido uma conexão TCP que está em curso. Então, o par A pode solicitar ao par B, por intermédio do par C, que inicie uma conexão TCP diretamente com ele. Uma vez estabelecida a conexão P2P TCP direta entre os pares A e B, os dois podem trocar mensagens ou arquivos. Essa solução, denominada **reversão de conexão**, na verdade é usada por muitas aplicações P2P para a NAT traversal. Se ambos os pares, A e B, estiverem por trás de suas próprias NATs, a situação é um pouco mais complicada, mas pode ser resolvida utilizando relés para aplicação, como vimos em relés do Skype, no Capítulo 2.

UPnP

A travessia da NAT traversal está sendo cada vez mais fornecida pelo *Universal Plug and Play* (UPnP), um protocolo que permite que um hospedeiro descubra e configure uma NAT próxima [UPnP Forum, 2009]. O UPnP exige que o hospedeiro e a NAT sejam compatíveis com ele. Com respeito ao UPnP, uma aplicação que roda em um hospedeiro pode solicitar um mapeamento da NAT entre seus (*endereço IP particular, número de porta particular*) e (*endereço IP público, número de porta pública*) para algum número de porta pública requisitado. Se a NAT aceitar a requisição e criar um mapeamento, então nós externos podem iniciar conexões TCP para (*endereço IP público, número de porta pública*). Além disso, o UPnP deixa a aplicação saber o valor de (*endereço IP público, número de porta pública*), de modo que ela possa anunciar ao mundo externo.

Como exemplo, suponha que seu hospedeiro, localizado atrás de uma NAT com o UPnP habilitado, possua o endereço IP particular 10.0.0.1 e esteja rodando o BitTorrent na porta 3345. Suponha também que o endereço IP público da NAT seja 138.76.29.7. Sua aplicação BitTorrent naturalmente quer poder aceitar conexões vindas de outros hospedeiros, para que blocos sejam adquiridos através delas. Para isso, a aplicação BitTorrent, em seu hospedeiro, solicita a NAT para criar uma “abertura” que mapeia (10.0.0.1, 3345) para (138.76.29.7, 5001). (A porta pública número 5001 é escolhida pela aplicação.) A aplicação BitTorrent em seu hospedeiro poderia, também, anunciar ao seu rastreador que está disponível em (138.76.29.7, 5001]. Desta maneira, um hospedeiro externo que está rodando o BitTorrent pode contatar o rastreador e descobrir que sua aplicação BitTorrent está rodando em (138.76.29.7, 5001). O hospedeiro externo pode enviar um pacote SYN TCP para (138.76.29.7, 5001). Quando a NAT receber o pacote SYN, ela alterará o endereço IP destinatário e o número da porta no pacote para (10.0.0.1, 3345) e encaminhará o pacote através da NAT.

Em resumo, o UPnP permite que hospedeiros externos iniciem sessões de comunicação com hospedeiros que utilizam NAT, usando TCP ou UDP. As NATs foram inimigas das aplicações P2P por um longo tempo; o UPnP, que apresenta uma solução eficaz e potente para a travessia da NAT transversal, pode ser o salvador. Nossa discussão sobre NAT e UPnP foi necessariamente breve. Para obter mais detalhes sobre a NAT, consulte [Huston, 2004; Cisco NAT, 2009].

4.4.3 Protocolo de Mensagens de Controle da Internet (ICMP)

Lembre-se de que a camada de rede da Internet tem três componentes principais: o protocolo IP, discutido na seção anterior; os protocolos de roteamento da Internet (entre eles RIP, OSPF e BGP), que serão estudados na Seção 4.6; e o ICMP, objeto desta seção.

O ICMP, especificado no [RFC 792], é usado por hospedeiros e roteadores para comunicar informações de camada de rede entre si. A utilização mais comum do ICMP é para comunicação de erros. Por exemplo, ao rodar

uma sessão Telnet, FTP ou HTTP, é possível que você já tenha encontrado uma mensagem de erro como “Rede de destino inalcançável”. Essa mensagem teve sua origem no ICMP. Em algum ponto, um roteador IP não conseguiu descobrir um caminho para o hospedeiro especificado em sua aplicação Telnet, FTP ou HTTP. O roteador criou e enviou uma mensagem ICMP do tipo 3 a seu hospedeiro indicando o erro.

O ICMP é frequentemente considerado parte do IP, mas, em termos de arquitetura, está logo acima do IP, pois mensagens ICMP são carregadas dentro de datagramas IP. Isto é, mensagens ICMP são carregadas como carga útil IP, exatamente como segmentos TCP ou UDP, que também são carregados como carga útil IP. De maneira semelhante, quando um hospedeiro recebe um datagrama IP com ICMP especificado como protocolo de camada superior, ele demultiplexa o conteúdo do datagrama para ICMP, exatamente como demultiplexaria o conteúdo de um datagrama para TCP ou UDP.

Mensagens ICMP têm um campo de tipo e um campo de código. Além disso, contêm o cabeçalho e os primeiros 8 bytes do datagrama IP que causou a criação da mensagem ICMP em primeiro lugar (de modo que o remetente pode determinar o datagrama que causou o erro). Alguns tipos de mensagens ICMP selecionadas são mostradas na Figura 4.23. Note que mensagens ICMP não são usadas somente para sinalizar condições de erro.

O conhecido programa ping envia uma mensagem ICMP do tipo 8 código 0 para o hospedeiro especificado. O hospedeiro de destino, ao ver a solicitação de eco, devolve uma resposta de eco ICMP do tipo 0 código 0. A maior parte das implementações de TCP/IP suportam o servidor ping diretamente no sistema operacional; isto é, o servidor não é um processo. O Capítulo 11 de [Stevens, 1990] fornece o código-fonte para o programa ping cliente. Note que o programa cliente tem de ser capaz de instruir o sistema operacional para que ele gere uma mensagem ICMP do tipo 8 código 0.

Uma outra mensagem ICMP interessante é a de redução de fonte. Essa mensagem é pouco usada na prática. Sua finalidade original era realizar controle de congestionamento — permitir que um roteador congestionado enviasse uma mensagem ICMP de redução de fonte a um hospedeiro para obrigar esse hospedeiro a reduzir sua velocidade de transmissão. Vimos no Capítulo 3 que o TCP tem seu próprio mecanismo de controle de congestionamento, que funciona na camada de transporte sem usar realimentação da camada de rede tal como a mensagem ICMP de redução de fonte.

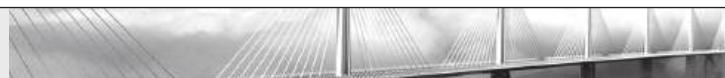
No Capítulo 1, apresentamos o programa Traceroute, que nos permite acompanhar a rota de um hospedeiro a qualquer outro hospedeiro no mundo. O interessante é que o Traceroute é implementado com mensagens ICMP. Para determinar os nomes e endereços de roteadores entre a fonte e o destino, o Traceroute da fonte envia

Tipo de mensagem ICMP	Código	Descrição
0	0	resposta de eco (para ping)
3	0	rede de destino inalcançável
3	1	hospedeiro de destino inalcançável
3	2	protocolo de destino inalcançável
3	3	porta de destino inalcançável
3	6	rede de destino desconhecida
3	7	hospedeiro de destino desconhecido
4	0	redução da fonte (controle de congestionamento)
8	0	solicitação de eco
9	0	anúncio do roteador
10	0	descoberta do roteador
11	0	TTL expirado
12	0	cabeçalho IP inválido

Figura 4.23 Tipos de mensagens ICMP

uma série de datagramas comuns ao destino. O primeiro desses datagramas tem um TTL de 1, o segundo tem um TTL de 2, o terceiro tem um TTL de 3 e assim por diante. A fonte também aciona temporizadores para cada um dos datagramas. Quando o enésimo datagrama chega ao enésimo roteador, o enésimo roteador observa que o TTL do datagrama acabou de expirar. Segundo as regras do protocolo IP, o roteador descarta o datagrama e envia uma mensagem ICMP de aviso à fonte (tipo 11 código 0). Essa mensagem de aviso inclui o nome do roteador e seu endereço IP. Quando chega à fonte, a mensagem obtém, do temporizador, o tempo de viagem de ida e volta e, da mensagem ICMP, o nome e o endereço IP do enésimo roteador.

Como uma fonte Traceroute sabe quando parar de enviar segmentos UDP? Lembre-se de que a fonte incrementa o campo do TTL para cada datagrama que envia. Assim, um dos datagramas finalmente conseguirá chegar ao hospedeiro de destino. Como esse datagrama contém um segmento UDP com um número de porta improvável, o hospedeiro de destino devolve à fonte uma mensagem ICMP indicando que a porta não pôde ser alcançada (mensagem tipo 3, código 3). Quando recebe essa mensagem ICMP particular, o hospedeiro da fonte sabe que não precisa enviar mais pacotes de sondagem. (Na verdade, o programa Traceroute padrão envia conjuntos de três pacotes com o mesmo TTL; assim, o Traceroute provê três resultados para cada TTL.)



Segurança em foco

Inspecionando datagramas: firewalls e sistemas de detecção de intrusão

Suponha que você deva administrar uma rede doméstica, departamental, universitária ou corporativa. Os atacantes, sabendo a faixa de endereço IP da sua rede, podem enviar facilmente datagramas IP para endereços da sua faixa. Esses datagramas são capazes de fazer todos os tipos de coisas desonestas, inclusive mapear sua rede, fazendo seu reconhecimento (ping sweep) e varredura de portas, prejudicar hospedeiros vulneráveis com pacotes defeituosos, inundar servidores com milhares de pacotes ICMP e infectar hospedeiros incluindo malwares nos pacotes. Como administrador de rede, o que você vai fazer com todos esses vilões capazes de enviar pacotes maliciosos à sua rede? Dois consagrados mecanismos de defesa para ataques maliciosos de pacote são os firewalls e os sistemas de setecção de intrusão (IDSs).

Como administrador de rede, você deve, primeiramente, tentar instalar um firewall entre sua rede e a Internet. (A maioria dos roteadores de acesso, hoje, possuem capacidade para firewall.) Os firewalls inspecionam o datagrama e campos do cabeçalho do segmento, evitando que datagramas suspeitos entrem na rede interna. Um firewall pode estar configurado, por exemplo, para bloquear todos os pacotes de requisição de eco ICMP, impedindo que um atacante realize um reconhecimento de rede por sua faixa de endereço IP. Os firewalls podem também bloquear pacotes baseando-se nos endereços IP remetente e destinatário e em números de porta. Além disso, os firewalls podem ser configurados para rastrear conexões TCP, permitindo a entrada somente de datagramas que pertençam a conexões aprovadas.

Uma proteção adicional pode ser fornecida com um IDS. Um IDS, normalmente localizado no limite de rede, realiza "uma inspeção profunda de pacote", examinando não apenas campos de cabeçalho como cargas úteis no datagrama (incluindo dados da camada de aplicação). Um IDS possui um banco de dados de assinaturas de pacote à medida que novos ataques são descobertos. Enquanto os pacotes percorrem o IDS, este tenta compatibilizar campos de cabeçalhos e cargas úteis às assinaturas em seu banco de dados de assinaturas. Se tal compatibilização é encontrada, cria-se um alerta. Os sistemas de prevenção de intrusão (IPS) são semelhantes a um IDS, exceto pelo fato de eles bloquearem pacotes além de criar alertas. No Capítulo 8, exploraremos mais detalhadamente firewalls e IDSs.

Os firewalls e os IDSs são capazes de proteger sua rede de todos os ataques? Obviamente, a resposta é não, visto que os atacantes encontram novas formas de ataques continuamente para os quais as assinaturas ainda não estão disponíveis. Mas os firewalls e os IDSs baseados em assinaturas tradicionais são úteis para proteger sua rede de ataques conhecidos.



Desse modo, o hospedeiro da fonte fica a par do número e das identidades de roteadores que estão entre ele e o hospedeiro de destino e o tempo de viagem de ida e volta entre os dois hospedeiros. Note que o programa cliente Traceroute tem de ser capaz de instruir o sistema operacional para que este gere datagramas UDP com valores específicos de TTL. Também tem de poder ser avisado por seu sistema operacional quando chegam mensagens ICMP.

Agora que você entende como o Traceroute funciona, é provável que queira voltar e brincar um pouco com ele.

4.4.4 IPv6

No começo da década de 1990, a IETF iniciou um esforço para desenvolver o sucessor do protocolo IPv4. Uma motivação primária para esse esforço foi o entendimento de que o espaço de endereços IP de 32 bits estava começando a escassear, com novas sub-redes e nós IP sendo anexados à Internet (e ainda recebendo endereços IP exclusivos) a uma velocidade estonteante. Para atender a essa necessidade de maior espaço para endereços IP, um novo protocolo IP, o IPv6, foi desenvolvido. Os projetistas do IPv6 também aproveitaram essa oportunidade para ajustar e ampliar outros aspectos do IPv4 com base na experiência operacional acumulada sobre esse protocolo.

O momento em que todos os endereços IPv4 estariam alocados (e, por conseguinte, mais nenhuma sub-rede poderia ser ligada à Internet) foi objeto de considerável debate. Os dois líderes do grupo de trabalho Expectativa de Tempo de Vida dos Endereços (Address Lifetime Expectations) da IETF estimaram que os endereços se esgotariam em 2008 e 2018, respectivamente [Solensky, 1996]. Uma análise mais recente [Huston, 2008] estima esse esgotamento em torno de 2010. Em 1996, o Registro Americano para Números da Internet (American Registry for Internet Numbers — ARIN) informou que já tinham sido alocados todos os endereços da classe A do IPv4, 62 por cento dos endereços da classe B e 37 por cento dos endereços da classe C [ARIN, 1996]. Para um relatório recente sobre o espaço de alocação do endereço IPv4, consulte [Hain, 2005]. Embora essas estimativas e números sugerissem que havia um tempo considerável até que o espaço de endereços IPv4 fosse exaurido, ficou claro que seria necessário um tempo expressivo para disponibilizar uma nova tecnologia em escala tão gigantesca. Assim, foi dado início ao esforço denominado Próxima Geração do IP (Next Generation IP — IPng) [Bradner, 1996; RFC 1752]. O resultado desse esforço foi a especificação IP versão 6 (IPv6) [RFC 2460]. (Uma pergunta que sempre é feita é o que aconteceu com o IPv5. Foi proposto inicialmente que o protocolo ST-2 se tornasse o IPv5, porém, mais tarde, este protocolo foi descartado em favor do RSVP, que discutiremos no Capítulo 7.)

Excelentes fontes de informação sobre o IPv6 podem ser encontradas em [Huitema, 1998, IPv6 2009].

Formato do datagrama IPv6

O formato do datagrama IPv6 é mostrado na Figura 4.24. As mudanças mais importantes introduzidas no IPv6 ficam evidentes no formato do datagrama:

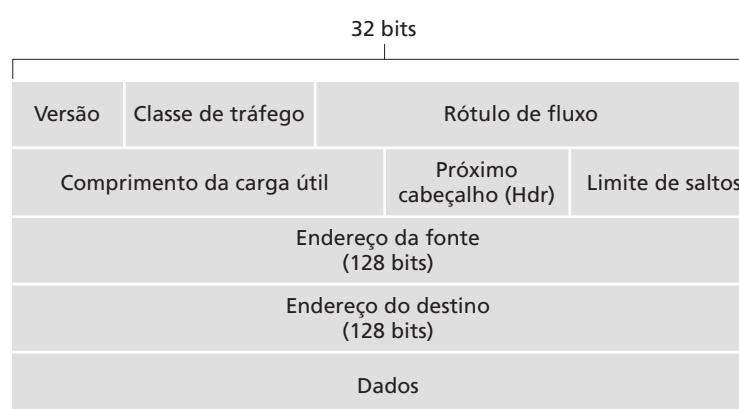


Figura 4.24 Formato do datagrama IPv6

Capacidade de endereçamento expandida. O IPv6 aumenta o tamanho do endereço IP de 32 bits para 128 bits. Isso garante que o mundo não ficará sem endereços IP. Agora cada grão de areia do planeta pode ter um endereço IP. Além dos endereços *multicast* e *unicast*, o IPv6 introduziu um novo tipo de endereço, denominado **endereço anycast**, que permite que um datagrama seja entregue a qualquer hospedeiro de um grupo. (Essa característica poderia ser usada, por exemplo, para enviar uma mensagem HTTP GET ao site mais próximo de um conjunto de sites espelhados que contenham um dado documento.)

Cabeçalho aprimorado de 40 bytes. Como discutiremos adiante, vários campos IPv4 foram descartados ou tornaram-se opcionais. O cabeçalho de comprimento fixo de 40 bytes resultante permite processamento mais veloz do datagrama IP. Uma nova codificação de opções permite um processamento de opções mais flexível.

Rotulação de fluxo e prioridade. O IPv6 tem uma definição dúbia de **fluxo**. O RFC 1752 e o RFC 2460 declararam que isso permite “rotular pacotes que pertencem a fluxos particulares para os quais o remetente requisita tratamento especial, tal como um serviço de qualidade não padrão ou um serviço de tempo real”. Por exemplo, a transmissão de áudio e vídeo poderia ser tratada como um fluxo. Por outro lado, as aplicações mais tradicionais, como transferência de arquivos e e-mail, poderiam não ser tratadas como fluxos. É possível que o tráfego carregado por um usuário de alta prioridade (digamos, alguém que paga por um serviço melhor de tráfego) seja também tratado como um fluxo. O que fica claro, contudo, é que os projetistas do IPv6 preveem a possível necessidade de poder diferenciar os fluxos, mesmo que o exato significado de fluxo ainda não tenha sido determinado. O cabeçalho IPv6 também tem um campo de 8 bits para classe de tráfego. Esse campo, como o campo TOS do IPv4, pode ser usado para dar prioridade a certos datagramas dentro de um fluxo ou a datagramas de certas aplicações (por exemplo, pacotes ICMP) em relação a datagramas de outras aplicações (por exemplo, notícias pela rede).

Como foi observado anteriormente, uma comparação entre as Figuras 4.24 e 4.13 revela uma estrutura mais simples e mais aprimorada para o datagrama IPv6. Os seguintes campos são definidos no IPv6:

Versão. Esse campo de 4 bits identifica o número da versão do IP. Não é surpresa que o IPv6 tenha o valor 6 nesse campo. Note que colocar 4 nesse campo não cria um datagrama IPv4 válido. (Se criasse, a vida seria bem mais simples — veja a discussão mais adiante, referente à transição do IPv4 para o IPv6.)

Classe de tráfego. Esse campo de 8 bits tem função semelhante à do campo TOS que vimos no IPv4.

Rótulo de fluxo. Como discutimos antes, esse campo de 20 bits é usado para identificar um fluxo de datagramas.

Comprimento da carga útil. Esse valor de 16 bits é tratado como um número inteiro sem sinal que dá o número de bytes no datagrama IPv6 que se segue ao pacote do cabeçalho, que tem tamanho fixo de 40 bytes.

Próximo cabeçalho. Esse campo identifica o protocolo ao qual o conteúdo (campo de dados) desse datagrama será entregue (por exemplo, TCP ou UDP). Usa os mesmos valores do campo de protocolo no cabeçalho IPv4.

Limite de saltos. O conteúdo desse campo é decrementado de um para cada roteador que repassa o datagrama. Se a contagem do limite de saltos chegar a zero, o datagrama será descartado.

Endereços de fonte e de destino. Os vários formatos do endereço de 128 bits do IPv6 são descritos no RFC 4291.

Dados. Esta é a parte da carga útil do datagrama IPv6. Quando o datagrama alcança seu destino, a carga útil pode ser extraída do datagrama IP e passada adiante para o protocolo especificado no campo próximo cabeçalho.

Nessa discussão, apresentamos a finalidade dos campos que estão incluídos no datagrama IPv6. Quando comparamos o formato do datagrama IPv6 da Figura 4.24 com o formato do datagrama IPv4 que vimos na Figura 4.13, notamos que diversos campos que aparecem no datagrama IPv4 não estão presentes no datagrama IPv6:

Fragmentação/remontagem. O IPv6 não permite fragmentação e remontagem em roteadores intermediários; essas operações podem ser realizadas somente pela fonte e pelo destino. Se um datagrama IPv6 recebido por um roteador for muito grande para ser repassado pelo enlace de saída, o roteador simplesmente descartará o datagrama e devolverá ao remetente uma mensagem ICMP de erro “Pacote muito grande” (veja a seguir). O remetente pode então reenviar os dados usando um datagrama IP de tamanho menor. Fragmentação e remontagem são operações que tomam muito tempo; retirar essas funcionalidades dos roteadores e colocá-las nos sistemas finais acelera consideravelmente o repasse IP para dentro da rede.

Soma de verificação do cabeçalho. Como os protocolos de camada de transporte (por exemplo, TCP e UDP) e de enlace de dados (por exemplo, Ethernet) nas camadas da Internet realizam soma de verificação, os projetistas do IP provavelmente acharam que essa funcionalidade era tão redundante na camada de rede que podia ser retirada. Mais uma vez, o processamento rápido de pacotes IP era uma preocupação principal. Lembre-se de que em nossa discussão sobre o IPv4 na Seção 4.4.1 vimos que, como o cabeçalho IPv4 contém um campo TTL (semelhante ao campo de limite de saltos no IPv6), a soma de verificação do cabeçalho IPv4 precisava ser recalculada em cada roteador. Como acontece com a fragmentação e a remontagem, esta também era uma operação de alto custo no IPv4.

Opções. O campo de opções não faz mais parte do cabeçalho-padrão do IP. Contudo, ele ainda não saiu de cena. Em vez disso, passou a ser um dos possíveis próximos cabeçalhos que poderão ser apontados pelo cabeçalho do IPv6. Em outras palavras, o campo de opções também poderá ser o próximo cabeçalho dentro de um pacote IP, exatamente como os cabeçalhos dos protocolos TCP e UDP. A remoção do campo de opções resulta em um cabeçalho IP de tamanho fixo de 40 bytes.

Lembre-se de que dissemos na Seção 4.4.3 que o protocolo ICMP é usado pelos nós IP para informar condições de erro e fornecer informações limitadas (por exemplo, a resposta de eco para uma mensagem ping) a um sistema final. Uma nova versão do ICMP foi definida para o IPv6 no RFC 4443. Além da reorganização das definições existentes de tipos e códigos ICMP, o ICMPv6 adicionou novos tipos e códigos requeridos pela nova funcionalidade do IPv6. Entre eles estão incluídos o tipo “Pacote muito grande” e um código de erro “opções IPv6 não reconhecidas”. Além disso, o ICMPv6 incorpora a funcionalidade do IGMP (Internet Group Management Protocol — protocolo de gerenciamento de grupos da Internet), que estudaremos na Seção 4.7. O IGMP, que é usado para gerenciar a adesão ou a saída de um hospedeiro de grupos *multicast*, era anteriormente um protocolo separado do ICMP no IPv4.

Transição do IPv4 para o IPv6

Agora que vimos os detalhes técnicos do IPv6, vamos tratar de um assunto muito prático: como a Internet pública, que é baseada no IPv4, fará a transição para o IPv6? O problema é que, enquanto os novos sistemas habilitados para IPv6 podem ser inversamente compatíveis, isto é, podem enviar, rotear e receber datagramas IPv4, os sistemas habilitados para IPv4 não podem manusear datagramas IPv6. Há várias opções possíveis.

Uma opção seria determinar um “dia da conversão” — uma data e um horário determinados em que todas as máquinas da Internet seriam desligadas e atualizadas, passando do IPv4 para o IPv6. A última transição importante de tecnologia (do uso do NPC para o uso do TCP para serviço confiável de transporte) ocorreu há quase 25 anos. E, mesmo naquela época [RFC 801], quando a Internet era pequenina e ainda era gerenciada por um número reduzido de ‘sabichões’, ficou claro que esse “dia da conversão” não era possível. Um dia como esse, envolvendo centenas de milhares de máquinas e milhões de gerenciadores de rede, é ainda mais impensável hoje. O RFC 4213 descreve duas abordagens (que podem ser usadas independentemente ou em conjunto) para a integração gradual dos hospedeiros e roteadores IPv4 ao mundo IPv6 (com a meta de longo prazo de fazer a transição de todos os nós IPv4 para IPv6).

Provavelmente, a maneira mais direta de introduzir nós habilitados ao IPv6 seja uma abordagem de **pilha dupla**, em que nós IPv6 também tenham uma implementação IPv4 completa. Esse nó, denominado nó IPv6/IPv4 no RFC 4213, estaria habilitado a enviar e receber tanto datagramas IPv4 quanto datagramas IPv6. Ao interagir com um nó IPv4, um nó IPv6/IPv4 poderá usar datagramas IPv4; ao interagir com um nó IPv6, poderá utilizar

IPv6. Nós IPv6/IPv4 devem ter endereços IPv6 e IPv4. Além disso, devem poder determinar se outro nó é habilitado para IPv6 ou somente para IPv4. Esse problema pode ser resolvido usando o DNS (veja o Capítulo 2), que poderá retornar um endereço IPv6 se o nome do nó a ser resolvido for capacitado para IPv6. Caso contrário, ele retornará um endereço IPv4. É claro que, se o nó que estiver emitindo a requisição DNS for habilitado apenas para IPv4, o DNS retornará apenas um endereço IPv4.

Na abordagem de pilha dupla, se o remetente ou o destinatário forem habilitados apenas para IPv4, um datagrama IPv4 deverá ser usado. Como resultado, é possível que dois nós habilitados para IPv6 acabem, em essência, enviando datagramas IPv4 um para o outro. Isso é ilustrado na Figura 4.25. Suponha que o nó A utiliza IPv6 e queira enviar um datagrama IP ao nó F, que também utiliza IPv6. Os nós A e B podem trocar um datagrama IPv6. Contudo, o nó B deve criar um datagrama IPv4 para enviar a C. É claro que o campo de dados do datagrama IPv6 pode ser copiado para o campo de dados do datagrama IPv4 e que o mapeamento adequado de endereço também pode ser feito. No entanto, ao realizar a conversão de IPv6 para IPv4, haverá campos IPv6 específicos no datagrama IPv6 (por exemplo, o campo do identificador de fluxo) que não terão correspondentes em IPv4. As informações contidas nesses campos serão perdidas. Assim, mesmo que E e F possam trocar datagramas IPv6, os datagramas IPv4 que chegarem a E e D não conterão todos os campos que estavam no datagrama IPv6 original enviado de A.

Uma alternativa para a abordagem de pilha dupla, também discutida no RFC 4213, é conhecida como **implantação de túnel**. O túnel pode resolver o problema observado anteriormente, permitindo, por exemplo, que E receba o datagrama IPv6 originado por A. A ideia básica por trás da implementação do túnel é a seguinte. Suponha que dois nós IPv6 (por exemplo, B e E na Figura 4.25) queiram interagir usando datagramas IPv6, mas estão conectados um ao outro por roteadores intervenientes IPv4. Referimo-nos ao conjunto de roteadores intervenientes IPv4 entre dois roteadores IPv6 como um **túnel**, como ilustrado na Figura 4.26.

Com a implementação do túnel, o nó IPv6 no lado remetente do túnel (por exemplo, B) pega o datagrama IPv6 inteiro e o coloca no campo de dados (carga útil) de um datagrama IPv4. Esse datagrama IPv4 é então endereçado ao nó IPv6 no lado receptor do túnel (por exemplo, E) e enviado ao primeiro nó do túnel (por exemplo, C). Os roteadores IPv4 intervenientes no túnel roteiam esse datagrama IPv4 entre eles, exatamente como fariam com qualquer outro datagrama, alheios ao fato de que o datagrama IPv4 contém um datagrama IPv6 completo. O nó IPv6 do lado receptor do túnel eventualmente recebe o datagrama IPv4 (pois ele é o destino do datagrama IPv4!), determina que o datagrama IPv4 contém um datagrama IPv6, extrai o datagrama IPv6 e, então, roteia o datagrama IPv6 exatamente como o faria se tivesse recebido o datagrama IPv6 de um vizinho IPv6 diretamente ligado a ele.

Encerramos esta seção mencionando que enquanto a adoção do IPv6 estava demorando para decolar [Lawton, 2001], a iniciativa foi tomada recentemente. Consulte [Huston, 2008b] para o emprego do IPv6 a partir de 2008. O Departamento de Gerenciamento do Orçamento da Casa Branca (OMB) ordenou que os roteadores centrais das redes do governo americano fossem habilitados para IPv6 em meados de 2008; um número de órgãos cumpriu tal ordem até a composição deste livro (novembro de 2008). A proliferação de dispositivos como telefones IP e outros dispositivos portáteis possam dar o empurrão que falta para a aplicação mais ampla do IPv6.

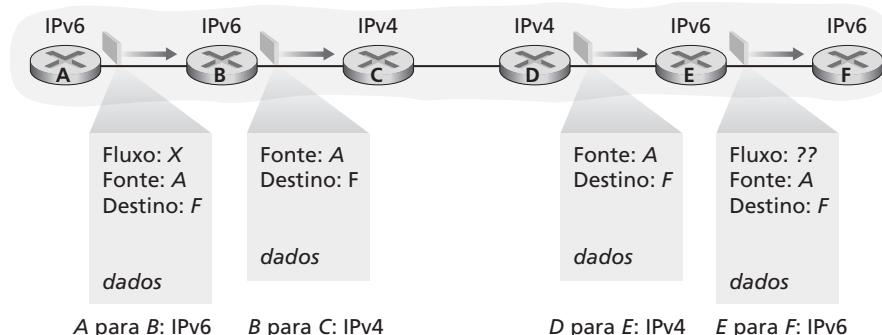


Figura 4.25 Abordagem de pilha dupla

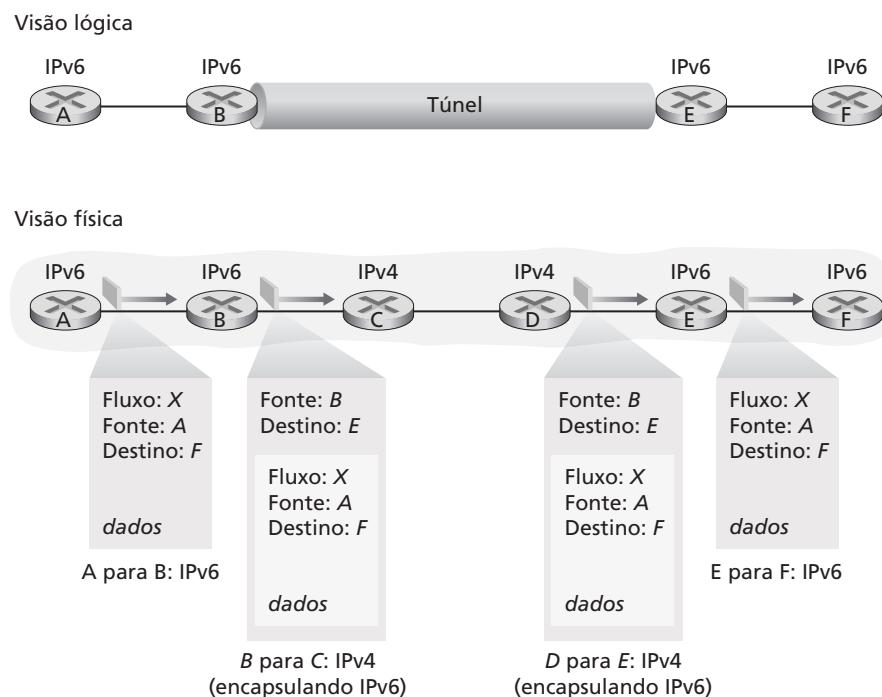


Figura 4.26 Implementação de túnel

O programa europeu Third Generation Partnership [3GPP, 2009] especificou o IPv6 como o padrão de endereçamento para multimídia móvel. Mesmo que o IPv6 não tenha sido disponibilizado amplamente nos primeiros dez anos de sua jovem vida, está bem claro que é preciso uma visão de longo prazo. O sistema de números de telefones existente hoje demorou várias décadas para se firmar, mas está vigente há quase meio século e não dá nenhum sinal de que pretende sair de cena. De modo semelhante, pode levar algum tempo para o IPv6 se firmar, mas, depois, provavelmente ele também será utilizado durante muito tempo. Brian Carpenter, antigo diretor do Internet Architecture Board [IAB, 2009] e autor de diversos RFCs relacionados ao IPv6, declarou: “Eu sempre considerei a adoção do IPv6 como um processo de 15 anos, começando em 1995.” [Lawton, 2001]. Pelas contas de Carpenter, já estamos perto dos dois terços desse tempo!

Uma lição importante que podemos aprender com a experiência do IPv6 é que há enorme dificuldade para mudar protocolos de camada de rede. Desde o início da década de 1990, numerosos novos protocolos de camada de rede foram anunciados como a próxima maior revolução da Internet, mas a maioria desses protocolos teve pouca aceitação até agora. Dentre eles, estão o IPv6, os protocolos *multicast* (veja a Seção 4.7) e os protocolos de reserva de recursos (Capítulo 7). Na verdade, introduzir novos protocolos na camada de rede é como substituir as fundações de uma casa — é difícil de fazer sem demolir a casa inteira ou, no mínimo, retirar os moradores temporariamente da residência. Por outro lado, a Internet vem testemunhando a rápida disponibilização de novos protocolos na camada de aplicação. Os exemplos clássicos são, é claro, a Web, a mensagem instantânea e o compartilhamento de arquivos P2P. Outros exemplos são a recepção de áudio e vídeo em tempo real e os jogos distribuídos. Introduzir novos protocolos de camada de aplicação é como pintar uma casa — é relativamente fácil de fazer e, se você escolher uma cor atraente, outras casas da vizinhança vão imitá-lo. Em resumo, no futuro podemos esperar mudanças na camada de rede da Internet, mas essas mudanças provavelmente ocorrerão dentro de uma escala de tempo bem mais lenta do que as que ocorrerão na camada de aplicação.

4.4.5 Uma breve investida em segurança IP

Na Seção 4.4.3 discutimos o IPv4 em alguns detalhes, incluindo os serviços que ele provê e como esses serviços são implementados. Ao ler esta seção, você pode ter percebido que nenhum serviço de segurança foi mencionado. Realmente, o IPv4 foi projetado em uma era (anos 1970) em que a Internet era utilizada, primordialmente,

entre pesquisadores de rede mutualmente confiáveis. Criar uma rede de computadores que integrava uma grande quantidade de tecnologias da camada de enlace já era desafiador, sem ter de se preocupar com a segurança.

Mas, com a segurança sendo a principal preocupação hoje, pesquisadores da Internet começaram a criar novos protocolos da camada de rede que oferecem uma variedade de serviços de segurança. Um desses protocolos é o IPsec, um dos protocolos da camada de rede mais conhecidos e empregados em Redes Virtuais Privadas (VPNs). Embora o IPsec e seus suportes sejam abordados no Capítulo 8, apresentamos uma breve introdução sobre os serviços do IPsec nesta seção.

O IPsec foi desenvolvido para ser compatível com o IPv4 e o IPv6. Particularmente, para obter os benefícios do IPv6, não precisamos substituir as pilhas dos protocolos em todos os roteadores e hospedeiros na Internet. Por exemplo, utilizando o modo transporte (um dos “modos” do IPsec), se dois hospedeiros querem se comunicar de modo seguro, o IPsec precisa estar disponível somente em dois hospedeiros. Todos os outros roteadores e hospedeiros podem continuar a rodar o IPv4 simples.

Para uma abordagem concreta, a partir daqui focaremos no modo transporte do IPsec. Neste modo, dois hospedeiros estabelecem, primeiramente, uma sessão entre si mesmos. (Assim, o IPsec é orientado a conexão!) Com a sessão em vigor, todos os segmentos TCP e UDP enviados entre os dois hospedeiros aproveitam os serviços de segurança providos pelo IPsec. No lado remetente, a camada de transporte passa um segmento para o IPsec. Este, então, codifica o segmento, acrescenta campos de segurança adicionais a ele e envolve a carga útil resultante em um datagrama IP comum. (Na verdade, isso é mais complicado, como veremos no Capítulo 8.) O hospedeiro remetente, envia, então o datagrama para a Internet, a qual o transporta para o hospedeiro destinatário. Em seguida, o IPsec decodifica o segmento e encaminha o segmento decodificado à camada de transporte.

Os serviços providos por uma sessão IPsec incluem:

- **Acordo criptográfico.** Mecanismos que permitem que dois hospedeiros de comunicação concordem nos algoritmos criptográficos e chaves.
- **Codificação das cargas úteis do datagrama IP.** Quando o hospedeiro destinatário recebe um segmento da camada de transporte, o IPsec codifica a carga útil, que pode ser somente decodificada pelo IPsec no hospedeiro destinatário.
- **Inegridade dos dados.** O IPsec permite que o hospedeiro destinatário verifique se os campos do cabeçalho do datagrama e a carga útil codificada não foram modificados enquanto o datagrama estava a caminho da fonte ao destino.
- **Autenticação de origem.** Quando um hospedeiro recebe um datagrama IPsec de uma fonte confiável (com uma chave confiável — consulte Capítulo 8), o hospedeiro está certo de que o endereço IP remetente no datagrama é a verdadeira fonte do datagrama.

Quando dois hospedeiros estabelecem uma sessão IPsec, todos os segmentos TCP e UDP enviados entre eles serão codificados e autenticados. O IPsec, portanto, oferece uma cobertura geral, protegendo toda a comunicação entre os dois hospedeiros para todas as aplicações de rede.

Uma empresa pode utilizar o IPsec para se comunicar de forma segura na Internet pública não segura. Para fins de ilustração, verificaremos um exemplo simples. Considere uma empresa que possua um grande número de vendedores que viajam, cada um possuindo um laptop da empresa. Suponha que os vendedores precisem consultar, frequentemente, informações confidenciais sobre a empresa (por exemplo, informações sobre preços e produtos) armazenadas em um servidor na matriz da empresa. Suponha, ainda, que os vendedores também precisem enviar documentos confidenciais um para o outro. Como isso pode ser feito com o IPsec? Como você pode imaginar, instalamos o IPsec no servidor e em todos os laptops dos vendedores. Com ele instalado nesses hospedeiros, quando um vendedor precisar se comunicar com o servidor ou com outro vendedor, a sessão de comunicação estará protegida.

4.5 Algoritmos de roteamento

Exploramos, neste capítulo, a função de repasse da camada de rede mais do que qualquer outra. Aprendemos que, quando um pacote chega a um roteador, o roteador indexa uma tabela de repasse e determina a interface de

enlace para a qual o pacote deve ser dirigido. Aprendemos também que algoritmos de roteamento que rodam em roteadores de rede trocam e calculam as informações que são utilizadas para configurar essas tabelas de repasse. A interação entre algoritmos de roteamento e tabelas de repasse foi ilustrada na Figura 4.2. Agora que já nos aprofundamos um pouco na questão do repasse, voltaremos nossa atenção para o outro tópico importante desse capítulo, a saber, a função crítica da camada de rede, o roteamento. Quer ofereça um serviço de datagramas (quando pacotes diferentes entre um determinado par fonte-destino podem pegar rotas diferentes) ou um serviço de VCs (quando todos os pacotes entre uma fonte e um destino determinados pegarão o mesmo caminho), a camada de rede deve, mesmo assim, determinar o caminho que os pacotes percorrem entre remetentes e destinatários. Veremos que a tarefa do roteamento é determinar bons caminhos (ou rotas) entre remetentes e destinatários através da rede de roteadores.

Normalmente um hospedeiro está ligado diretamente a um roteador, o **roteador default** para esse hospedeiro (também denominado **roteador do primeiro salto** para o hospedeiro). Sempre que um hospedeiro emitir um pacote, o pacote será transferido para seu roteador default. Denominamos **roteador da fonte** o roteador default do hospedeiro de origem e **roteador de destino** o roteador default do hospedeiro de destino. O problema de rotear um pacote do hospedeiro de origem até o hospedeiro destinatário se reduz, claramente, ao problema de rotear o pacote do roteador da fonte ao roteador de destino, que é o foco desta seção.

Portanto, a finalidade de um algoritmo de roteamento é simples: dado um conjunto de roteadores conectados por enlaces, um algoritmo de roteamento descobre um ‘bom’ caminho entre o roteador de fonte e o roteador de destino. Normalmente, um ‘bom’ caminho é aquele que tem o ‘menor custo’. No entanto, veremos que, na prática, preocupações do mundo real, como questões de política (por exemplo, uma regra que determina que “o roteador X, de propriedade da organização Y, não deverá repassar nenhum pacote originário da rede de propriedade da organização Z”), também entram em jogo para complicar algoritmos conceitualmente simples e elegantes, cuja teoria fundamenta a prática de roteamento nas redes de hoje.

Um grafo é usado para formular problemas de roteamento. Lembre-se de que um **grafo** $G = (N, E)$ é um conjunto N de nós e uma coleção E de arestas, no qual cada aresta é um par de nós do conjunto N . No contexto do roteamento da camada de rede, os nós do grafo representam roteadores — os pontos nos quais são tomadas decisões de repasse de pacotes — e as arestas que conectam esses nós representam os enlaces físicos entre esses roteadores. Uma abstração gráfica de uma rede de computadores está ilustrada na Figura 4.27.

Para ver alguns grafos que representam mapas de redes reais, consulte [Dodge, 2007, Cheswick, 2000]. Para uma discussão sobre quanto bem os diferentes modelos fundamentados em grafos modelam a Internet, consulte [Zegura, 1997, Faloutsos, 1999, Li, 2004]. Como ilustrado na Figura 4.27, uma aresta também tem um valor que representa seu custo. Normalmente, o custo de uma aresta pode refletir o tamanho físico do enlace correspondente (por exemplo, um enlace transoceânico poderia ter um custo mais alto do que um enlace terrestre de curta distância), a velocidade do enlace ou o custo monetário a ele associado. Para nossos objetivos, consideraremos os custos das arestas apenas como um dado e não nos preocuparemos com o modo como eles são determinados. Para qualquer aresta (x, y) em E , denominamos $c(x, y)$ o custo da aresta entre os nós x e y . Se o par (x, y) não pertencer a E , estabelecemos $c(x, y) = \infty$. Além disso, sempre consideraremos somente grafos não direcionados

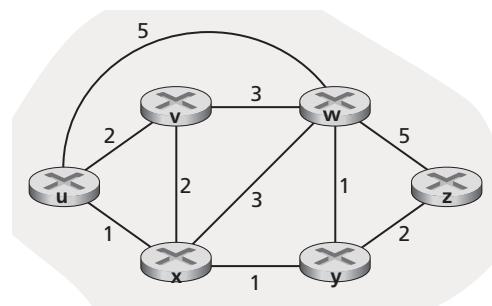


Figura 4.27 Modelo abstrato de grafo de uma rede de computadores

(isto é, grafos cujas arestas não têm uma direção), de modo que a aresta (x,y) é a mesma que a aresta (y,x) e $c(x,y) = c(y,x)$. Dizemos também que y é um **vizinho** do nó x se (x,y) pertencer a E .

Dado que são atribuídos custos às várias arestas na abstração do grafo, uma meta natural de um algoritmo de roteamento é identificar o caminho de menor custo entre fontes e destinos. Para tornar esse problema mais preciso, lembremos que um **caminho** em um grafo $G = (N,E)$ é uma sequência de nós (x_1, x_2, \dots, x_p) tal que cada um dos pares $(x_1, x_2), (x_2, x_3), \dots, (x_{p-1}, x_p)$ são arestas em E . O custo de um caminho (x_1, x_2, \dots, x_p) é simplesmente a soma de todos os custos das arestas ao longo do caminho, ou seja, $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$. Dados quaisquer dois nós x e y , normalmente há muitos caminhos entre os dois, e cada caminho tem um custo. Um ou mais desses caminhos é um **caminho de menor custo**. Por conseguinte, o problema do menor custo é claro: descobrir um caminho entre a fonte e o destino que tenha o menor custo. Na Figura 4.27, por exemplo, o caminho de menor custo entre o nó da fonte u e o nó de destino w é (u, x, y, w) , cujo custo de caminho é 3. Note que, se todas as arestas do grafo tiverem o mesmo custo, o caminho de menor custo também é o **caminho mais curto** (isto é, o caminho que tem o menor número de enlaces entre a fonte e o destino).

Apenas como simples exercício, tente descobrir o caminho de menor custo entre os nós u e z na Figura 4.27 e reflita um pouco sobre como você calculou esse caminho. Se você for como a maioria das pessoas, descobriu o caminho de u a z examinando a figura, traçando algumas rotas de u a z , e se convencendo, de algum modo, que o caminho que você escolheu tinha o menor custo entre todos os caminhos possíveis. (Você verificou todos os 17 possíveis caminhos entre u e z ? Provavelmente, não!) Esse cálculo é um exemplo de um algoritmo de roteamento centralizado — o algoritmo é rodado em um local, o seu cérebro, com informações completas sobre a rede. De modo geral, uma maneira possível de classificar algoritmos de roteamento é como globais ou centralizados.

Um **algoritmo de roteamento global** calcula o caminho de menor custo entre uma fonte e um destino usando conhecimento completo e global sobre a rede. Em outras palavras, o algoritmo considera como dados de cálculo a conectividade entre todos os nós e todos os custos dos enlaces. E isso exige que o algoritmo obtenha essas informações, de algum modo, antes de realmente realizar o cálculo. O cálculo em si pode ser rodado em um local (um algoritmo de roteamento global centralizado) ou duplicado em vários locais. Contudo, a principal característica distintiva, nesse caso, é que um algoritmo global tem informação completa sobre conectividade e custo de enlaces. Na prática, algoritmos com informação global de estado são frequentemente denominados **algoritmos de estado de enlace** (*link-state* — LS), já que devem estar a par dos custos de cada enlace na rede. Estudaremos algoritmos de estado de enlace na Seção 4.5.1

Em um **algoritmo de roteamento descentralizado**, o cálculo do caminho de menor custo é realizado de modo iterativo e distribuído. Nenhum nó tem informação completa sobre os custos de todos os enlaces da rede. Em vez disso, cada nó começa sabendo apenas os custos dos enlaces diretamente ligados a ele. Então, por meio de um processo iterativo de cálculo e de troca de informações com seus nós vizinhos (isto é, nós que estão na outra extremidade dos enlaces aos quais ele próprio está ligado), um nó gradualmente calcula o caminho de menor custo até um destino ou um conjunto de destinos. O algoritmo de roteamento descentralizado que estudaremos logo adiante na Seção 4.5.2 é denominado **algoritmo de vetor de distâncias** (*distance-vector algorithm* — DV) porque cada nó mantém um vetor de estimativas de custos (distâncias) de um nó até todos os outros nós da rede.

Uma segunda maneira geral de classificar algoritmos de roteamento é como estáticos ou dinâmicos. Em **algoritmos de roteamento estáticos**, as rotas mudam muito lentamente ao longo do tempo, muitas vezes como resultado de intervenção humana (por exemplo, uma pessoa editando manualmente a tabela de repasse do roteador). **Algoritmos de roteamento dinâmicos** mudam os caminhos de roteamento à medida que mudam as cargas de tráfego ou a topologia da rede. Um algoritmo dinâmico pode ser rodado periodicamente ou como reação direta à mudanças de topologia ou de custo dos enlaces. Ao mesmo tempo que são mais sensíveis à mudanças na rede, algoritmos dinâmicos também são mais suscetíveis a problemas como loops de roteamento e oscilação em rotas.

Uma terceira maneira de classificar algoritmos de roteamento é como sensíveis à carga ou insensíveis à carga. Em um **algoritmo sensível à carga**, custos de enlace variam dinamicamente para refletir o nível corrente de congestionamento no enlace subjacente. Se houver um alto custo associado com um enlace que está correntemente

congestionado, um algoritmo de roteamento tenderá a escolher rotas que evitem esse enlace congestionado. Embora antigos algoritmos da ARPAnet fossem sensíveis à carga [McQuillan, 1980], foram encontradas várias dificuldades [Huitema, 1998]. Os algoritmos de roteamento utilizados na Internet hoje (como RIP, OSPF e BGP) são **insensíveis à carga**, pois o custo de um enlace não reflete explicitamente seu nível de congestionamento corrente (nem o de congestionamento mais recente).

4.5.1 O algoritmo de roteamento de estado de enlace (LS)

Lembre-se de que em um algoritmo de estado de enlace a topologia da rede e todos os custos de enlace são conhecidos, isto é, estão disponíveis como dados para o algoritmo de estado de enlace. Na prática, isso se consegue fazendo com que cada nó transmita pacotes de estado de enlace a todos os outros nós da rede, sendo que cada um desses pacotes contém as identidades e os custos dos enlaces ligados a ele. Na prática (por exemplo, com o protocolo de roteamento OSPF da Internet, discutido na Seção 4.6.1) isso frequentemente é conseguido com um algoritmo de **transmissão broadcast de estado de enlace** [Perlman, 1999]. Algoritmos de transmissão broadcast serão estudados na Seção 4.7. O resultado da transmissão broadcast dos nós é que todos os nós têm uma visão idêntica e completa da rede. Cada nó pode, então, rodar o algoritmo de estado de enlace e calcular o mesmo conjunto de caminhos de menor custo como todos os outros nós.

O algoritmo de roteamento de estado de enlace que apresentamos adiante é conhecido como *algoritmo de Dijkstra*, o nome de seu inventor. Um algoritmo que guarda relações muito próximas com ele é o algoritmo de Prim. Consulte [Cormen, 2001] para uma discussão geral sobre algoritmos de grafo. O algoritmo de Dijkstra calcula o caminho de menor custo entre um nó (a fonte, que chamaremos de u) e todos os outros nós da rede. É um algoritmo iterativo e tem a propriedade de, após a k -ésima iteração, conhecer os caminhos de menor custo para k nós de destino e, dentre os caminhos de menor custo até todos os nós de destino, esses k caminhos terão os k menores custos. Vamos definir a seguinte notação:

$D(v)$: custo do caminho de menor custo entre o nó da fonte e o destino v até essa iteração do algoritmo.

$p(v)$: nó anterior (vizinho de v) ao longo do caminho de menor custo corrente desde a fonte até v .

N' : subconjunto de nós; v pertence a N' se o caminho de menor custo entre a fonte e v for inequivocavelmente conhecido.

O algoritmo de roteamento global consiste em uma etapa de inicialização seguida de um loop. O número de vezes que o loop é rodado é igual ao número de nós na rede. Ao terminar, o algoritmo terá calculado os caminhos mais curtos desde o nó da fonte u até todos os outros nós da rede.

Algoritmo de estado de enlace para o nó da fonte u

```

1 Inicialização:
2      $N' = \{u\}$ 
3     para todos os nós  $v$ 
4         se  $v$  for um vizinho de  $u$ 
5             então  $D(v) = c(u, v)$ 
6         senão  $D(v) = \infty$ 
7
8 Loop
9     encontre  $w$  não em  $N'$ , tal que  $D(w)$  é um mínimo
10    adicione  $w$  a  $N'$ 
11    atualize  $D(v)$  para cada vizinho  $v$  de  $w$  e não em  $N'$ :
12         $D(v) = \min(D(v), D(w) + c(w, v))$ 
13    /* o novo custo para  $v$  é o velho custo para  $v$ 
14    ou o custo do menor caminho conhecido para  $w$  mais o custo de  $w$  para  $v$  */
15 até  $N' = N$ 

```

Como exemplo, vamos considerar a rede da Figura 4.27 e calcular os caminhos de menor custo de u até todos os destinos possíveis. Os cálculos do algoritmo estão ilustrados na Tabela 4.3, na qual cada linha fornece os valores das variáveis do algoritmo ao final da iteração. Vamos considerar detalhadamente alguns dos primeiros estágios:

No estágio de inicialização, os caminhos de menor custo correntemente conhecidos de u até os vizinhos diretamente ligados a ele (v , w e x) são inicializados para 2, 1 e 5, respectivamente. Note, em particular, que o custo até w é estabelecido em 5 (embora logo veremos que, na realidade, existe um trajeto cujo custo é ainda menor), já que este é o custo do enlace (um salto) direto u a w . Os custos até y e z são estabelecidos como infinito, porque eles não estão diretamente conectados a u .

Na primeira iteração, examinamos os nós que ainda não foram adicionados ao conjunto N' e descobrimos o nó de menor custo ao final da iteração anterior. Este é o nó x , com um custo de 1, e, assim, x é adicionado ao conjunto N' . A linha 12 do algoritmo de vetor de distâncias (LS) é então rodada para atualizar $D(v)$ para todos os nós v , produzindo os resultados mostrados na segunda linha (Etapa 1) da Tabela 4.3. O custo do caminho até v não muda. Descobriremos que o custo do caminho até w através do nó x (que era 5 ao final da inicialização), é 4. Por conseguinte, esse caminho de custo mais baixo é selecionado e o predecessor de w ao longo do caminho mais curto a partir de u é definido como x . De maneira semelhante, o custo até y (através de x) é computado como 2 e a tabela é atualizada de acordo com isso.

Na segunda iteração, verificamos que os nós v e y são os que têm os caminhos de menor custo (2); decidimos o empate arbitrariamente e adicionamos y ao conjunto N' de modo que N' agora contém u , x e y . O custo dos nós remanescentes que ainda não estão em N' (isto é, nós v , w e z) são atualizados pela linha 12 do algoritmo LS, produzindo os resultados mostrados na terceira linha da Tabela 4.3.

E assim por diante...

Quando o algoritmo LS termina, temos, para cada nó, seu predecessor ao longo do caminho de menor custo a partir do nó da fonte. Temos também o predecessor para cada um desses predecessores; desse modo, podemos construir o caminho inteiro desde a fonte até todos os destinos. Então, a tabela de repasse em um nó, por exemplo, u , pode ser construída a partir dessas informações, armazenando, para cada destino, o nó do salto seguinte no caminho de menor custo de u até o destino. A Figura 4.28 mostra os caminhos de menor custo resultantes e a tabela de repasse em u para a rede na Figura 4.27

Qual é a complexidade do cálculo desse algoritmo? Isto é, dados n nós (sem contar a fonte), quanto cálculo é preciso efetuar no pior caso para descobrir os caminhos de menor custo entre a fonte e todos os destinos? Na primeira iteração, precisamos pesquisar todos os n nós para determinar o nó w , que não está em N' , e que tem o custo mínimo. Na segunda iteração, precisamos verificar $n - 1$ nós para determinar o custo mínimo. Na terceira iteração, $n - 2$ nós. E assim por diante. Em termos gerais, o número total de nós que precisamos pesquisar em todas as iterações é $n(n + 1)/2$, e, assim, dizemos que a complexidade da implementação desse algoritmo de estado de enlace para o pior caso é de ordem n ao quadrado: $O(n^2)$. (Uma implementação mais sofisticada desse

Etapa	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	∞	∞
1	ux	2, u	4, x		2, x	∞
2	uxy	2, u	3, y			4, y
3	$uxyv$		3, y			4, y
4	$uxyvw$					4, y
5	$uxyvwz$					

Tabela 4.3 Execução do algoritmo de estado de enlace na rede da Figura 4.25

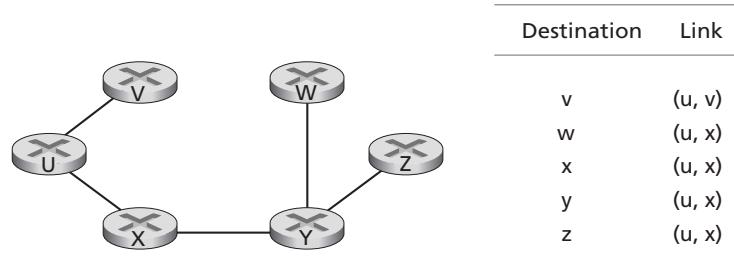


Figura 4.28 Caminhos de menor custo resultantes e a tabela de repasse para o nó u

algoritmo, que utiliza uma estrutura de dados conhecida como uma pilha, pode descobrir o mínimo na linha 9 em tempo logarítmico e não-linear, reduzindo assim a complexidade.)

Antes de concluirmos nossa discussão sobre o algoritmo LS, vamos considerar uma patologia que pode surgir. A Figura 4.29 mostra uma topologia de rede simples em que os custos dos enlaces são iguais à carga transportada pelo enlace, refletindo, por exemplo, o atraso que seria experimentado. Nesse exemplo, os custos dos enlaces não são simétricos, isto é, $c(u,v)$ é igual a $c(v,u)$ somente se a carga transportada em ambas as direções do enlace (u,v) for a mesma. Nesse exemplo, o nó z origina uma unidade de tráfego destinada a w, o nó x também origina uma unidade de tráfego destinada a w e o nó y injeta uma quantidade de tráfego igual a e também destinada a w. O roteamento inicial é mostrado na Figura 4.29(a) com os custos dos enlaces correspondentes à quantidade de tráfego carregada.

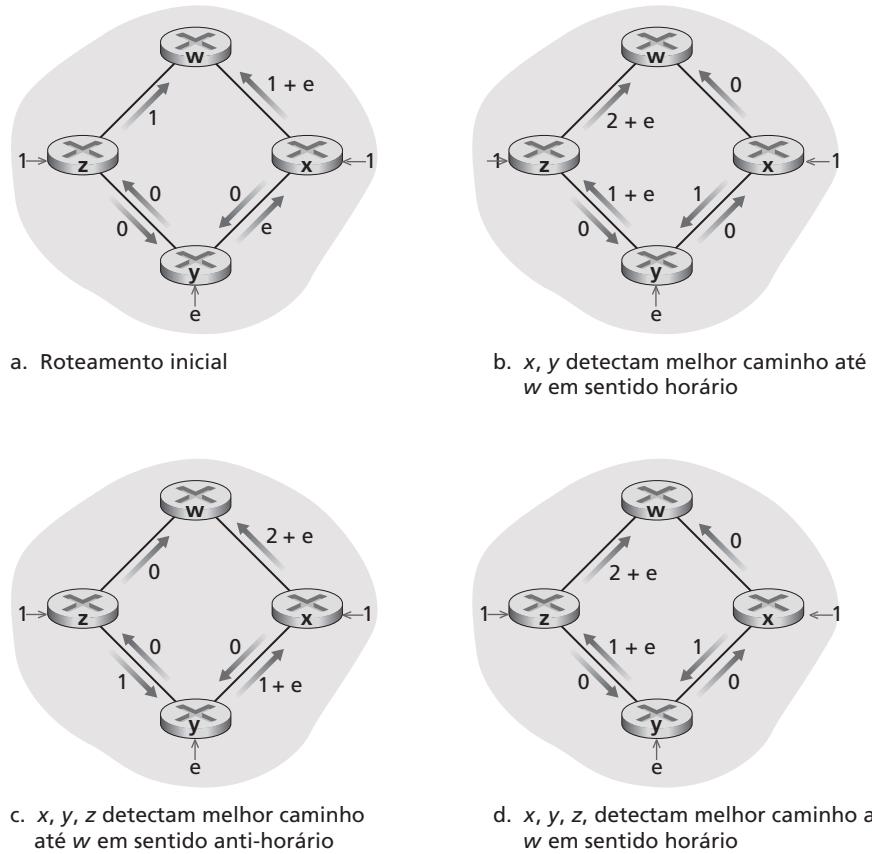


Figura 4.29 Oscilações com roteamento de congestionamento

Quando o algoritmo LS é rodado novamente, o nó y determina — baseado nos custos dos enlaces mostrados na Figura 4.29(a) — que o caminho em sentido horário até w tem um custo de 1, ao passo que o caminho em sentido anti-horário até w (que estava sendo usado) tem um custo de $1 + e$. Por conseguinte, o caminho de menor custo de y até w é agora em sentido horário. De maneira semelhante, x determina que seu novo caminho de menor custo até w é também em sentido horário, resultando nos custos mostrados na Figura 4.29(b). Na próxima vez em que o algoritmo LS é rodado, os nós x , y , e z detectam um caminho de custo zero até w na direção anti-horária, e todos dirigem seu tráfego para as rotas anti-horárias. Na próxima vez em que o algoritmo LS é rodado, os nós x , y , e z então dirigem seu tráfego para as rotas em sentido horário.

O que pode ser feito para evitar essas oscilações (que podem ocorrer com qualquer algoritmo, e não apenas com um algoritmo LS, que use uma métrica de enlace baseada em congestionamento ou em atraso)? Uma solução seria tornar obrigatório que os custos dos enlaces não dependessem da quantidade de tráfego carregada — uma solução inaceitável, já que um dos objetivos do roteamento é evitar enlaces muito congestionados (por exemplo, enlaces com grande atraso). Outra solução seria assegurar que nem todos os roteadores rodassem o algoritmo LS ao mesmo tempo. Esta parece ser uma solução mais razoável, já que é de esperar que, mesmo que os roteadores rodem o algoritmo LS com idêntica periodicidade, o instante de execução do algoritmo não seja o mesmo em cada nó. O interessante é que recentemente os pesquisadores descobriram que os roteadores da Internet podem se auto-sincronizar [Floyd Synchronization, 1994], isto é, mesmo que inicialmente rodem o algoritmo com o mesmo período, mas em diferentes momentos, o instante de execução do algoritmo pode finalmente se tornar, e permanecer, sincronizado nos roteadores. Um modo de evitar essa autossincronização é cada roteador variar aleatoriamente o instante em que envia um anúncio de enlace.

Agora que examinamos o algoritmo de estado de enlace, vamos analisar outro importante algoritmo usado hoje na prática — o algoritmo de roteamento de vetor de distâncias.

4.5.2 O algoritmo de roteamento de vetor de distâncias (DV)

Enquanto o algoritmo LS usa informação global, o algoritmo de vetor de distâncias (**distance-vector — DV**) é iterativo, assíncrono e distribuído. É distribuído porque cada nó recebe alguma informação de um ou mais vizinhos diretamente ligados a ele, realiza cálculos e, em seguida, distribui os resultados de seus cálculos para seus vizinhos. É iterativo porque esse processo continua até que mais nenhuma informação seja trocada entre vizinhos. (O interessante é que este é um algoritmo finito — não há nenhum sinal de que o cálculo deve parar; ele apenas para.) O algoritmo é assíncrono porque não requer que todos os nós rodem simultaneamente. Veremos que um algoritmo assíncrono, iterativo, finito e distribuído é muito mais interessante e divertido do que um algoritmo centralizado!

Antes de apresentar o algoritmo DV, é bom discutir uma relação importante que existe entre os custos dos caminhos de menor custo. Seja $d_x(y)$ o custo do caminho de menor custo do nó x ao nó y . Então, os menores custos estão relacionados segundo a famosa equação de Bellman-Ford, a saber:

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}, \quad (4.1)$$

onde o \min_v da equação é calculado para todos os vizinhos de x . A equação de Bellman-Ford é bastante intuitiva. Realmente, se após transitarmos de x para v tomarmos o caminho de menor custo de v a y , o custo do caminho será $c(x,v) + d_v(y)$. Como devemos começar viajando até algum vizinho v , o caminho de menor custo de x a y é o mínimo do conjunto dos $c(x,v) + d_v(y)$ calculados para todos os vizinhos v .

Mas, para aqueles que ainda se mostrem céticos quanto à validade da equação, vamos verificá-la para o nó de fonte u e o nó de destino z na Figura 4.27. O nó da fonte u tem três vizinhos: nós v , x e w . Percorrendo vários caminhos no grafo, é fácil ver que $d_v(z) = 5$, $d_x(z) = 3$ e $d_w(z) = 3$. Passando esses valores para a Equação 4.1, juntamente com os custos $c(u,v) = 2$, $c(u,x) = 1$ e $c(u,w) = 5$, temos $d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$, que é, obviamente, verdade e que é, exatamente, o resultado conseguido com o algoritmo de Dijkstra para a mesma rede. Essa verificação rápida deve ajudá-lo a vencer qualquer ceticismo que ainda possa ter.



A equação de Bellman-Ford não é apenas uma curiosidade intelectual. Na verdade, ela tem uma importância prática significativa. Em particular, a solução da equação de Bellman-Ford fornece os registros da tabela de repasse do nó x . Para verificar isso, seja v^* qualquer nó vizinho que represente o mínimo na Equação 4.1. Então, se o nó x quiser enviar um pacote ao nó y pelo caminho de menor custo, deverá, em primeiro lugar, repassar o pacote para o nó v^* . Assim, a tabela de repasse do nó x especificaria o nó v^* como o roteador do próximo salto para o destino final y . Uma outra contribuição importante da equação de Bellman-Ford é que ela sugere a forma da comunicação vizinho para vizinho que ocorrerá no algoritmo DV.

A ideia básica é a seguinte. Cada nó começa com $D_x(y)$, uma estimativa do custo do caminho de menor custo entre ele mesmo e o nó y , para todos os nós em N . Seja $\mathbf{D}_x = [D_x(y): y \in N]$ o vetor de distâncias do nó x , que é o vetor de estimativas de custo de x até todos os outros nós, y , em N . Com o algoritmo DV cada nó x mantém os seguintes dados de roteamento:

- Para cada vizinho v , o custo $c(x,v)$ de x até o vizinho diretamente ligado a ele, v
- O vetor de distâncias do nó x , isto é, $\mathbf{D}_x = [D_x(y): y \in N]$, contendo a estimativa de x para seus custos até todos os destinos, y , em N
- Os vetores de distâncias de seus vizinhos, isto é, $\mathbf{D}_v = [D_v(y): y \in N]$ para cada vizinho v de x

No algoritmo distribuído, assíncrono, cada nó envia, a intervalos regulares, uma cópia do seu vetor de distâncias a cada um de seus vizinhos. Quando um nó x recebe um novo vetor de distâncias de qualquer de seus vizinhos v , ele armazena o vetor de distâncias de v e então usa a equação de Bellman-Ford para atualizar seu próprio vetor de distâncias, como segue:

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\} \text{ para cada nó } y \text{ em } N$$

Se o vetor de distâncias do nó x tiver mudado como resultado dessa etapa de atualização, o nó x então enviará seu vetor de distâncias atualizado para cada um de seus vizinhos que, por sua vez, podem atualizar seus próprios vetores de distâncias. Parece milagre, mas, contanto que todos os nós continuem a trocar seus vetores de distâncias assincronamente, cada estimativa de custo $D_x(y)$ convergirá para $d_x(y)$, que é, na verdade, o custo do caminho de menor custo do nó x ao nó y [Bertsekas, 1991]!

Algoritmo vetor de distância (DV)

Para cada nó, x :

```

1 Inicialização:
2   para todos os destinos  $y$  em  $N$ :
3      $D_x(y) = c(x,y)$  /* se  $y$  não é um vizinho então  $c(x,y) = \infty$  */
4   para cada vizinho  $w$ 
5      $D_w(y) = ?$  para todos os destinos  $y$  em  $N$ 
6   para cada vizinho  $w$ 
7     envia um vetor de distâncias  $\mathbf{D}_x = [D_x(y): y \in N]$  para  $w$ 
8
9 loop
10 wait (até que ocorra uma mudança no custo do enlace ao vizinho  $w$  ou
11       até a recepção de um vetor de distâncias do vizinho  $w$ )
12
13 para cada  $y$  em  $N$ :
14    $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ 
15
16 if  $D_x(y)$  mudou para algum destino  $y$ 
17   envia um vetor de distâncias  $\mathbf{D}_x = [D_x(y): y \in N]$  para todos os vizinhos
18
19 forever

```

No algoritmo DV um nó x atualiza sua estimativa do vetor de distâncias quando percebe um mudança de custo em um dos enlaces ligados diretamente a ele ou recebe uma atualização do vetor de distâncias de algum vizinho. Mas, para atualizar sua própria tabela de repasse para um dado destino y , o que o nó x realmente precisa saber não é a distância do caminho mais curto até y , mas qual nó vizinho $v^*(y)$ é o roteador do próximo salto ao longo do caminho mais curto até y . Como era de esperar, o roteador do próximo salto $v^*(y)$ é o vizinho v que representar o mínimo na Linha 14 do algoritmo DV. (Se houver vários vizinhos v que representem o mínimo, então $v^*(y)$ pode ser qualquer um dos vizinhos minimizadores.) Assim, nas Linhas 13-14, para cada destino y , o nó x também determina $v^*(y)$ e atualiza sua tabela de repasse para o destino y .

Lembre-se de que o algoritmo LS é um algoritmo global no sentido de que requer que cada nó obtenha, em primeiro lugar, um mapa completo da rede antes de rodar o algoritmo de Dijkstra. O algoritmo DV é descentralizado e não usa essa informação global. Realmente, a única informação que um nó terá são os custos dos enlaces até os vizinhos diretamente ligados a ele e as informações que recebe desses vizinhos. Cada nó espera uma atualização de qualquer vizinho (Linhas 10-11), calcula seu novo vetor de distâncias ao receber uma atualização (Linha 14) e distribui seu novo vetor de distâncias a seus vizinhos (Linhas 16-17). Algoritmos semelhantes ao DV são utilizados em muitos protocolos de roteamento na prática, entre eles o RIP e o BGP da Internet, o ISO IDRP, o IPX da Novell, e o ARPAnet original.

A Figura 4.30 ilustra a operação do algoritmo DV para a rede simples de três nós mostrada na parte superior da figura. A operação do algoritmo é ilustrada de um modo síncrono, no qual todos os nós recebem vetores de distâncias simultaneamente de seus vizinhos, calculam seus novos vetores de distâncias e informam a seus vizinhos se esses vetores mudaram. Após estudar esse exemplo, você deve se convencer de que o algoritmo também opera corretamente em modo assíncrono, com cálculos de nós e atualizações de geração/recepção ocorrendo a qualquer instante.

A coluna mais à esquerda na figura mostra três **tabelas de roteamento** iniciais para cada um dos três nós. Por exemplo, a tabela no canto superior à esquerda é a tabela de roteamento inicial do nó x . Dentro de uma tabela de roteamento específica, cada linha é um vetor de distâncias — especificamente, a tabela de roteamento de cada nó inclui seu próprio vetor de distâncias e os vetores de cada um de seus vizinhos. Assim, a primeira linha da tabela de roteamento inicial do nó x é $D_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. A segunda linha e a terceira linha nessa tabela são os vetores de distâncias recebidos mais recentemente dos nós y e z , respectivamente. Como na inicialização o nó x não recebeu nada do nó y ou z , os registros da segunda linha e da terceira linha estão definidos como infinito.

Após a inicialização, cada nó envia seu vetor de distâncias a cada um de seus dois vizinhos. Isso é ilustrado na Figura 4.30 pelas setas que vão da primeira coluna de tabelas até a segunda coluna de tabelas. Por exemplo, o nó x envia seu vetor de distâncias $D_x = [0, 2, 7]$ a ambos os nós y e z . Após receber as atualizações, cada nó recalcula seu próprio vetor de distâncias. Por exemplo, o nó x calcula

$$\begin{aligned} D_x(x) &= 0 \\ D_x(y) &= \min\{c(x, y) + D_y(y), c(x, z) + D_z(y)\} = \min\{2 + 0, 7 + 1\} = 2 \\ D_x(z) &= \min\{c(x, y) + D_y(z), c(x, z) + D_z(z)\} = \min\{2 + 1, 7 + 0\} = 3 \end{aligned}$$

Por conseguinte, a segunda coluna mostra, para cada nó, o novo vetor de distâncias do nó, juntamente com os vetores de distâncias que acabou de receber de seus vizinhos. Observe, por exemplo, que a estimativa do nó x para o menor custo até o nó z , $D_x(z)$, mudou de 7 para 3. Note também que, para ambos os nós y e z , o nó y representa os mínimos correspondentes. Assim, nesse estágio do algoritmo, os roteadores do próximo salto são $v^*(y) = y$ e $v^*(z) = y$.

Depois que recalculam seus vetores de distâncias, os nós enviam novamente seus vetores de distâncias recalculados a seus vizinhos (se houver uma mudança). Isso é ilustrado na Figura 4.30 pelas setas que vão da segunda coluna de tabelas até a terceira coluna de tabelas. Note que somente os nós x e z enviam atualizações: o vetor de distâncias do nó y não mudou, então o nó y não envia uma atualização. Após receber as atualizações, os nós então recalculam seus vetores de distâncias e atualizam suas tabelas de roteamento, que são mostradas na terceira coluna.

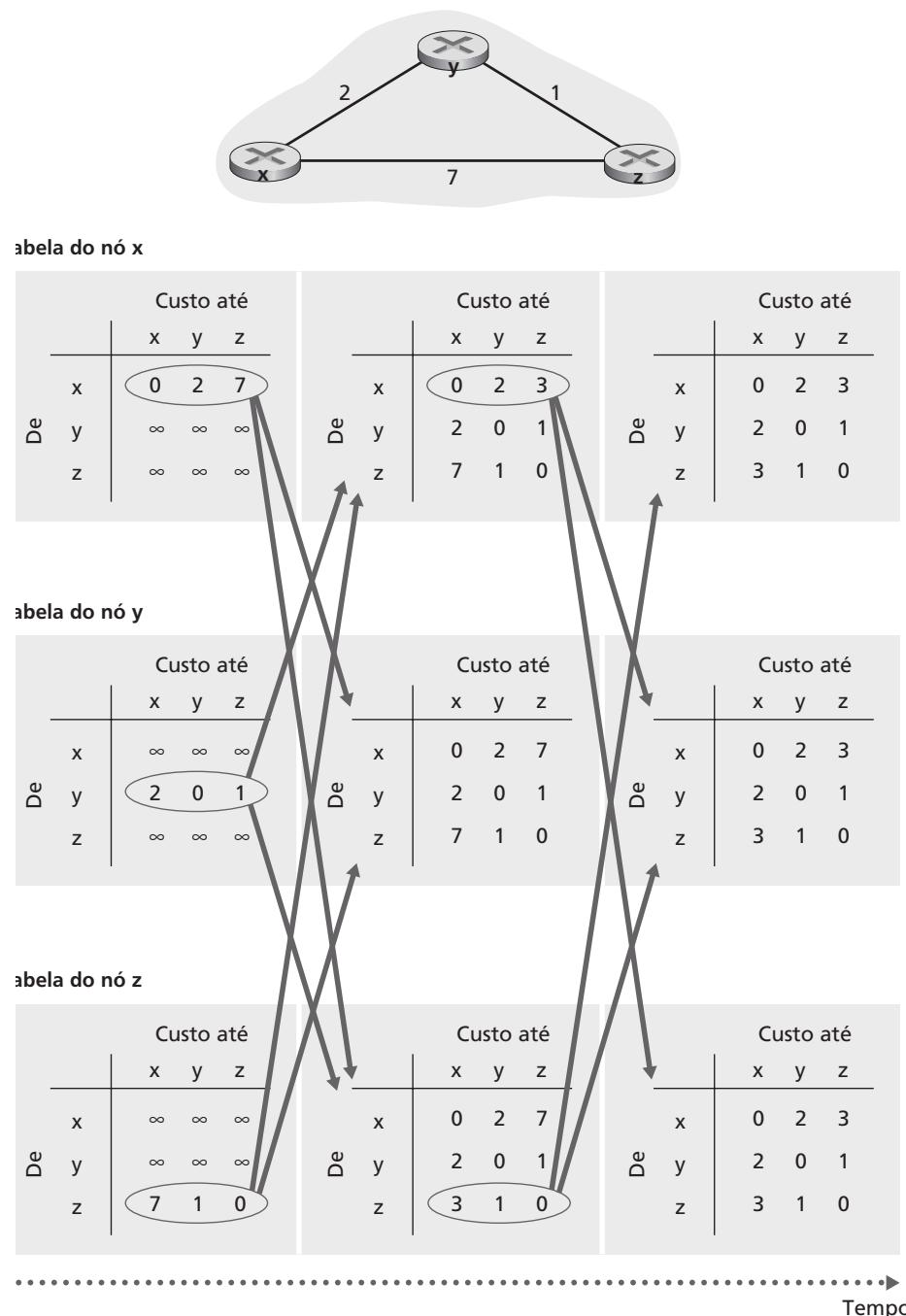


Figura 4.30 Algoritmo de vetor de distâncias (DV)

O processo de receber vetores de distâncias atualizados de vizinhos, recalcular os registros de tabelas de roteamento e informar aos vizinhos os custos modificados do caminho de menor custo até o destino continua até que mais nenhuma mensagem de atualização seja enviada. Nesse ponto, já que mais nenhuma mensagem de atualização é enviada, não ocorrerá mais nenhum cálculo de tabela de roteamento e o algoritmo entra em estado de inatividade; isto é, todos os nós estarão rodando a espera nas Linhas 10-11 do algoritmo DV. O algoritmo permanece no estado de inatividade até que o custo de um enlace mude, como discutiremos a seguir.

Algoritmo de vetor de distâncias: mudanças no custo do enlace e falha no enlace

Quando um nó que está rodando o algoritmo DV detecta uma mudança no custo do enlace dele mesmo até um vizinho (Linhas 10-11), ele atualiza seu vetor de distâncias (Linhas 13-14) e, se houver uma modificação no custo do caminho de menor custo, informa a seus vizinhos (Linhas 16-17) seu novo vetor de distâncias. A Figura 4.31(a) ilustra um cenário em que o custo do enlace de y a x muda de 4 para 1. Destacamos aqui somente os registros na tabela de distâncias de y e z até o destino x . O algoritmo DV causa a ocorrência da seguinte sequência de eventos:

- No tempo t_0 , y detecta a mudança no custo do enlace (o custo mudou de 4 para 1), atualiza seu vetor de distâncias e informa essa mudança a seus vizinhos, já que o vetor mudou.
- No tempo t_1 , z recebe a atualização de y e atualiza sua própria tabela. Calcula um novo menor custo para x (cujo custo diminuiu de 5 para 2), e envia seu novo vetor de distâncias a seus vizinhos.
- No tempo t_2 , y recebe a atualização de z e atualiza sua tabela de distâncias. Os menores custos de y não mudaram e, por conseguinte, y não envia nenhuma mensagem a z . O algoritmo entra em estado de inatividade.

Assim, apenas duas iterações são necessárias para o algoritmo DV alcançar o estado de inatividade. A boa notícia sobre a redução do custo entre x e y se propagou rapidamente pela rede.

Agora vamos considerar o que pode acontecer quando o custo de um enlace *aumenta*. Suponha que o custo do enlace entre x e y aumente de 4 para 60, como mostra a Figura 4.31(b).

1. Antes da mudança do custo do enlace, $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$, e $D_z(x) = 5$. No tempo t_0 , y detecta uma mudança no custo do enlace (o custo mudou de 4 para 60). y calcula seu novo caminho de custo mínimo até x , de modo a ter um custo de

$$D_y(x) = \min\{c(y, x) + D_x(y), c(y, z) + D_z(x)\} = \min\{60 + 0,1 + 5\} = 6$$

É claro que, com nossa visão global da rede, podemos ver que esse novo custo via z está errado. Mas as únicas informações que o nó y tem é que seu custo direto até x é 60 e que z disse a y que z pode chegar a x com um custo de 5. Assim, para chegar a x , y teria de fazer a rota através de z , com a expectativa de que z será capaz de chegar a x com um custo de 5. A partir de t_1 , temos um **loop de roteamento** — para poder chegar a x , y faz a rota através de z , que por sua vez faz a rota através de y . Um loop de roteamento é como um buraco negro — um pacote destinado a x que chegar a y ou a z a partir do momento t_1 vai ricochetear entre um e outro desses dois nós para sempre (ou até que as tabelas de repasse sejam mudadas).

2. Tão logo o nó y tenha calculado um novo custo mínimo até x , ele informará a z esse novo vetor de distâncias no tempo t_1 .
3. Algum tempo depois de t_1 , z recebe o novo vetor de distâncias de y , que indica que o custo mínimo de y até x é 6. z sabe que pode chegar até y com um custo de 1 e, por conseguinte, calcula um novo menor custo até x , $D_z(x) = \min\{50 + 0,1 + 6\} = 7$. Uma vez que o custo mínimo de z até x aumentou, z informa a y o seu novo vetor de distâncias em t_2 .

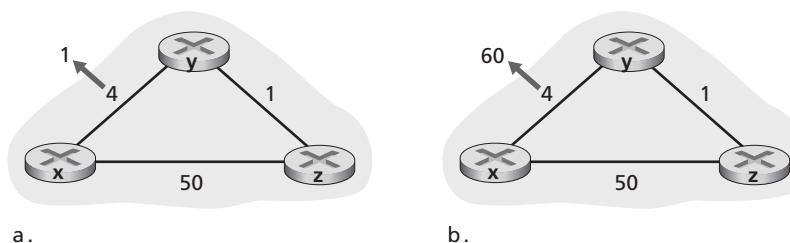


Figura 4.31 Mudanças no custo do enlace

4. De maneira semelhante, após receber o novo vetor de distâncias de z , y determina $D_y(x) = 8$ e envia a z seu vetor de distâncias. Então z determina $D_z(x) = 9$ e envia a y seu vetor de distâncias e assim por diante.

Por quanto tempo esse processo continua? Pode ter certeza de que o loop persistirá por 44 iterações (trocas de mensagens entre y e z) até que z possa, finalmente, calcular que o custo de seu caminho via y é maior do que 50. Nesse ponto, z (finalmente!) determinará que seu caminho de menor custo até x é via sua conexão direta com x . y então fará a rota até x via z . O resultado das más notícias sobre o aumento do custo do enlace na verdade viajou devagar! O que teria acontecido se o custo do enlace $c(y, x)$ tivesse mudado de 4 para 10.000 e o custo $c(z, x)$ fosse 9999? Por causa de cenários como esses, o problema que acabamos de examinar é, às vezes, denominado problema de contagem ao infinito.

Algoritmo de vetor de distâncias: adição de reversão envenenada

O cenário específico de looping que acabamos de descrever pode ser evitado usando uma técnica denominada reversão envenenada (*poisoned reverse*). A ideia é simples — se a rota de z para chegar a x passa por y , então z deve anunciar a y que sua distância a x é infinita, isto é, z anunciará a y que $D_z(x) = \infty$ (mesmo que z saiba que, na verdade, $D_z(x) = 5$). z continuará contando essa mentirinha inocente a y enquanto a rota de z a x estiver passando por y . Enquanto y acreditar que z não tem nenhum caminho até x , y jamais tentará a rota até x por z , contanto que a rota de z a x continue a passar por y (e ele minta sobre isso).

Agora vamos ver como a reversão envenenada resolve o problema específico do looping que encontramos anteriormente na Figura 4.31(b). Como resultado da reversão envenenada, a tabela de distâncias de y indica que $D_z(x) = \infty$. Quando o custo do enlace (x, y) muda de 4 para 60 no tempo t_0 , y atualiza sua tabela e continua a estabelecer rotas diretamente para x , embora com um custo mais alto do que 60, e informa z o seu novo custo até x , isto é, $D_y(x) = 60$. Após receber a atualização em t_1 , z imediatamente desloca sua rota para x , para que seja via o enlace direto (z, x) a um custo de 50. Como este é um novo menor custo até x , e uma vez que o caminho não passa mais por y , z agora informa a y que $D_z(x) = 50$ em t_2 . Após receber a atualização de z , y atualiza sua tabela de distâncias com $D_y(x) = 51$. E, também, uma vez que z está agora no caminho de menor custo até x , y envenena o caminho inverso de z a x , informando a z , no tempo t_3 , que $D_y(x) = \infty$ (mesmo que y saiba que, na verdade, $D_y(x) = 51$).

A reversão envenenada resolve o problema geral da contagem até o infinito? Não resolve. É bom que você se convença de que loops que envolvem três ou mais nós (e não apenas dois nós imediatamente vizinhos) não serão detectados pela técnica da reversão envenenada.

Uma comparação entre os algoritmos de roteamento de estado de enlace e de vetor de distâncias

Os algoritmos DV e LS adotam abordagens complementares em relação ao cálculo do roteamento. No algoritmo DV, cada nó fala *somente* com os vizinhos diretamente conectados a ele, mas informa a esses vizinhos as estimativas de menor custo entre ele mesmo e *todos* os outros nós da rede (isto é, todos os que ele sabe que existem). No algoritmo LS, cada nó fala com *todos* os outros nós (via *broadcast*), mas informa *somente* os custos dos enlaces diretamente ligados a ele. Vamos concluir nosso estudo sobre algoritmos de estado de enlace e de vetor de distâncias com uma rápida comparação de alguns de seus atributos. Lembre-se de que N é o conjunto de nós (roteadores) e E é o conjunto de arestas (enlaces).

Complexidade da mensagem. Vimos que o LS requer que cada nó saiba o custo de cada enlace da rede. Isso exige que sejam enviadas $O(|N| |E|)$. E, também, sempre que o custo de um enlace muda, o novo custo deve ser enviado a todos os nós. O algoritmo DV requer troca de mensagens entre vizinhos diretamente conectados a cada iteração. Já vimos que o tempo necessário para que o algoritmo converja pode depender de muitos fatores. Quando o custo do enlace muda, o algoritmo DV propaga os resultados do custo modificado do enlace somente se o novo custo resultar em mudança no caminho de menor custo para um dos nós ligado ao enlace.

Velocidade de convergência. Já vimos que nossa implementação de LS é um algoritmo $O(|N|^2)$ que requer $O(|N| |E|)$ mensagens. O algoritmo DV pode convergir lentamente e pode ter loops de roteamento enquanto estiver convergindo. O algoritmo DV também tem o problema da contagem até o infinito.

Robustez. O que pode acontecer se um roteador falhar, se comportar mal ou for sabotado? Sob o LS, um roteador poderia transmitir um custo incorreto para um de seus enlaces diretos (mas não para outros). Um nó poderia também corromper ou descartar quaisquer pacotes recebidos como parte de uma transmissão de estado de enlace. Mas um nó LS está calculando somente suas próprias tabelas de roteamento; os outros nós estão realizando cálculos semelhantes para si próprios. Isso significa que, sob o LS, os cálculos de rota são, de certa forma, isolados, fornecendo um grau de robustez. Sob o DV, um nó pode anunciar incorretamente caminhos de menor custo para qualquer destino, ou para todos os destinos. (Na verdade, em 1997, um roteador que estava funcionando mal em um pequeno ISP forneceu aos roteadores nacionais de backbone tabelas de roteamento errôneas. Isso fez com que outros roteadores inundassem de tráfego o roteador que estava funcionando mal. Com isso, grandes porções da Internet ficaram desconectadas durante muitas horas [Neumann, 1997].) De modo geral, notamos que, a cada iteração, um cálculo de nó em DV é passado adiante a seu vizinho e, em seguida, indiretamente ao vizinho de seu vizinho na iteração seguinte. Nesse sentido, sob o DV, um cálculo incorreto do nó pode ser difundido pela rede inteira.

No final, nenhum algoritmo ganha do outro. Realmente, ambos são usados na Internet.

Outros algoritmos de roteamento

Os algoritmos LS e DV que estudamos não somente são utilizados em grande escala na prática, mas também são, essencialmente, os únicos algoritmos utilizados hoje na Internet.

Não obstante, muitos algoritmos de roteamento foram propostos por pesquisadores nos últimos 30 anos, abrangendo desde o extremamente simples até o muito sofisticado e complexo. Há uma grande classe de algoritmos de roteamento cuja base é considerar o tráfego como fluxos entre fontes e destinos em uma rede. Nessa abordagem, o problema do roteamento pode ser formulado em termos matemáticos como um problema de otimização restrita, conhecido como problema de fluxo da rede [Bertsekas, 1991]. Ainda um outro conjunto de algoritmos de roteamento que mencionamos aqui são os derivados do mundo da telefonia. Esses **algoritmos de roteamento de comutação de circuitos** são de interesse para redes de comutação de circuitos nos casos em que devem ser reservados recursos (por exemplo, buffers ou uma fração da largura de banda do enlace) por enlace para cada conexão roteada pelo enlace. Embora a formulação do problema do roteamento talvez pareça bem diferente da formulação do problema do roteamento de menor custo que vimos neste capítulo, existem muitas semelhanças, ao menos no que se refere ao algoritmo de busca de caminhos (algoritmo de roteamento). Veja [Ash, 1998; Ross, 1995; Girard, 1990] para uma discussão detalhada dessa área de pesquisa.

4.5.3 Roteamento hierárquico

Quando estudamos os algoritmos LS e DV, consideramos a rede simplesmente como uma coleção de roteadores interconectados. Um roteador não se distinguia de outro no sentido de que todos rodavam o mesmo algoritmo de roteamento para calcular os caminhos de roteamento pela rede inteira. Na prática, esse modelo e sua visão de um conjunto homogêneo de roteadores, todos rodando o mesmo algoritmo de roteamento, é um tanto simplista por pelo menos duas razões importantes:

Escala. À medida que aumenta o número de roteadores, a sobrecarga relativa ao cálculo, ao armazenamento e à comunicação de informações para a tabela de roteamento (por exemplo, atualizações de estado de enlace ou de mudança no caminho de menor custo) se torna proibitiva. A Internet pública de hoje consiste em centenas de milhões de hospedeiros. Armazenar informações de roteamento para cada um desses hospedeiros evidentemente exigiria quantidades enormes de memória. Com a sobrecarga exigida para transmitir atualizações do estado de enlace entre todos os roteadores da Internet não sobraria nenhuma largura de banda para enviar pacotes de dados! Um algoritmo DV que fizesse iterações entre esse número tão grande de roteadores seguramente jamais convergiria! Fica claro que algo deve ser feito para reduzir a complexidade do cálculo de rotas em redes tão grandes como a Internet pública.

Autonomia administrativa. Embora os pesquisadores tendam a ignorar questões como o desejo das empresas de controlar seus roteadores como bem entendem (por exemplo, rodar qualquer algoritmo de roteamento que escolherem) ou ocultar do público externo aspectos da organização interna das redes, essas considerações são importantes. Idealmente, uma organização deveria poder executar e administrar sua rede como bem entendesse, mantendo a capacidade de conectar sua rede a outras redes externas.

Esses problemas podem ser resolvidos agrupando roteadores em **sistemas autônomos** (*autonomous systems* — ASs), com cada AS consistindo em um grupo de roteadores sob o mesmo controle administrativo (isto é, operados pelo mesmo ISP ou pertencentes a uma mesma rede corporativa). Todos os roteadores dentro do mesmo AS rodam o mesmo algoritmo de roteamento (por exemplo, algoritmo LS ou DV) e dispõem das informações sobre cada um dos outros — exatamente como foi o caso do modelo idealizado da seção anterior. O algoritmo de roteamento que roda dentro de um AS é denominado um **protocolo de roteamento intrassistema autônomo**. É claro que será necessário conectar os ASs entre si e, assim, um ou mais dos roteadores em um AS terá a tarefa adicional de ficar responsável por transmitir pacotes a destinos externos ao AS — esses roteadores são denominados **roteadores de borda** (*gateway routers*).

A Figura 4.32 ilustra um exemplo simples com três ASs: AS1, AS2 e AS3. Na figura, as linhas escuras representam conexões diretas de enlaces entre pares e roteadores. As linhas mais finas e interrompidas que saem dos roteadores representam sub-redes conectadas diretamente a eles. O AS1 tem quatro roteadores, 1.a, 1.b, 1.c e 1.d, e cada qual roda o protocolo de roteamento utilizado dentro do AS1. Assim, cada um desses quatro roteadores sabe como transmitir pacotes ao longo do caminho ótimo para qualquer destino dentro de AS1. De maneira semelhante, cada um dos sistemas autônomos AS2 e AS3 tem três roteadores. Note que os protocolos de roteamento intra-AS que rodam em AS1, AS2 e AS3 não precisam ser os mesmos. Note também que os roteadores 1b, 1c, 2a e 3a são roteadores de borda.

Agora já deve estar claro como os roteadores em um AS determinam caminhos de roteamento para pares fonte-destino internos ao sistema. Mas ainda há uma peça faltando nesse quebra-cabeça de roteamento fim a fim. Como um roteador que está dentro de algum AS sabe como rotear um pacote até um destino que está fora do AS? Essa pergunta é fácil de responder se o AS tiver somente um roteador de borda que se conecta com somente um outro AS. Nesse caso, como o algoritmo de roteamento intra-AS do sistema autônomo determinou o caminho de menor custo entre cada roteador interno e o roteador de borda, cada roteador interno sabe como deve enviar o pacote. Ao receber o pacote, o roteador de borda o repassa para o único enlace que leva ao exterior do AS. Então, o AS que está na outra

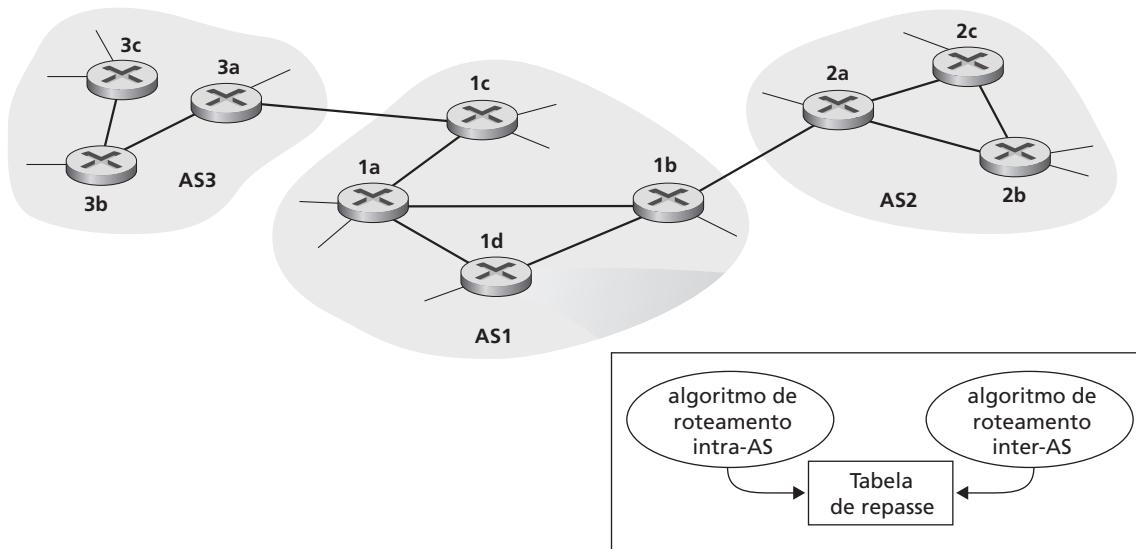


Figura 4.32 Um exemplo de sistemas autônomos interconectados

extremidade do enlace assume a responsabilidade de rotear o pacote até seu destino final. Como exemplo, suponha que o roteador 2b da Figura 4.32 receba um pacote cujo destino está fora do AS2. O roteador 2b então transmite o pacote ao roteador 2a ou 2c, como especificado pela tabela de repasse do roteador 2b, que foi configurada pelo protocolo de roteamento intra-AS de AS2. O pacote eventualmente chegará ao roteador de borda 2a, que o retransmitirá ao roteador 1b. Tão logo o pacote tenha saído de 2a, termina o trabalho de AS2 com referência a esse pacote.

Portanto, o problema é fácil quando o AS de origem tem somente um enlace que leva para fora do AS. Mas, e se o AS de origem tiver dois ou mais enlaces (passando por um ou mais roteadores de borda) que levam para fora do AS? Então, o problema de saber para onde repassar o pacote torna-se significativamente mais desafiador. Por exemplo, considere um roteador em AS1 e suponha que ele recebe um pacote cujo destino está fora do AS. É claro que o roteador deveria repassar o pacote para um de seus dois roteadores de borda, 1b ou 1c, mas para qual deles? Para resolver esse problema, AS1 (1) precisa saber quais destinos podem ser alcançados via o AS2 e quais podem ser alcançados via o AS3 e (2) precisa propagar essa informação para todos os roteadores dentro de AS1, de modo que cada roteador possa configurar sua tabela de repasse para manipular destinos externos ao AS. Essas duas tarefas — obter informações sobre as condições de alcance de ASs vizinhos e propagar essas informações a todos os outros roteadores internos a AS — são gerenciadas pelo **protocolo de roteamento inter-AS**. Visto que o protocolo de roteamento inter-AS envolve comunicação entre dois ASs, esses dois ASs comunicantes devem rodar o mesmo protocolo de roteamento inter-AS. Realmente, na Internet, todos os ASs rodam o mesmo protocolo de roteamento inter-AS, denominado BGP4, que discutiremos na próxima seção. Como ilustrado na Figura 4.32, cada roteador recebe informações de um protocolo de roteamento intra-AS e de um protocolo de roteamento inter-AS, e usa as informações de ambos os protocolos para configurar sua tabela de repasse.

Como exemplo, considere uma sub-rede x (identificada por seu endereço “ciderizado”) e suponha que AS1 sabe, por meio do protocolo de roteamento inter-AS, que a sub-rede x pode ser alcançada de AS3, mas não pode ser alcançada de AS2. Então, AS1 propaga essa informação a todos os seus roteadores. Quando o roteador 1d fica sabendo que a sub-rede x pode ser alcançada de AS3 e, por conseguinte, do roteador de borda 1c, determina, com base na informação fornecida pelo protocolo de roteamento intra-AS, a interface de roteador que está no caminho de menor custo entre o roteador 1d e o roteador de borda 1c. Seja I essa interface. O roteador 1d então pode colocar o registro (x, I) , em sua tabela de repasse. (Esse exemplo, e outros apresentados nesta seção, passam as ideias gerais, mas são simplificações do que realmente acontece na Internet. Na seção seguinte daremos uma descrição mais detalhada, se bem que mais complicada, quando discutirmos o BGP.)

Continuando com nosso exemplo anterior, suponha agora que AS2 e AS3 estão conectados com outros ASs, que não aparecem no diagrama. Suponha também que AS1 fica sabendo, pelo protocolo de roteamento inter-AS, que a sub-rede x pode ser alcançada de AS2, via o roteador de borda 1b, e também de AS3, via o roteador de borda 1c. Então, AS1 propagaria essa informação a todos os seus roteadores, incluindo o roteador 1d. Para configurar sua tabela de repasse, o roteador 1d teria de determinar para qual roteador de borda, 1b ou 1c, deve dirigir pacotes destinados à sub-rede x . Uma abordagem, que é frequentemente empregada na prática, é utilizar o **roteamento da batata quente**. Com esse roteamento, o AS se livra do pacote (a batata quente) o mais rapidamente possível (mais exatamente, com o menor custo possível). Isso é feito obrigando um roteador a enviar o pacote ao roteador de borda que tiver o menor custo roteador-roteador de borda entre todos os roteadores de borda que têm um caminho para o destino. No contexto do exemplo que estamos examinando, o roteamento da batata quente, rodando em 1d, usaria informação recebida do protocolo de roteamento intra-AS para determinar os custos de caminho até 1b e 1c, e então escolheria o caminho de menor custo. Uma vez escolhido este caminho, o roteador 1d adiciona em sua tabela de repasse um registro para a sub-rede x . A Figura 4.33 resume as ações executadas no roteador 1d para adicionar à tabela de repasse o novo registro para x .

Quando um AS fica sabendo de um destino por meio de um AS vizinho, ele pode anunciar essa informação de roteamento a alguns de seus outros ASs vizinhos. Por exemplo, suponha que AS1 fica sabendo, por AS2, que a sub-rede x pode ser alcançada via AS2. Então, AS1 poderia dizer a AS3 que x pode ser alcançada via AS1. Desse modo, se AS3 precisar rotear um pacote destinado a x , repassaria o pacote a AS1 que, por sua vez, repassaria o pacote para AS2. Como veremos quando discutirmos BGP, um AS tem bastante flexibilidade para decidir quais destinos anuncia a seus ASs vizinhos. Esta é uma decisão política que normalmente depende mais de questões econômicas do que técnicas.

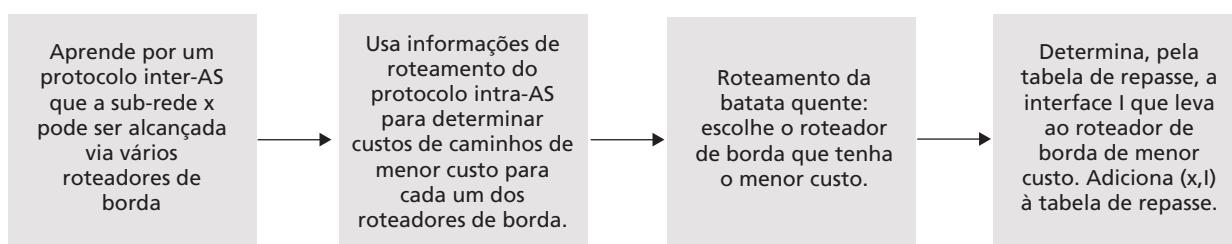


Figura 4.33 Etapas da adição de um destino fora do AS à tabela de repasse de um roteador

Lembre-se de que dissemos na Seção 1.5 que a Internet consiste em uma hierarquia de ISPs interconectados. Então, qual é a relação entre ISPs e ASs? Você talvez pense que os roteadores em um ISP e os enlaces que os interconectam constituem um único AS. Embora esse seja frequentemente o caso, muitos ISPs dividem sua rede em vários ASs. Por exemplo, alguns ISPs de nível 1 utilizam um AS para toda a sua rede; outros subdividem sua rede em dezenas de ASs interconectados.

Em resumo, os problemas de escala e de autoridade administrativa são resolvidos pela definição de sistemas autônomos. Dentro de um AS, todos os roteadores rodam o mesmo protocolo de roteamento intrassistema autônomo. Entre eles, os ASs rodam o mesmo protocolo de roteamento inter-AS. O problema de escala é resolvido porque um roteador intra-AS precisa saber apenas dos outros roteadores dentro do AS. O problema de autoridade administrativa é resolvido, já que uma organização pode rodar o protocolo de roteamento intra-AS que quiser; todavia, cada par de ASs conectados precisa rodar o mesmo protocolo de roteamento inter-AS para trocar informações de alcance.

Na seção seguinte, examinaremos dois protocolos de roteamento intra-AS (RIP e OSPF) e o protocolo inter-AS (BGP), que são usados na Internet de hoje. Esses estudos de casos dão um bom arremate ao nosso estudo de roteamento hierárquico.

4.6 Roteamento na Internet

Agora que já estudamos endereçamento na Internet e o protocolo IP, vamos voltar nossa atenção aos protocolos de roteamento da Internet. A tarefa desses protocolos é determinar o caminho tomado por um datagrama entre a fonte e o destino. Veremos que os protocolos de roteamento da Internet incorporam muitos dos princípios que aprendemos anteriormente neste capítulo. As abordagens de estado de enlace e de vetor de distâncias estudadas nas seções 4.5.1 e 4.5.2 e a ideia de um sistema autônomo considerada na Seção 4.5.3 são fundamentais para o modo como o roteamento é feito na Internet de hoje.

Lembre-se de que na Seção 4.5.3 vimos que um sistema autônomo (AS) é um conjunto de roteadores que estão sob o mesmo controle administrativo e técnico e que rodam, todos, o mesmo protocolo de roteamento entre eles. Cada AS, por sua vez, normalmente contém várias sub-redes (aqui, usamos o termo sub-rede no sentido preciso de endereçamento, como na Seção 4.4.2).

4.6.1 Roteamento intra-AS na Internet: RIP

Um protocolo de roteamento intra-AS é usado para determinar como é rodado o roteamento dentro de um sistema autônomo (AS). Protocolos de roteamento intra-AS são também conhecidos como **protocolos de roteadores internos** (IGP — *Interior Gateway Protocols*). Historicamente, dois protocolos de roteamento têm sido usados extensivamente para roteamento dentro de um sistema autônomo na Internet: o **RIP** (*Routing Information Protocol* — Protocolo de Informação de Roteamento) e o **OSPF** (*open shortest path first*). Um protocolo muito relacionado com o OSPF é o protocolo **IS-IS** [RFC 1142, Perlman, 1999]. Em primeiro lugar, discutiremos o RIP e, em seguida, consideraremos o OSPF.

O RIP foi um dos primeiros protocolos de roteamento intra-AS da Internet, e seu uso é ainda amplamente disseminado. Sua origem e seu nome podem ser traçados até a arquitetura XNS (Xerox Network Systems). A ampla disponibilização do RIP se deveu em grande parte à sua inclusão, em 1982, na versão do UNIX do Berkeley Software Distribution (BSD), que suportava TCP/IP. A versão 1 do RIP está definida no [RFC 1058] e a versão 2, compatível com a versão 1, no [RFC 2453].

O RIP é um protocolo de vetor de distâncias que funciona de um modo muito parecido com o protocolo DV idealizado que examinamos na Seção 4.5.2. A versão do RIP especificada no RFC 1058 usa contagem de saltos como métrica de custo, isto é, cada enlace tem um custo 1. Por simplicidade, no algoritmo DV da Seção 4.5.2 os custos foram definidos entre pares de roteadores. No RIP (e também no OSPF), na realidade, os custos são definidos desde um roteador de origem até uma sub-rede de destino. O RIP usa o termo *salto*, que é o número de sub-redes percorridas ao longo do caminho mais curto entre o roteador de origem e uma sub-rede de destino, incluindo a sub-rede de destino. A Figura 4.34 ilustra um AS com seis sub-redes folha. A tabela da figura indica o número de saltos desde o roteador de origem A até todas as sub-redes folha.

O custo máximo de um caminho é limitado a 15, limitando assim o uso do RIP a sistemas autônomos que têm menos de 15 saltos de diâmetro. Lembre-se de que em protocolos DV, roteadores vizinhos trocam vetores de distância entre si. O vetor de distâncias para qualquer roteador é a estimativa corrente das distâncias dos caminhos de menor custo entre aquele roteador e as sub-redes no AS. No RIP, tabelas de roteamento são trocadas entre vizinhos a cada 30 segundos aproximadamente, usando uma mensagem de **resposta RIP**. A mensagem de resposta enviada por um roteador ou um hospedeiro contém uma lista de até 25 sub-redes de destino dentro do AS, bem como as distâncias entre o remetente e cada uma dessas sub-redes. Mensagens de resposta também são conhecidas como **anúncios RIP**.

Vamos examinar um exemplo simples de como funcionam os anúncios RIP. Considere a parte de um AS mostrada na Figura 4.35. Nessa figura, as linhas que conectam os roteadores representam sub-redes. Somente os roteadores (A, B, C e D) e as redes (w, x, y, z) selecionados são rotulados. As linhas tracejadas indicam que o AS continua; portanto, esse sistema autônomo tem muito mais roteadores e enlaces do que os mostrados na figura.

Cada roteador mantém uma tabela RIP denominada **tabela de roteamento**. A tabela de roteamento de um roteador inclui o vetor de distâncias e a tabela de repasse desse roteador. A Figura 4.36 mostra a tabela de roteamento do roteador D. Note que a tabela de roteamento tem três colunas. A primeira coluna é para a sub-rede de destino, a segunda coluna indica a identidade do roteador seguinte ao longo do caminho mais curto até a sub-rede de destino e a terceira coluna indica o número de saltos (isto é, o número de sub-redes que têm de ser atravessadas, incluindo a rede de destino) para chegar à sub-rede de destino ao longo do caminho mais curto. Para este exemplo, a tabela mostra que, para enviar um datagrama do roteador D até a sub-rede de destino w, o datagrama deve primeiramente ser enviado ao roteador vizinho A; a tabela também mostra que a sub-rede de destino w está a dois saltos de distância ao longo do caminho mais curto. De modo semelhante, a tabela indica que a sub-rede z está a sete saltos de distância via o roteador B. Em princípio, uma tabela de roteamento terá apenas uma linha para cada sub-rede no AS, embora a versão 2 do RIP permita a agregação de registros de sub-redes usando técnicas de agregação de rotas semelhantes às que examinamos na Seção 4.4. A tabela na Figura 4.36 e as tabelas subsequentes estão apenas parcialmente completas.

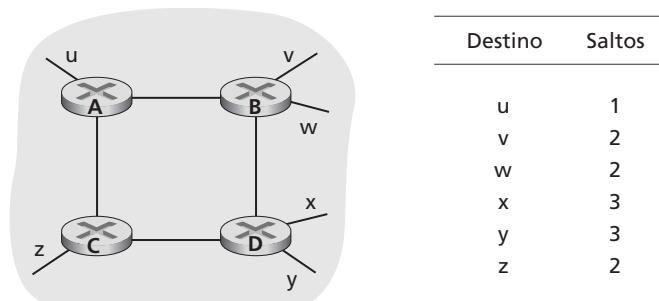


Figura 4.34 Número de saltos do roteador de origem, A, até várias sub-redes

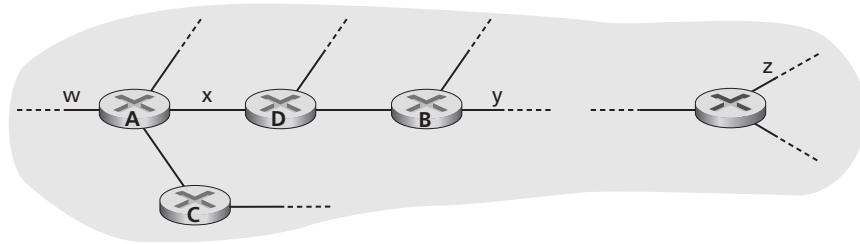


Figura 4.35 Uma parte de um sistema autônomo

Suponha agora que 30 segundos mais tarde o roteador *D* receba do roteador *A* o anúncio mostrado na Figura 4.37. Note que esse anúncio nada mais é do que informações da tabela de roteamento do roteador *A*! Essa informação indica, em particular, que a sub-rede *z* está a apenas quatro saltos do roteador *A*. Ao receber esse anúncio, o roteador *D* o funde (Figura 4.37) com a tabela de roteamento antiga (Figura 4.36). Em particular, o roteador *D* fica sabendo que agora há um novo caminho através do roteador *A* até a sub-rede *z* que é mais curto do que o caminho pelo roteador *B*. Assim, o roteador *D* atualiza sua tabela de roteamento para levar em conta o mais curto dos caminhos mais curtos, como mostra a Figura 4.38. Você poderia perguntar como o caminho mais curto até a sub-rede *z* se tornou mais curto ainda? Possivelmente, o algoritmo de vetor de distâncias descentralizado ainda estava em processo de convergência (veja a Seção 4.5.2) ou, talvez, novos enlaces e/ou roteadores tenham sido adicionados ao AS, mudando assim os caminhos mais curtos dentro dele.

Vamos agora considerar alguns dos aspectos da implementação do RIP. Lembre-se de que os roteadores RIP trocam anúncios a cada 30 segundos aproximadamente. Se um roteador não ouvir nada de seu vizinho ao menos uma vez a cada 180 segundos, esse vizinho será considerado impossível de ser alcançado dali em diante, isto é, o vizinho está inoperante ou o enlace de conexão caiu. Quando isso acontece, o RIP modifica a tabela de roteamento local e, em seguida, propaga essa informação enviando anúncios a seus roteadores vizinhos (os que ainda podem ser alcançados). Um roteador pode também requisitar informação sobre o custo de seu vizinho até um dado destino usando uma mensagem de requisição RIP. Roteadores enviam mensagens RIP de requisição e de resposta uns aos outros usando o número de porta 520. O segmento UDP é carregado entre roteadores dentro de um datagrama IP padrão. O fato de o RIP usar um protocolo de camada de transporte (UDP) sobre um protocolo de camada de rede (IP) para implementar funcionalidade de camada de rede (um algoritmo de roteamento) pode parecer bastante confuso (e é!). Um exame mais profundo sobre como o RIP é implementado esclarecerá esse assunto.

A Figura 4.39 ilustra esquematicamente como o RIP é comumente implementado em um sistema UNIX, como, por exemplo, uma estação de trabalho UNIX que está servindo como um roteador. Um processo denominado *routed* roda o RIP, isto é, mantém informações de roteamento e troca mensagens com processos *routed* que rodam em roteadores vizinhos. Como o RIP é implementado como um processo da camada de aplicação (se bem que um processo muito especial, que é capaz de manipular as tabelas de roteamento dentro do núcleo do UNIX), ele pode enviar e receber mensagens por uma porta padrão e usar um protocolo de transporte padrão. Assim, o RIP é um protocolo de camada de aplicação (veja o Capítulo 2) que roda sobre UDP.

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
w	A	2
y	B	2
z	B	7
x	—	1
....

Figura 4.36 Tabela de roteamento no roteador *D* antes de receber anúncio do roteador *A*

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
z	C	4
w	—	1
x	—	1
....

Figura 4.37 Anúncio vindo do roteador A

4.6.2 Roteamento intra-AS na Internet: OSPF

Como o RIP, o roteamento OSPF é usado amplamente para roteamento intra-AS na Internet. O OSPF e seu primo, IS-IS, muito parecido com ele, são comumente disponibilizados em ISPs de níveis mais altos, ao passo que o RIP é disponibilizado em ISPs de níveis mais baixos e redes corporativas. O ‘open’ do OSPF significa que as especificações do protocolo de roteamento estão disponíveis ao público (ao contrário do protocolo EIGRP da Cisco, por exemplo). A versão mais recente do OSPF, versão 2, está definida no RFC 2178 — um documento público.

O OSPF foi concebido como sucessor do RIP e como tal tem uma série de características avançadas. Em seu âmago, contudo, ele é um protocolo de estado de enlace que usa *broadcasting* de informação de estado de enlace e um algoritmo de caminho de menor custo de Dijkstra. Com o OSPF, um roteador constrói um mapa topológico completo (isto é, um grafo) de todo o sistema autônomo. O roteador então roda localmente o algoritmo do caminho mais curto de Dijkstra para determinar uma árvore de caminho mais curto para todas as sub-redes, sendo ele próprio o nó raiz. Os custos de enlaces individuais são configurados pelo administrador da rede (veja Princípios na prática: determinando pesos OSPF). O administrador pode optar por estabelecer todos os custos de enlace em 1, conseguindo assim roteamento de salto mínimo, ou pode optar por designar para os enlaces pesos inversamente proporcionais à capacidade do enlace de modo a desincentivar o tráfego a usar enlaces de largura de banda baixa. O OSPF não impõe uma política para o modo como são determinados os pesos dos enlaces (essa tarefa é do administrador da rede); em vez disso, provê os mecanismos (protocolo) para determinar o caminho de roteamento de menor custo para um dado conjunto de pesos de enlaces.

Com OSPF, um roteador transmite informações de roteamento a *todos* os outros roteadores no sistema autônomo, e não apenas a seus roteadores vizinhos. Um roteador transmite informações de estado de enlace sempre que houver uma mudança no estado de um enlace (por exemplo, uma mudança de custo ou uma mudança de estado para cima/para baixo). Também transmite o estado de um enlace periodicamente (pelo menos a cada 30 minutos), mesmo que o estado não tenha mudado. O RFC 2328 observa que “essa atualização periódica de anúncios de enlace adiciona robustez ao algoritmo de estado de enlace”. Anúncios OSPF são contidos em mensagens OSPF que são carregadas diretamente por IP, com um código protocolo de camada superior igual a 89 para OSPF. Assim, o próprio protocolo OSPF tem de implementar funcionalidades como transferência confiável de mensagem e transmissão broadcast de estado de enlace. O protocolo OSPF também verifica se os enlaces estão operacionais (via uma mensagem HELLO enviada a um vizinho ligado ao enlace) e permite que um roteador OSPF obtenha o banco de dados de um roteador vizinho referente ao estado do enlace no âmbito da rede.

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
w	A	2
y	B	2
z	A	5
....

Figura 4.38 Tabela de roteamento no roteador D após ter recebido anúncio do roteador A

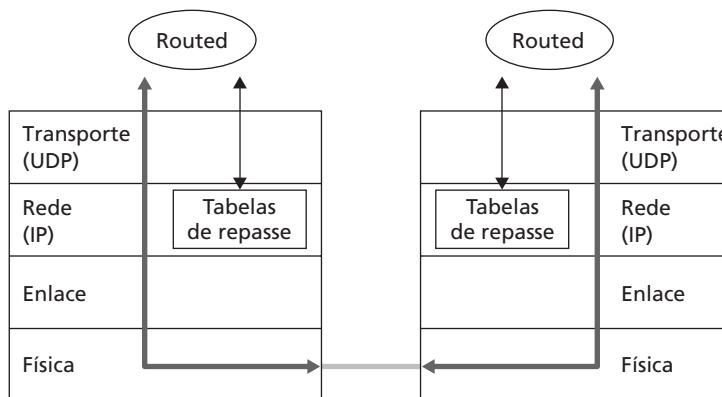


Figura 4.39 Implementação do RIP como um daemon *routed*

Alguns dos avanços incorporados ao OSPF são:

Segurança. Trocas entre roteadores OSPF (por exemplo, atualizações do estado de enlace) são autenticadas. A autenticação garante que apenas roteadores de confiança podem participar do protocolo OSPF dentro de um AS, evitando, assim, que intrusos mal-intencionados (ou estudantes de rede testando, por brincadeira, seu conhecimento recentemente adquirido) injetem informações incorretas em tabelas de roteamento. Por default, pacotes OSPF entre roteadores não são autenticados e poderiam ser forjados. Dois tipos de autenticação podem ser configurados — simples e MD5 (veja o Capítulo 8 para uma discussão sobre MD5 e autenticação em geral). Com autenticação simples, a mesma senha é configurada em cada roteador. Quando um roteador envia um pacote OSPF, inclui a senha em texto normal (não criptografado). É claro que a autenticação simples não é segura. A autenticação MD5 é baseada em chaves secretas compartilhadas que são configuradas em todos os roteadores. Para cada pacote OSPF o roteador calcula o hash MD5 do conteúdo do pacote adicionado com a chave secreta. (Consulte a discussão sobre códigos de autenticação de mensagem no Capítulo 7). O roteador inclui, então, o valor de hash resultante no pacote OSPF. O roteador receptor, usando a chave secreta preconfigurada, calculará um hash MD5 do pacote e comparará com o valor de hash que o pacote carrega, verificando assim a autenticidade do pacote. Números de sequência também são utilizados com autenticação MD5 para proteção contra ataques por reenvio.

Caminhos múltiplos de igual custo. Quando vários caminhos até o destino têm o mesmo custo, o OSPF permite que sejam usados vários caminhos (isto é, não é preciso escolher um caminho único para carregar todo o tráfego quando existem vários caminhos de igual custo).

Suporte integrado para roteamento unicast e multicast. O OSPF Multicast (MOSPF) [RFC 1584] fornece extensões simples ao OSPF para prover roteamento *multicast* (um tópico que examinaremos com mais profundidade na Seção 4.8). O MOSPF usa o banco de dados de enlaces existente no OSPF e adiciona um novo tipo de anúncio de estado de enlace ao mecanismo OSPF existente de transmissão de estado de enlace.

Suporte para hierarquia dentro de um único domínio de roteamento. Talvez o avanço mais significativo do OSPF seja a capacidade de estruturar hierarquicamente um sistema autônomo. Na Seção 4.5.3, vimos as muitas vantagens de estruturas de roteamento hierárquicas. Examinaremos a implementação do roteamento OSPF hierárquico no restante desta seção.

Um sistema autônomo OSPF pode ser configurado hierarquicamente em áreas. Cada área roda seu próprio algoritmo de roteamento de estado de enlace OSPF, sendo que cada roteador em uma área transmite seu estado de enlace a todos os outros roteadores daquela área.

Dentro de cada área, um ou mais **roteadores de borda de área** são responsáveis pelo roteamento de pacotes fora da área. Por fim, exatamente uma área OSPF no AS é configurada para ser a área de **backbone**. O papel



Princípios na prática

Configurando os pesos de enlaces no OSPF

Nossa discussão de roteamento de estado de enlace admitiu implicitamente que os pesos dos enlaces são determinados, que um algoritmo de roteamento como o OSPF é rodado e que o tráfego flui de acordo com as tabelas de roteamento calculadas pelo algoritmo LS. Em termos de causa e efeito, os pesos dos enlaces são dados (isto é, vêm em primeiro lugar) e resultam (via o algoritmo de Dijkstra) em caminhos de roteamento que minimizam o custo geral. Desse ponto de vista, pesos de enlaces refletem o custo da utilização de um enlace (por exemplo, se os pesos dos enlaces forem inversamente proporcionais à capacidade, então a utilização de enlaces de alta capacidade teria pesos menores e, assim, seriam mais atraentes do ponto de vista de roteamento) e o algoritmo de Dijkstra serve para minimizar o custo geral.

Na prática, a relação causa e efeito entre pesos de enlaces e caminhos de roteamento pode ser invertida — operadores de rede configuram pesos de enlaces de modo a obter caminhos de roteamento que cumpram certas metas de engenharia de tráfego [Fortz, 2000; Fortz, 2002]. Por exemplo, suponha que um operador de rede tenha uma estimativa de fluxo do tráfego que entra na rede em cada ponto de ingresso na rede e destinado a cada ponto de saída da rede. Então, o operador poderia querer instituir um roteamento de tráfego do ponto de ingresso até o ponto de saída que minimizasse a utilização máxima em todos os enlaces da rede. Mas, com um algoritmo de roteamento como o OSPF, os principais controles de que o operador dispõe para ajustar o roteamento de fluxos através da rede são os pesos dos enlaces. Assim, para cumprir a meta de minimizar a utilização máxima do enlace, o operador tem de descobrir o conjunto de pesos de enlaces que alcance esse objetivo. Essa é um inversão da relação causa e efeito — o roteamento de fluxos desejado é conhecido e os pesos dos enlaces OSPF têm de ser encontrados de um modo tal que o algoritmo de roteamento OSPF resulte nesse roteamento de fluxos desejado.

primordial da área de backbone é rotear tráfego entre as outras áreas do AS. O backbone sempre contém todos os roteadores de borda de área que estão dentro do AS e pode conter também roteadores que não são de borda. O roteamento interárea dentro do AS requer que o pacote seja primeiramente roteado até um roteador de borda de área (roteamento intra-área), em seguida roteado por meio do backbone até o roteador de borda de área que está na área de destino e, então, roteado até seu destino final.

O OSPF é um protocolo relativamente complexo e aqui, nosso tratamento foi necessariamente breve; [Huitema, 1998; Moy, 1998; RFC 2328] fornecem detalhes adicionais.

4.6.3 Roteamento externo a sistemas autônomos: BGP

Acabamos de aprender como ISPs utilizam RIP e OSPF para determinar caminhos ótimos para pares fonte-destino internos ao mesmo AS. Agora vamos examinar como são determinados caminhos para pares fonte-destino que abrangem vários ASs. A versão 4 do **Protocolo de Roteador de Borda (Border Gateway Protocol — BGP)**, especificada no RFC 4271 (veja também [RFC 4274; RFC 4276]), é o padrão, de facto, para roteamento entre sistemas autônomos na Internet de hoje. Esse protocolo é comumente denominado BGP4 ou simplesmente **BGP**. Na qualidade de um protocolo de roteamento inter-ASs (veja Seção 4.5.3), o BGP provê a cada AS meios de:

1. Obter de ASs vizinhos informações de atingibilidade de sub-redes.
2. Propagar a informação de atingibilidade a todos os roteadores internos ao AS.
3. Determinar rotas ‘boas’ para sub-redes com base na informação de atingibilidade e na política do AS.

O BGP, sobretudo, permite que cada sub-rede anuncie sua existência ao restante da Internet. Uma sub-rede grita “Eu existo e estou aqui” e o BGP garante que todos os ASs da Internet saibam da existência dessa sub-rede e como chegar até ela. Não fosse o BGP, cada sub-rede ficaria isolada — sozinha e desconhecida pelo restante da Internet.

O básico do BGP

O BGP é extremamente complexo; livros inteiros foram dedicados ao assunto e muitas questões ainda não estão claras [Yannuzzi, 2005]. Além disso, mesmo após ler os livros e os RFCs, ainda assim você talvez ache difícil dominar o BGP totalmente sem ter trabalhado com ele na prática durante muitos meses (se não anos) como projetista ou administrador de um ISP de nível superior. Não obstante, como o BGP é um protocolo absolutamente crítico para a Internet — em essência, é o protocolo que agrupa tudo —, precisamos ao menos entender os aspectos rudimentares do seu funcionamento. Começamos descrevendo como o BGP poderia funcionar no contexto do exemplo de rede simples que estudamos anteriormente na Seção 4.5.3; aconselhamos que você leia novamente esse material.

No BGP, pares de roteadores trocam informações de roteamento por conexões TCP semipermanentes usando a porta 179. As conexões TCP semipermanentes para a rede da Figura 4.32 são mostradas na Figura 4.40. Normalmente há uma dessas conexões BGP TCP para cada enlace que conecta diretamente dois roteadores que estão em dois ASs diferentes; assim, na Figura 4.40 há uma conexão TCP entre os roteadores de borda 3a e 1c e uma outra conexão TCP entre os roteadores de borda 1b e 2a. Também há conexões BGP TCP semipermanentes entre roteadores dentro de um AS. Em particular, a Figura 4.40 apresenta uma configuração comum de uma conexão TCP para cada par de roteadores internos a um AS, criando uma malha de conexões TCP dentro de cada AS. Os dois roteadores nas extremidades de cada conexão TCP são denominados **pares BGP**, e a conexão TCP, juntamente com todas as mensagens BGP enviadas pela conexão, é denominada uma **sessão BGP**. Além disso, uma sessão BGP que abrange dois ASs é denominada uma **sessão BGP externa (eBGP)** e uma sessão BGP entre dois roteadores no mesmo AS é denominada uma **sessão BGP interna (iBGP)**. Na Figura 4.40, as sessões eBGP são indicadas pelas linhas de traços longos; as sessões iBGP são representadas pelas linhas de traços mais curtos. Note que as linhas das sessões BGP na Figura 4.40 nem sempre correspondem aos enlaces físicos na Figura 4.32.

O BGP permite que cada AS conheça quais destinos podem ser alcançados via seus ASs vizinhos. Em BGP, os destinos não são hospedeiros, mas **prefixos** cíderizados, sendo que cada prefixo representa uma sub-rede ou um conjunto de sub-redes. Assim, por exemplo, suponha que haja quatro sub-redes conectadas ao AS2: 138.16.64/24, 138.16.65/24, 138.16.66/24 e 138.16.67/24. O AS2 então poderia agrregar os prefixos dessas quatro sub-redes e utilizar o BGP para anunciar ao AS1 o prefixo único 138.16.64/22. Como outro exemplo, suponha que somente as primeiras três dessas quatro sub-redes estão em AS2 e que a quarta sub-rede, 138.16.67/24, está em AS3. Então, como descrito no quadro Princípios na Prática na Seção 4.4.2, os roteadores usam compatibilização de prefixo mais longo para repassar datagramas. Portanto, o AS3 poderia anunciar a AS1 o prefixo mais específico 138.16.67/24 e o AS2 ainda poderia anunciar ao AS1 o prefixo agregado 138.16.64/22.

Agora vamos examinar como o BGP distribuiria informações sobre a atingibilidade de prefixos pelas sessões BGP mostradas na Figura 4.40. Como seria de esperar, usando a sessão eBGP entre os roteadores de borda 3a e 1c, AS3 envia a AS1 a lista de prefixos que podem ser alcançados a partir de AS3; e AS1 envia a AS3 a lista de prefixos que podem ser alcançados de AS1. De modo semelhante, AS1 e AS2 trocam informações de atingibilidade

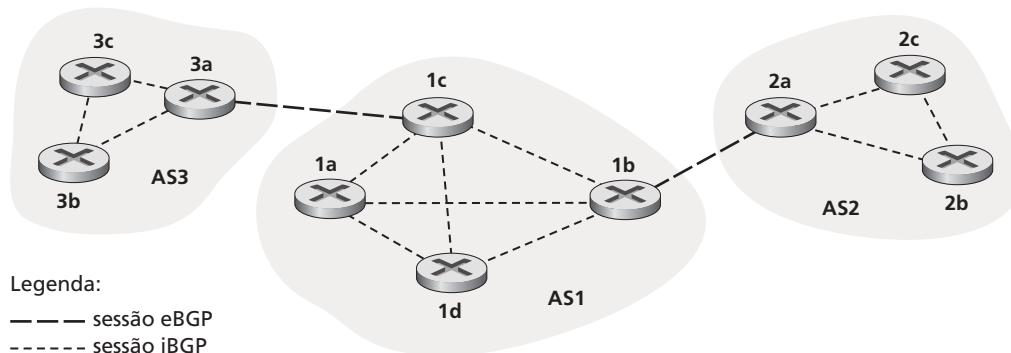


Figura 4.40 Sessões eBGP e iBGP

de prefixos por meio de seus roteadores de borda 1b e 2a. E, como também era de esperar, quando um roteador de borda (em qualquer AS) recebe prefixos conhecidos pelo BGP, ele usa suas sessões iBGP para distribuir os prefixos aos outros roteadores no AS. Assim, todos os roteadores no AS1 se informarão sobre os prefixos de AS3, incluindo o roteador de borda 1b. O roteador de borda 1b (em AS1) pode reanunciar a AS2 os prefixos de AS3. Quando um roteador (seja ou não de borda) fica sabendo de um novo prefixo, cria um registro para esse prefixo em sua tabela de repasse, como descrito na Seção 4.5.3.

Atributos de caminho e roteadores BGP

Agora que já temos um conhecimento preliminar do BGP, vamos um pouco mais fundo (e, mesmo assim, estaremos varrendo para baixo do tapete alguns dos detalhes menos importantes!). No BGP, um sistema autônomo é identificado por seu **número de sistema autônomo** (ASN) globalmente exclusivo [RFC 1930]. (Tecnicamente, nem todo AS tem um ASN. Em particular, um *stub AS* que carregue somente tráfego do qual é uma fonte ou um destino normalmente não terá um ASN; ignoramos essa tecnicidade em nossa discussão para que os detalhes não nos impeçam de observar melhor o quadro geral.) Números de ASs, tal como endereços IP, são designados por entidades regionais ICANN de registro [ICANN, 2009].

Quando um roteador anuncia um prefixo para uma sessão BGP, inclui vários **atributos BGP** juntamente com o prefixo. Na terminologia do BGP, um prefixo, juntamente com seus atributos, é denominado uma **rota**. Assim, pares BGP anunciam rotas uns aos outros. Dois dos atributos mais importantes são AS-PATH e NEXT-HOP:

■ **AS-PATH.** Este atributo contém os ASs pelos quais passou o anúncio para o prefixo. Quando um prefixo é passado para dentro de um AS, o AS adiciona seu ASN ao atributo AS-PATH. Por exemplo, considere a Figura 4.40 e suponha que o prefixo 138.16.64/24 seja anunciado primeiramente de AS2 para AS1; se AS1 então anunciar o prefixo a AS3, o AS-PATH seria AS2 AS1. Roteadores usam o atributo AS-PATH para detectar e evitar looping de anúncios; especificamente, se um roteador perceber que seu AS está contido na lista de caminhos, rejeitará o anúncio. Como discutiremos em breve, roteadores também usam o atributo AS-PATH ao escolher entre vários caminhos para o mesmo prefixo.

■ Prover o enlace crítico entre os protocolos de roteamento inter-AS e intra-AS, o NEXT-HOP possui uma útil mas importante utilidade. O *NEXT-HOP* é a interface do roteador que inicia o *as-path*. Para compreender mais esse atributo, vamos, novamente, nos referir à Figura 4.40. Considere o que acontece quando o roteador de borda 3a em AS3 anuncia uma rota a um roteador de borda 1c em AS1 utilizando o BGP. A rota inclui o prefixo anunciado, que chamaremos *x*, e um AS-PATH para o prefixo. Esse anúncio também inclui o NEXT-HOP, que é o endereço IP da interface 3a do roteador que conduz a 1c. (Lembre-se de que um roteador possui diversos endereços IP, um para cada uma das interfaces.) Considere agora o que acontece quando o roteador 1d é informado sobre essa rota pelo iBGP. Após descobrir essa rota para *x*, o roteador 1d pode querer encaminhar pacotes para *x* ao longo da rota, ou seja, o roteador 1d pode querer incluir a entrada (*x*, *l*) em sua tabela de repasse, na qual *l* representa sua interface que inicia o caminho de menor custo partindo de 1d em direção ao roteador de borda 1c. Para determinar *l*, 1d provê o endereço IP no NEXT-HOP a seu módulo de roteamento intra-AS. Observe que o algoritmo de roteamento intra-AS determinou o enlace de menor custo entre 1c e 3a. Desse caminho de menor custo de 1d à sub-rede 1c-3a, 1d determina sua interface do roteador *l* que inicia esse caminho e, então, adiciona a entrada (*x*, *l*) à sua tabela de repasse. Até que enfim! Em resumo, o atributo AS-PATH é usado por roteadores para configurar suas tabelas de repasse adequadamente.

A Figura 4.41 ilustra outra situação em que o AS-PATH é requisitado. Na figura, o AS1 e o AS2 estão conectados por dois enlaces interconectados. Um roteador AS1 poderia descobrir duas rotas diferentes para o mesmo prefixo *x*. Essas duas rotas poderiam ter o mesmo AS-PATH para *x*, mas poderiam ter diferentes valores NEXT-HOP correspondentes aos enlaces interconectados. Utilizando os valores AS-PATH e o algoritmo de roteamento intra-AS, o roteador pode determinar o custo do caminho para cada enlace interconectado e, então, aplicar o roteamento batata quente (consulte Seção 4.5.3) para determinar a interface apropriada.

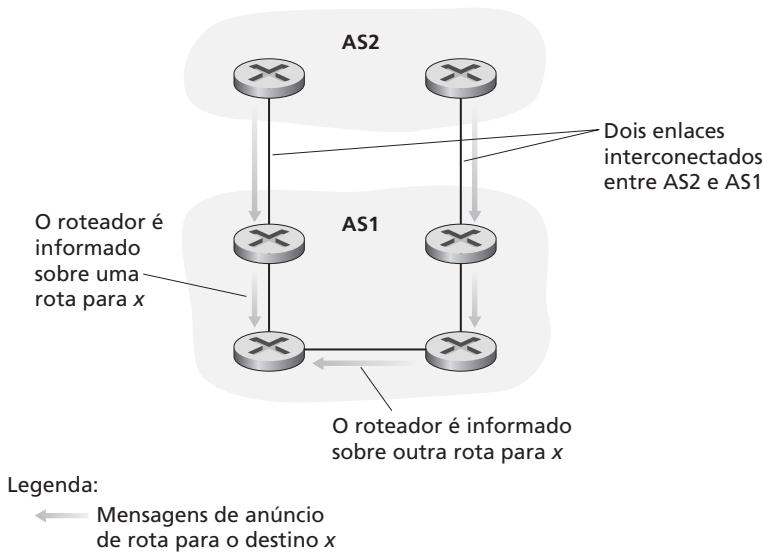


Figura 4.41 Atributos NEXT-HOP em anúncios servem para determinar qual enlace interconectado utilizar

O BGP também inclui atributos que permitem que roteadores designem a métrica de sua preferência às rotas, e um atributo que indica como o prefixo foi inserido no BGP na origem. Para uma discussão completa de atributos de rota, veja [Griffin, 2009; Stewart, 1999; Halabi, 2000; Feamster, 2004; RFC 4271].

Quando um roteador de borda recebe um anúncio de roteador, ele utiliza sua **política de importação** para decidir se aceita ou filtra a rota e se estabelece certos atributos, tal como a métrica de preferência do roteador. A política de importação pode filtrar uma rota porque o AS pode não querer enviar tráfego por um dos ASs no AS-PATH. O roteador de borda também pode filtrar uma rota porque já conhece uma rota preferencial para o mesmo prefixo.

Seleção de rota do BGP

Como descrito anteriormente nesta seção, o BGP usa eBGP e iBGP para distribuir rotas a todos os roteadores dentro de ASs. Com essa distribuição, um roteador pode conhecer mais do que uma rota para qualquer prefixo determinado, caso em que deve selecionar uma das rotas possíveis. O dado usado por esse processo de seleção de rota é o conjunto de todas as rotas que o roteador conhece e aceitou. Se houver duas ou mais rotas para o mesmo prefixo, então o BGP invoca sequencialmente as seguintes regras de eliminação até sobrar apenas uma rota:

Rotas recebem, como um de seus atributos, um valor de preferência local. A preferência local de uma rota pode ter sido estabelecida pelo roteador ou pode ter sido aprendida por um outro roteador no mesmo AS. Essa é uma decisão política que fica a cargo do administrador da rede do AS. (Mais adiante discutiremos questões de política do BGP.) São selecionadas as rotas que têm os valores de preferência local mais altos.

No caso das outras rotas remanescentes (todas com o mesmo valor de preferência local), é selecionada a rota que tenha o AS-PATH mais curto. Se essa fosse a única regra de seleção de rotas, então o BGP estaria usando um algoritmo DV para determinação de caminho no qual a métrica da distância utiliza o número de saltos de AS em vez do número de saltos de roteadores.

Dentre as rotas remanescentes (todas com o mesmo valor de preferência local e com o mesmo comprimento de AS-PATH) é selecionada a rota que tenha o roteador NEXT-HOP mais próximo. Nesse caso, mais próximo significa o roteador que tenha o menor custo de caminho de menor custo determinado pelo algoritmo intra-AS. Esse processo é normalmente denominado roteamento da batata quente.

Se ainda restar mais de uma rota, o roteador usa identificadores BGP para selecionar a rota; veja [Stewart, 1999].

As regras de eliminação são ainda mais complicadas do que descrevemos aqui. Para evitar pesadelos com o BGP, é melhor aprender suas regras de seleção em pequenas doses!

Política de roteamento

Vamos ilustrar alguns dos conceitos básicos do roteamento BGP com um exemplo simples. A Figura 4.42 mostra seis sistemas autônomos interconectados: A, B, C, W, X, e Y. É importante notar que A, B, C, W, X, e Y são ASs, e não roteadores. Vamos admitir que os sistemas autônomos W, X, e Y são redes stub e que A, B, e C são redes provedoras de backbone. Admitiremos também que A, B e C, todos conectados uns aos outros, fornecem informação completa sobre o BGP a suas redes cliente. Todo o tráfego que entrar em uma **rede stub** deve ser destinado a essa rede, e todo o tráfego que sair da rede stub deve ter sido originado naquela rede. W e Y são claramente redes stub. X é uma **rede stub com múltiplas interconexões**, visto que está conectado ao resto da rede por meio de dois provedores diferentes (um cenário que está se tornando cada vez mais comum na prática). Todavia, tal como W e Y, o próprio X deve ser a origem/destino de todo o tráfego que entra/sai de X. Mas, como esse comportamento da rede stub será implementado e imposto? Como X será impedido de repassar tráfego entre B e C? Isso pode ser conseguido facilmente controlando o modo como as rotas BGP são anunciadas. Em particular, X funcionará como uma rede stub se anunciar (a seus vizinhos B e C) que não há nenhum caminho para quaisquer outros destinos a não ser ele mesmo. Isto é, mesmo que X conheça um caminho, digamos, XCY, que chegue até a rede Y, ele *não* anunciará esse caminho a B. Uma vez que B não fica sabendo que X tem um caminho para Y, B nunca repassaria tráfego destinado a Y (ou a C) via X. Esse exemplo simples ilustra como uma política seletiva de anúncio de rota pode ser usada para implementar relacionamentos de roteamento cliente/provedor.

Em seguida, vamos focalizar uma rede provedora, digamos AS B. Suponha que B ficasse sabendo (por A), que A tem um caminho AW para W. Assim, B pode instalar a rota BAW em sua base de informações de roteamento. É claro que B também quer anunciar o caminho BAW a seu cliente, X, de modo que X saiba que pode rotear para W via B. Mas, B deveria anunciar o caminho BAW a C? Se o fizer, então C poderia rotear tráfego para W via CBAW. Se A, B e C forem todos provedores de backbone, então B poderia sentir-se no direito de achar que não deveria ter de suportar a carga (e o custo!) de carregar tráfego em trânsito entre A e B. B poderia sentir-se no direito de achar que é de A e C o trabalho (e o custo) de garantir que C possa rotear de/para clientes de A via uma conexão direta entre A e C. Atualmente não existe nenhum padrão oficial que determine como ISPs de backbone devem rotear entre si. Todavia, os ISPs comerciais adotam uma regra prática que diz que qualquer tráfego que esteja fluindo por uma rede de backbone de um ISP deve ter ou uma origem ou um destino (ou ambos) em uma rede que seja cliente daquele ISP; caso contrário, o tráfego estaria pegando uma carona gratuita na rede do ISP. Acordos individuais de parceria (*peering*) (para reger questões como as levantadas acima) normalmente são negociados entre pares de ISPs e, em geral, são confidenciais; [Huston, 1999a] provê uma discussão interessante sobre acordos de parceria. Se quiser uma descrição detalhada sobre como a política de roteamento reflete os relacionamentos comerciais entre ISPs, veja [Gao, 2001; Dimitriopoulos, 2007]. Para uma abordagem recente sobre políticas de roteamento BGP, de um ponto de vista do ISP, consulte [Caesar, 2005].

Como já observamos, o BGP é um padrão *de facto* para roteamento inter-AS na Internet pública. Para ver o conteúdo de várias tabelas de roteamento BGP (grandes!) extraídas de roteadores pertencentes a ISPs de nível 1, consulte <http://www.routeviews.org>. Tabelas de roteamento BGP geralmente contêm dezenas de milhares de

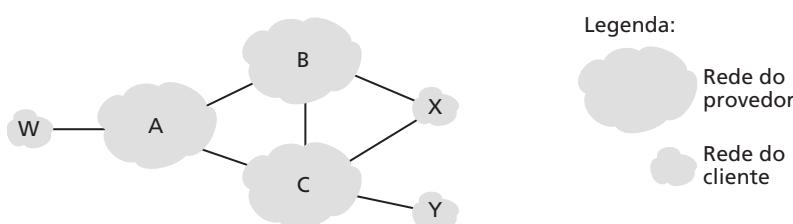


Figura 4.42 Um cenário BGP simples



prefixos e atributos correspondentes. Estatísticas sobre tamanho e características de tabelas de roteamento BGP são apresentadas em [Huston, 2001; Meng, 2005; Potaroo, 2009].

Com isso concluímos nossa breve introdução ao BGP. Entender esse protocolo é importante porque ele desempenha um papel central na Internet. Aconselhamos você a consultar as referências [Griffin, 2002; Stewart, 1999; Labovitz, 1997; Halabi, 2000; Huitema, 1998; Gao, 2001; Feamster, 2004; Caesar, 2005; Li, 2007] para aprender mais sobre BGP.

4.7 Roteamento broadcast e multicast

Até este ponto do capítulo, focalizamos protocolos de roteamento que suportam comunicação *unicast* (isto é, ponto a ponto) na qual um único nó de fonte envia um pacote a um único nó de destino. Nesta seção, voltaremos nossa atenção a protocolos de roteamento *broadcast* e *multicast*. No **roteamento broadcast** a camada de rede provê um serviço de entrega de um pacote enviado de um nó de fonte a todos os outros nós da rede; o **roteamento multicast** habilita um único nó de fonte a enviar uma cópia de um pacote a um subconjunto de nós das outras redes. Na Seção 4.7.1 consideraremos algoritmos de roteamento *broadcast* e sua incorporação em protocolos de roteamento. Examinaremos roteamento *multicast* na Seção 4.7.2.



Princípios na prática

Por que há diferentes protocolos de roteamento inter-AS e intra-AS?

Agora que já examinamos os detalhes de protocolos de roteamento inter-AS e intra-AS específicos utilizados pela Internet de hoje, vamos concluir considerando a questão talvez mais fundamental que, antes de mais nada, poderíamos levantar sobre esses protocolos (esperamos que você tenha estado preocupado com isso o tempo todo e que não tenha deixado de enxergar o quadro geral por causa dos detalhes!). Por que são usados diferentes protocolos de roteamento inter-AS e intra-AS?

A resposta a essa pergunta expõe o âmago da diferença entre os objetivos do roteamento interno a um AS e entre ASs:

- Política. Entre ASs, as questões políticas dominam. Pode até ser importante que o tráfego que se origina em um dado AS não possa passar através de um outro AS específico. De maneira semelhante, um dado AS pode muito bem querer controlar o tráfego em trânsito que ele carrega entre outros ASs. Vimos que o BGP carrega atributos de caminho e oferece distribuição controlada de informação de roteamento, de modo que essas decisões de roteamento baseadas em políticas possam ser tomadas. Dentro de um AS, tudo está nominalmente sob o mesmo controle administrativo. Assim, as questões de políticas de roteamento desempenham um papel bem menos importante na escolha de rotas dentro do AS.
- Escalabilidade. A escalabilidade de um algoritmo de roteamento e de suas estruturas de dados para manipular o roteamento para/entre grandes números de redes é uma questão fundamental para o roteamento inter-AS. Dentro de um AS, a escalabilidade é uma preocupação menor. Isso porque, se um único domínio administrativo ficar muito grande, é sempre possível dividi-lo em dois ASs e realizar roteamento inter-AS entre esses dois novos ASs. (Lembre-se de que o OSPF permite que essa hierarquia seja construída dividindo um AS em áreas.)
- Desempenho. Dado que o roteamento inter-AS é bastante orientado pelas políticas, a qualidade (por exemplo, o desempenho) dos roteadores usados é frequentemente uma preocupação secundária (isto é, uma rota mais longa ou de custo mais alto que satisfaça certos critérios políticos pode muito bem prevalecer sobre uma rota que é mais curta, mas que não satisfaz esses critérios). Na verdade, vimos que entre ASs não há nem mesmo a ideia de custo associado às rotas (exceto a contagem de saltos). Dentro de um AS individual, contudo, essas preocupações com as políticas podem ser ignoradas, permitindo que o roteamento se concentre mais no nível de desempenho atingido em uma rota.

4.7.1 Algoritmos de roteamento broadcast

Talvez o modo mais direto de conseguir comunicação *broadcast* é o nó remetente enviar uma cópia separada do pacote para cada destino, como mostra a Figura 4.43(a). Dados N nós de destino, o nó da fonte simplesmente faz N cópias do pacote, endereça cada cópia a um destino diferente e então transmite N cópias aos N destinos usando roteamento *unicast*. Essa abordagem ***unicast de N caminhos*** da transmissão *broadcast* é simples — não é preciso nenhum novo protocolo de roteamento de camada de rede, nem duplicação de pacotes, nem funcionalidade de repasse. Porém, essa abordagem tem várias desvantagens. A primeira desvantagem é sua ineficiência. Se o nó da fonte estiver conectado ao resto da rede por um único enlace, então N cópias separadas do (mesmo) pacote transitarão por esse único enlace. Evidentemente seria mais eficiente enviar somente uma única cópia de um pacote até esse primeiro salto e então fazer com que o nó na outra extremidade do primeiro salto produzisse e transmitisse quaisquer cópias adicionais necessárias. Isto é, seria mais eficiente que os próprios nós da rede (em vez de apenas o nó da fonte) criassem e duplicassem cópias de um pacote. Por exemplo, na Figura 4.43(b), somente uma única cópia de um pacote transita pelo enlace R1-R2. Aquele pacote então é duplicado em R2, e uma única cópia será enviada pelos enlaces R2-R3 e R2-R4.

As desvantagens adicionais do *unicast de N caminhos* talvez sejam mais sutis, mas nem por isso menos importantes. Uma premissa implícita desse tipo de roteamento é que o remetente conheça os destinatários do *broadcast* e seus endereços. Mas, como essa informação é obtida? Muito provavelmente seriam exigidos mecanismos de protocolo adicionais (tais como associação ao *broadcast* ou protocolo de registro de destino), o que adicionaria mais sobrecarga e, o que é importante, complexidade adicional a um protocolo que, inicialmente parecia bastante simples. Uma desvantagem final do *unicast de N caminhos* está relacionada aos propósitos para os quais o *broadcast* seria utilizado. Na Seção 4.5 aprendemos que protocolos de roteamento de estado de enlace usam *broadcast* para propagar informações de estado de enlace que são usadas para calcular rotas *unicast*. É claro que, em situações em que o *broadcast* é usado para criar e atualizar rotas *unicast*, seria imprudente (no mínimo!) confiar em infraestrutura de roteamento *unicast* para implantar *broadcast*.

Dadas as diversas desvantagens do *broadcast de N caminhos*, são claramente de interesse abordagens nas quais os próprios nós da rede desempenham um papel ativo na duplicação de pacotes, no repasse de pacotes e no cálculo de rotas de *broadcast*. A seguir, examinaremos diversas dessas abordagens e, mais uma vez, adotaremos a notação de grafo apresentada na Seção 4.5. Modelaremos novamente a rede como um grafo $G = (N, E)$, onde N é um conjunto de nós e uma coleção E de arestas, sendo que cada aresta é um par de nós de N . Não seremos muito exigentes quanto à notação e adotaremos N para denotar ambos os conjuntos de nós, e a cardinalidade ($|N|$) ou o tamanho daquele conjunto, quando não houver possibilidade de confusão.

Inundação não controlada

A técnica mais óbvia para conseguir *broadcast* é uma abordagem de **inundação** na qual o nó da fonte envia uma cópia do pacote a todos os seus vizinhos. Quando um nó recebe um pacote *broadcast*, ele duplica o pacote e o repassa a todos os seus vizinhos (exceto ao vizinho do qual recebeu o pacote). É claro que, se o grafo for

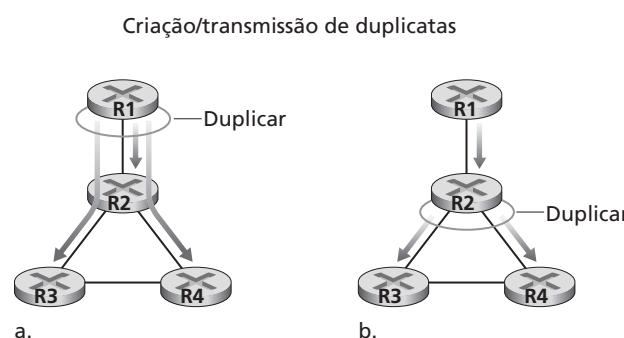


Figura 4.43 Duplicação de fonte *versus* duplicação dentro da rede

conectado, esse esquema finalmente entregará uma cópia do pacote *broadcast* a todos os nós no grafo. Embora esse esquema seja simples e elegante, tem uma falha fatal (antes de continuar, tente imaginar qual seria essa falha): se o grafo tiver ciclos, então uma ou mais cópias de cada pacote de *broadcast* permanecerão em ciclo indefinidamente. Por exemplo, na Figura 4.43, R2 inundará R3, R3 inundará R4, R4 inundará R2 e R2 inundará (novamente!) R3 e assim por diante. Esse cenário simples resulta no ciclo sem fim de dois pacotes *broadcast*, um em sentido horário e um em sentido anti-horário. Mas pode haver uma falha fatal ainda mais catastrófica: quando um nó estiver conectado a mais de dois outros nós, criará e repassará várias cópias do pacote *broadcast*, cada uma das quais criará várias cópias de si mesmas (em outros nós com mais de dois vizinhos) e assim por diante. Essa **tempestade de broadcast**, gerada pela infundável multiplicação de pacotes *broadcast*, finalmente resultaria na criação de uma quantidade tão grande de pacotes *broadcast* que a rede ficaria inutilizada. (Consulte nos exercícios de fixação ao final do capítulo um problema que analisa a velocidade com que essa tempestade de *broadcast* cresce.)

Inundação controlada

A chave para evitar uma tempestade de *broadcast* é um nó escolher sensatamente quando repassa um pacote (por exemplo, se já tiver recebido e repassado uma cópia anterior do pacote) e quando não repassa um pacote. Na prática, há vários modos de fazer isso.

Na inundação controlada com número de sequência, um nó de fonte coloca seu endereço (ou outro identificador exclusivo), bem como um número de sequência de ***broadcast*** em um pacote *broadcast* e então envia o pacote a todos os seus vizinhos. Cada nó mantém uma lista de endereços de fonte e números de sequência para cada pacote *broadcast* que já recebeu, duplicou e repassou. Quando um nó recebe um pacote *broadcast*, primeiramente verifica se o pacote está nessa lista. Se estiver, o pacote é descartado; se não estiver, é duplicado e repassado para todos os vizinhos do nó (exceto para o nó de quem o pacote acabou de ser recebido). O protocolo Gnutella, discutido no Capítulo 2, usa inundação controlada com número de sequência para fazer transmissão *broadcast* de consultas em sua rede de sobreposição. (Em Gnutella, a duplicação e transmissão de mensagens é realizada na camada de aplicação, e não na camada de rede.)

Uma segunda abordagem da inundação controlada é conhecida como **repasse pelo caminho inverso** (*reverse path forwarding* — RPF) [Dalal, 1978], às vezes também denominada *broadcast* pelo caminho inverso (*reverse path broadcast* — RPB). A ideia por trás do repasse pelo caminho inverso é simples, mas elegante. Quando um roteador recebe um pacote *broadcast* com um dado endereço de fonte, ele transmite o pacote para todos os seus enlaces de saída (exceto para aquele do qual o pacote foi recebido) somente se o pacote chegou pelo enlace que está em seu próprio caminho *unicast* mais curto de volta ao remetente. Caso contrário, o roteador simplesmente descarta o pacote que está entrando sem repassá-lo para nenhum de seus enlaces de saída. Esse pacote pode ser descartado porque o roteador sabe que receberá, ou já recebeu, uma cópia desse pacote no enlace que está em seu próprio caminho mais curto de volta ao remetente. (É provável que você esteja querendo se convencer de que isso, de fato, acontecerá, e que não ocorrerão nem looping nem tempestades de *broadcast*.) Note que o RPF não usa roteamento *unicast* para entregar um pacote a um destino, nem exige que um roteador conheça o caminho mais curto completo entre ele mesmo e a fonte. O RPF precisa conhecer apenas o próximo vizinho em seu caminho *unicast* mais curto até o remetente; usa a identidade desse vizinho apenas para determinar se repassa ou não repassa um pacote *broadcast* recebido.

A Figura 4.44 ilustra o RPF. Suponha que os enlaces com as linhas mais grossas representem os caminhos de menor custo desde os destinatários até a fonte (A). O nó A inicialmente faz a transmissão *broadcast* de um pacote da fonte A até os nós C e B. O nó B repassará o pacote da fonte A que recebeu de A (uma vez que A está em seu caminho de menor custo até A) para C e D. B ignorará (descartará sem repassar) quaisquer pacotes da fonte A que receba de quaisquer outros nós (por exemplo, dos roteadores C ou D). Vamos considerar agora o nó C, que receberá um pacote da fonte A diretamente de A e também de B. Uma vez que B não está no caminho mais curto de C de retorno a A, C ignorará quaisquer pacotes da fonte A que receba de B. Por outro lado, quando C receber um pacote da fonte A diretamente de A, ele o repassará aos nós B, E e F.

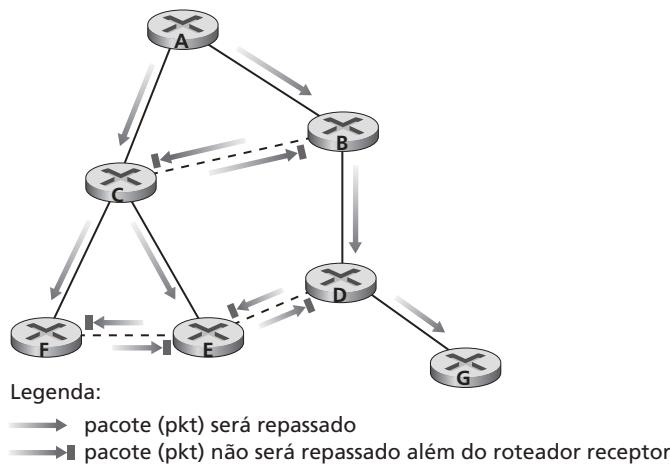


Figura 4.44 Repasse pelo caminho inverso

Broadcast por spanning tree

Conquanto a inundação controlada com números de sequência e o RPF impeçam tempestades de broadcast, não evitam completamente a transmissão de pacotes broadcast redundantes. Por exemplo, na Figura 4.45, os nós *B*, *C*, *D* e *F* recebem ou um ou dois pacotes redundantes. Idealmente, cada nó deveria receber somente uma cópia do pacote broadcast.

Examinando a árvore formada pelos nós conectados por linhas grossas na Figura 4.45(a), podemos ver que, se os pacotes broadcast fossem repassados por enlaces dessa árvore, cada nó da rede receberia exatamente uma cópia do pacote broadcast — exatamente a solução que procurávamos! Esse é um exemplo de **spanning tree** — uma árvore que contém todos os nós, sem exceção, em um grafo. Mais formalmente, uma spanning tree de um grafo $G = (N, E)$ é um grafo $G' = (N, E')$ tal que E' é um subconjunto de E , G' é conectado, G' não contém nenhum ciclo e G' contém todos os nós originais em G . Se cada enlace tiver um custo associado e o custo de uma árvore for a soma dos custos dos enlaces, então uma árvore cujo custo seja o mínimo entre todas as spanning trees do gráfico é denominada uma **spanning tree mínima** (o que não é nenhuma surpresa).

Assim, uma outra abordagem para o fornecimento de broadcast é os nós da rede construírem uma spanning tree, em primeiro lugar. Quando um nó de fonte quiser enviar um pacote broadcast, enviará o pacote por todos

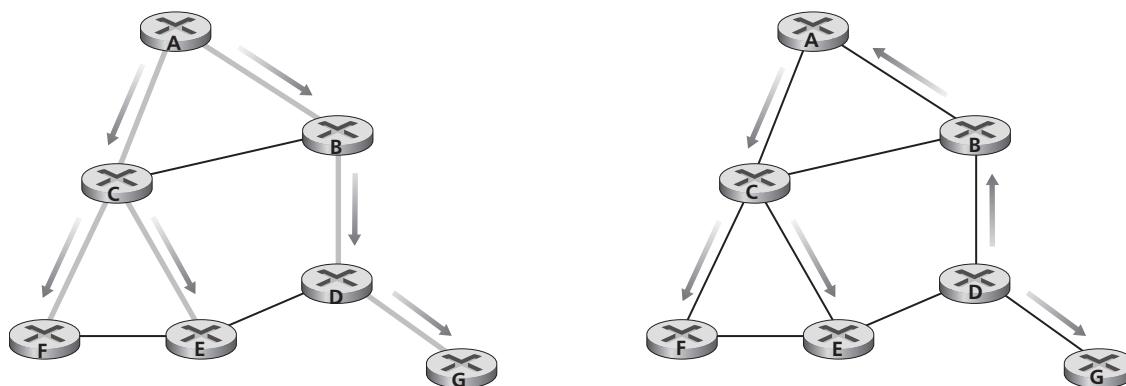


Figura 4.45 Broadcast por uma spanning tree

os enlaces incidentes que pertençam à *spanning tree*. Um nó que receba um pacote *broadcast* então o repassa a todos os seus vizinhos na *spanning tree* (exceto para o vizinho do qual recebeu o pacote). A *spanning tree* não apenas elimina pacotes *broadcast* redundantes mas, uma vez instalada, pode ser usada por qualquer nó para iniciar um *broadcast*, como mostram as Figuras 4.45(a) e 4.45(b). Note que um nó não precisa estar ciente da árvore inteira; ele precisa simplesmente saber qual de seus vizinhos em G são vizinhos que pertencem à *spanning tree*.

A principal complexidade associada com a abordagem de *spanning tree* é sua criação e manutenção. Foram desenvolvidos numerosos algoritmos distribuídos de *spanning tree* [Gallager, 1983; Gartner, 2003]. Aqui, consideramos apenas um algoritmo simples. Na **abordagem de nó central** da construção de uma *spanning tree*, é definido um nó central (também conhecido como **ponto de encontro** ou **núcleo**). Os nós então transmitem em *unicast* mensagens de adesão à árvore endereçadas ao nó central. Uma mensagem de adesão à árvore é repassada usando roteamento *unicast* em direção ao centro até que ela chegue a um roteador que já pertence à árvore *spanning* ou até que chegue ao centro. Em qualquer um dos casos, o caminho que a mensagem de adesão à árvore seguiu define o ramo da *spanning tree* entre o roteador de porta que enviou a mensagem de adesão à árvore e o centro. Esse novo caminho pode ser visto como um *enxerto* à *spanning tree* existente.

A Figura 4.46 ilustra a construção de uma *spanning tree* com centro. Suponha que o nó E seja selecionado como centro da árvore. Suponha que o nó F primeiramente se junte à árvore e repasse uma mensagem de adesão à árvore a E. O único enlace EF se torna a *spanning tree* inicial. O nó B então se junta à *spanning tree* enviando a E sua mensagem de adesão à árvore. Suponha que a rota do caminho *unicast* de E para B seja por D. Nesse caso, a mensagem de adesão à árvore resulta no *enxerto* do caminho BDE à *spanning tree*. Em seguida, o nó A se junta ao grupo crescente repassando sua mensagem de adesão à árvore em direção a E. Se o caminho *unicast* de A a E passar por B, então, uma vez que B já se juntou à *spanning tree*, a chegada da mensagem de adesão à árvore de A em B resultará no *enxerto* imediato do enlace AB à *spanning tree*. Em seguida, o nó C se junta à *spanning tree* repassando sua mensagem de adesão à árvore diretamente a E. Finalmente, como o roteamento *unicast* de G para E tem de passar pelo nó D, quando G enviar sua mensagem de adesão à árvore a E, o enlace GD será *enxertado* à *spanning tree* no nó D.

Algoritmos de broadcast na prática

Protocolos *broadcast* são usados na prática nas camadas de aplicação e de rede. Gnutella [Gnutella, 2009] usa *broadcast* no nível de aplicação para fazer transmissão *broadcast* de consultas de conteúdo entre pares Gnutella. Nesse caso, um enlace entre dois processos pares distribuídos no nível de aplicação na rede Gnutella é, na verdade, uma conexão TCP. O Gnutella usa uma forma de inundação controlada com número de sequência na qual um identificador de 16 bits e um descritor de carga útil de 16 bits (que identifica o tipo de mensagem Gnutella)

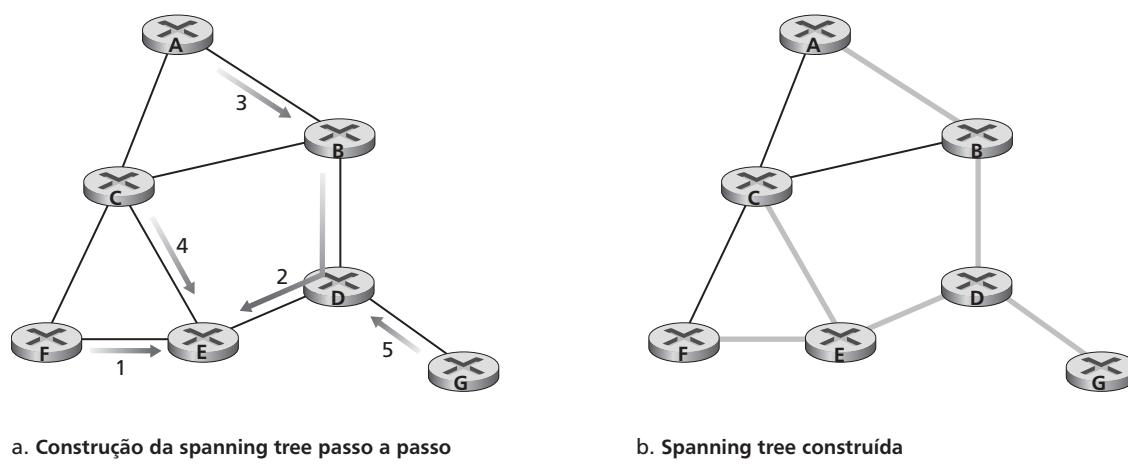


Figura 4.46 Construção de uma *spanning tree* com centro

são usados para detectar se uma consulta *broadcast* recebida já foi recebida, duplicada e repassada anteriormente. O Gnutella usa um campo de tempo de vida (TTL) para limitar o número de saltos pelos quais uma consulta será repassada. Quando um processo Gnutella recebe e duplica uma consulta, ele decrementa o campo de TTL antes de retransmiti-la. Assim, uma consulta Gnutella em *broadcast* alcançará apenas pares que estão dentro de um dado número (o valor inicial do TTL) de saltos de nível de aplicação desde o iniciador da consulta. Por isso, o mecanismo de consultas por inundação do Gnutella às vezes é denominado *inundação de escopo limitado*.

Uma forma de inundação controlada com número de sequência também é usada para fazer a transmissão *broadcast* de anúncios de estado de enlace (*link-state advertisements* — LSAs) no algoritmo de roteamento OSPF [RFC 2328; Perlman, 1999] e no algoritmo de roteamento Sistema Intermediário a Sistema Intermediário (*Intermediate-System-to-Intermediate-System* — IS-IS) [RFC 1142; Perlman, 1999]. O OSPF usa um número de sequência de 32 bits, bem como um campo de idade de 16 bits para identificar anúncios de estado de enlace (LSAs). Lembre-se de que um nó OSPF faz *broadcast* periódico de LSAs para os enlaces ligados a ele quando o custo de enlace até um vizinho muda ou quando um enlace ativa ou desativa. Números de sequência LSA são usados para detectar LSAs duplicados, mas também cumprem uma segunda função importante no OSPF. Com inundação, é possível que um LSA gerado pela fonte no tempo t chegue *após* um LSA mais novo que foi gerado pela mesma fonte no tempo $t + \delta$. Os números de sequência usados pelo nó da fonte permitem que um LSA mais velho seja distinguido de um LSA mais novo. O campo de idade cumpre uma finalidade semelhante à de um valor TTL. O valor inicial do campo de idade é estabelecido em zero e é incrementado a cada salto à medida que este é retransmitido, e também é incrementado enquanto fica na memória de um roteador esperando para ser enviado. Embora nossa descrição do algoritmo de inundação LSA tenha sido apenas breve, observamos que, na verdade, projetar protocolos de *broadcast* LSA pode ser um negócio muito delicado. [RFC 789; Perlman, 1999] descrevem um incidente no qual LSAs transmitidos incorretamente por dois roteadores defeituosos fizeram com que uma versão antiga do algoritmo de inundação LSA derrubasse a ARPAnet inteira!

4.7.2 Multicast

Vimos, na seção anterior, que, com serviço *broadcast*, pacotes são entregues a cada um e a todos os nós em uma rede. Nessa seção, voltamos nossa atenção para o serviço **multicast**, no qual um pacote *multicast* é entregue a apenas um *subgrupo* de nós de rede. Uma série de aplicações emergentes de rede requer a entrega de pacotes de um ou mais remetentes a um grupo de destinatários. Entre essas aplicações estão a transferência de dados em grandes volumes (por exemplo, a transferência de uma atualização de software do desenvolvedor do software para os usuários que necessitam da atualização), a recepção de aplicações de taxa constante (por exemplo, transferência de áudio, vídeo e texto de uma palestra ao vivo para um conjunto de participantes distribuídos), as aplicações de dados compartilhados (por exemplo, uma videoconferência compartilhada por muitos participantes distribuídos), a alimentação de dados (por exemplo, cotação de ações), a atualização de arquivos de memória intermediária e os jogos interativos (por exemplo, ambientes virtuais interativos distribuídos ou jogos multiusuários).

Na comunicação *multicast*, enfrentamos imediatamente dois problemas — como identificar os destinatários de um pacote *multicast* e como endereçar um pacote enviado a um desses destinatários. No caso da comunicação *unicast*, o endereço IP do destinatário (receptor) é levado em cada datagrama IP *unicast* e identifica o único destinatário; no caso do *multicast*, temos vários destinatários. Tem sentido que cada pacote *multicast* carregue os endereços IP de todos os vários destinatários? Embora essa abordagem possa ser funcional com um número pequeno de destinatários, não tem boa escalabilidade para o caso de centenas de milhares de destinatários; a quantidade de informações de endereçamento no datagrama sobrepujaria a quantidade de dados que é realmente carregada no campo de carga útil do pacote. A identificação explícita dos destinatários pelo remetente também exige que o remetente conheça as identidades e os endereços de todos os destinatários. Veremos em breve que há casos em que essa exigência pode ser indesejável.

Por essas razões, na arquitetura da Internet (e em outras arquiteturas de rede como ATM [Black, 1995]), um pacote *multicast* é endereçado usando **endereço indireto**, isto é, um único identificador é utilizado para o grupo de destinatários e uma cópia do pacote que é endereçada ao grupo usando esse único identificador é entregue a todos os destinatários *multicast* associados ao grupo em questão. Na Internet, o identificador único que representa

um grupo de destinatários é um endereço IP *multicast* classe D. O grupo de destinatários associados a um endereço classe D é denominado **grupo multicast**. A abstração do grupo *multicast* é ilustrada na Figura 4.47. Nessa figura, quatro hospedeiros (no tom mais claro) estão associados ao endereço de grupo *multicast* 226.17.30.197 e receberão todos os datagramas endereçados a esse endereço *multicast*. A dificuldade que ainda temos de enfrentar é o fato de que cada hospedeiro tem um endereço *unicast* exclusivo que é completamente independente do endereço do grupo *multicast* do qual ele está participando.

Embora seja simples, a abstração *multicast* provoca uma série de perguntas. Como um grupo começa e como termina? Como é escolhido o endereço do grupo? Como novos hospedeiros são incorporados ao grupo (seja como remetentes, seja como destinatários)? Qualquer um pode se juntar ao grupo (e enviar para esse grupo e receber dele) ou a participação no grupo é limitada e, se for limitada, quem a limita? Os membros do grupo ficam conhecendo as identidades dos outros membros como parte do protocolo de camada de rede? Como os nós da rede interagem para entregar um datagrama *multicast* a todos os membros do grupo? Para a Internet, as respostas a todas essas perguntas envolvem o Protocolo de Gerenciamento de Grupo da Internet (*Internet Group Management Protocol* — IGMP [RFC 3376]). Assim, vamos considerar o protocolo IGMP. Mais adiante, voltaremos a essas perguntas de caráter mais geral.

Internet Group Management Protocol — IGMP

O protocolo IGMP, versão 3 [RFC 3376] opera entre um hospedeiro e o roteador diretamente conectado a ele (em termos informais, pense no roteador diretamente conectado ao hospedeiro como o roteador de primeiro salto que um hospedeiro veria no caminho até qualquer hospedeiro externo à sua própria rede local, ou como o roteador de último salto em qualquer caminho até esse hospedeiro), como mostra a Figura 4.48. Essa figura mostra os três roteadores *multicast* do primeiro salto, cada um conectado ao hospedeiro ao qual está ligado por uma interface local de saída. Essa interface local está ligada a uma LAN nesse exemplo e, embora cada LAN

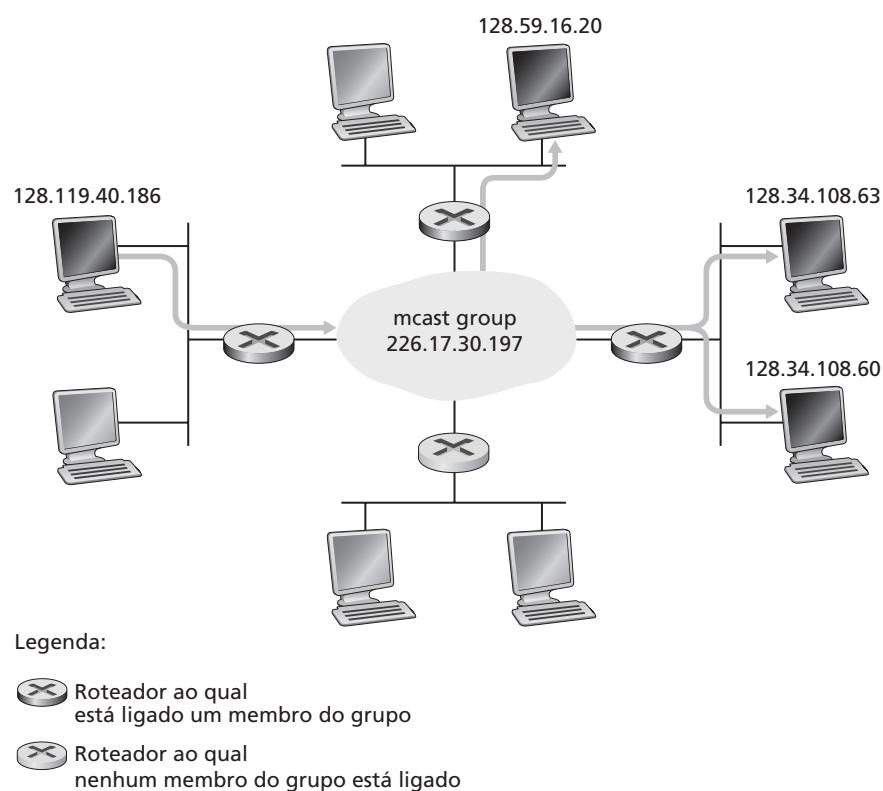


Figura 4.47 O grupo multicast: um datagrama endereçado ao grupo é entregue a todos os membros do grupo multicast

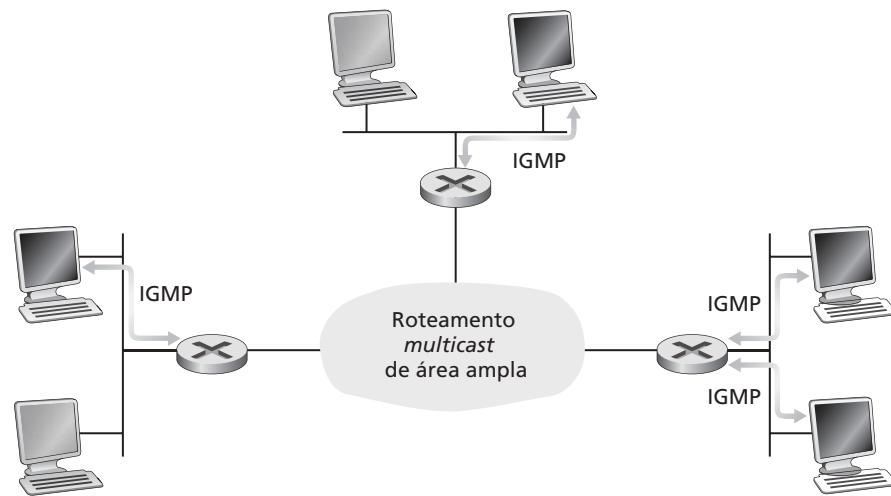


Figura 4.48 Os dois componentes de multicast de camada de rede: IGMP e protocolos de roteamento multicast

tenha vários hospedeiros ligados a ela, no máximo apenas alguns deles comumente pertencerão a um dado grupo *multicast* a qualquer determinado instante.

O IGMP provê os meios para um hospedeiro informar ao roteador conectado a ele que uma aplicação que está funcionando no hospedeiro quer se juntar a um grupo *multicast* específico. Dado que o escopo de interação do IGMP é limitado a um hospedeiro e seu roteador conectado, um outro protocolo é claramente necessário para coordenar os roteadores *multicast* (incluindo os roteadores conectados) por toda a Internet, de modo que os datagramas *multicast* sejam roteados a seus destinos finais. Essa última funcionalidade é realizada por algoritmos de roteamento *multicast* de camada de rede, como o PIM (*protocol independent multicast* — *multicast* independente de protocolo), o DVMRP (*distance vector multicast routing protocol* — protocolo de roteamento *multicast* por vetor de distância) e o MOSPF (OSPF *multicast*).

O IGMP tem apenas três tipos de mensagens. Como as mensagens ICMP, as mensagens IGMP são carregadas (encapsuladas) dentro de um datagrama IP, com um número de protocolo igual a 2. Uma mensagem de pesquisa geral de associação ao grupo (*membership_query*) é enviada por um roteador a todos os hospedeiros em uma interface conectada (por exemplo, para todos os hospedeiros em uma LAN) para determinar o conjunto de todos os grupos *multicast* aos quais se ligaram os hospedeiros naquela interface.

Os hospedeiros respondem a uma mensagem *membership_query* com uma mensagem *membership_report*. As mensagens *membership_report* podem também ser geradas por um hospedeiro quando uma aplicação se junta pela primeira vez ao grupo *multicast* sem esperar por uma mensagem *membership_query* vinda do roteador.

O último tipo de mensagem IGMP é a mensagem *leave_group*. O interessante é que essa mensagem é opcional. Mas, se é opcional, como um roteador detecta se não há mais nenhum hospedeiro em uma interface conectada que ainda está ligado a um determinado grupo *multicast*? A resposta a essa pergunta está no uso da mensagem *membership_query* IGMP. O roteador *infere* que não há nenhum hospedeiro ligado a um dado grupo *multicast* quando nenhum hospedeiro responde a uma mensagem *membership_query* com o dado endereço de grupo. Isso é um exemplo do que às vezes é denominado **estado flexível** em um protocolo de Internet. Em um protocolo de estado flexível, o estado (no caso do IGMP, o fato de haver hospedeiros ligados a um dado grupo *multicast*) será encerrado por um evento de esgotamento de temporização (nesse caso, por uma mensagem *membership_query* periódica emitida pelo roteador) se não for explicitamente renovado (nesse caso, por uma mensagem *membership_report* vinda de um hospedeiro conectado). Argumenta-se que protocolos de estado flexível resultam em controle mais simples do que o dos protocolos de estado, que não somente exigem que o estado seja explicitamente iniciado e encerrado, mas também requerem mecanismos para se recuperar da situação em

que uma entidade responsável pelo encerramento do estado encerrou prematuramente ou falhou. Uma discussão excelente sobre o estado flexível pode ser encontrada em [Raman, 1999; Ji, 2003, Lui 2004].

Algoritmos de roteamento multicast

O problema de roteamento multicast está ilustrado na Figura 4.49. Os hospedeiros agregados a esse grupo *multicast* são os de tom mais claro; os roteadores imediatamente conectados a eles também estão nesse tom. Como mostra essa figura, dentro da população de roteadores *multicast*, somente um subconjunto desses roteadores (os que têm hospedeiros conectados que se juntaram ao grupo *multicast*) realmente precisa receber o tráfego *multicast*. Ou seja, somente os roteadores A, B, E e F precisam receber esse tráfego. Uma vez que nenhum dos hospedeiros conectados ao roteador D está ligado ao grupo *multicast* e o roteador C não tem hospedeiros conectados, nem C nem D precisam receber o tráfego do grupo *multicast*.

A meta do roteamento *multicast* é encontrar uma árvore de enlaces que conecte todos os roteadores que têm hospedeiros conectados pertencentes ao grupo *multicast*. Então, pacotes *multicast* serão roteados ao longo dessa árvore desde o remetente até todos os hospedeiros pertencentes à árvore *multicast*. Evidentemente, a árvore pode conter roteadores que não têm hospedeiros conectados pertencentes ao grupo *multicast* (por exemplo, na Figura 4.48, é impossível conectar os roteadores A, B, E e F em uma árvore sem envolver o roteador C e/ou D).

Na prática, duas abordagens foram adotadas para determinar a árvore de roteamento *multicast* e ambas já foram estudadas no contexto do roteamento *broadcast*, portanto vamos somente mencioná-las. A diferença entre essas duas abordagens refere-se ao uso de uma única árvore compartilhada pelo grupo para distribuir o tráfego para todos os remetentes do grupo ou à construção de uma árvore de roteamento específica para cada remetente individual:

Roteamento multicast usando uma árvore compartilhada pelo grupo. Como no caso do *broadcast* de *spanning tree*, o roteamento *multicast* por uma árvore compartilhada por um grupo é baseado na construção de uma árvore que inclui todos os roteadores de borda cujos hospedeiros a ele conectados pertencem ao grupo *multicast*. Na prática, é usada uma abordagem centralizada para construir a árvore de roteamento

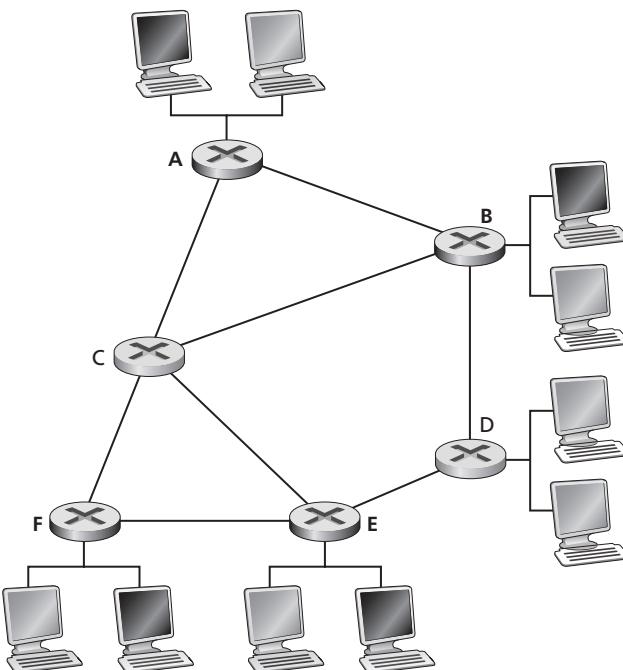


Figura 4.49 Hospedeiros multicast, seus roteadores conectados e outros roteadores

multicast — roteadores de aresta cujos hospedeiros a eles ligados pertencem ao grupo *multicast* enviam (via *unicast*) mensagens de adesão endereçadas ao nó central. Como no caso do *broadcast*, uma mensagem de adesão é transmitida usando roteamento *unicast* em direção ao centro até que a mensagem chegue a um roteador que já pertence à árvore *multicast* ou chegue até o centro. Todos os roteadores ao longo do caminho percorrido pela mensagem de adesão então transmitirão os pacotes *multicast* recebidos ao roteador de aresta que iniciou a adesão *multicast*. Uma questão crítica para o roteamento *multicast* baseado por árvore de centro é o processo utilizado para selecionar o centro. Algoritmos de seleção de centro são discutidos em [Wall, 1980; Thaler, 1997; Estrin, 1997].

Roteamento multicast usando uma árvore baseada na fonte. Enquanto o roteamento *multicast* por árvore compartilhada pelo grupo constrói uma única árvore de roteamento compartilhada para rotear pacotes de todos os hospedeiros, a segunda abordagem constrói uma árvore de roteamento *multicast* para cada fonte no grupo *multicast*. Na prática, é usado um algoritmo RPF (com nó de fonte x) para construir uma árvore de transmissão *multicast* para fazer *multicast* de datagramas que se originam na fonte x . Para ser usado em *multicast*, o algoritmo RPF que estudamos precisa sofrer uma certa adaptação. Para ver por que, considere o roteador D na Figura 4.50. Com *broadcast* RPF, o algoritmo repassaria pacotes ao roteador G , mesmo que esse roteador não tivesse nenhum hospedeiro conectado a ele pertencente ao grupo *multicast*. Embora isso não seja tão ruim para este caso, em que D tem somente um roteador adiante dele, G , imagine o que aconteceria se houvesse milhares de roteadores adiante de D ! Cada um desses milhares de roteadores receberia pacotes *multicast* indesejados. (Esse cenário não está tão longe da realidade como parece. A Mbone inicial (Casner, 1992; Macedonia, 1994), a primeira rede *multicast* global sofria precisamente desse problema no início.) A solução para o problema do recebimento de pacotes *multicast* indesejados sob RPF é conhecida como **poda**. Um roteador *multicast* que recebe pacotes *multicast* e não tenha nenhum hospedeiro conectado a ele pertencente àquele grupo enviará uma mensagem de poda ao roteador que está antes dele. Se um roteador receber mensagens de poda de cada um dos roteadores que estão adiante dele, então pode repassar uma mensagem de poda aos roteadores que estão antes dele.

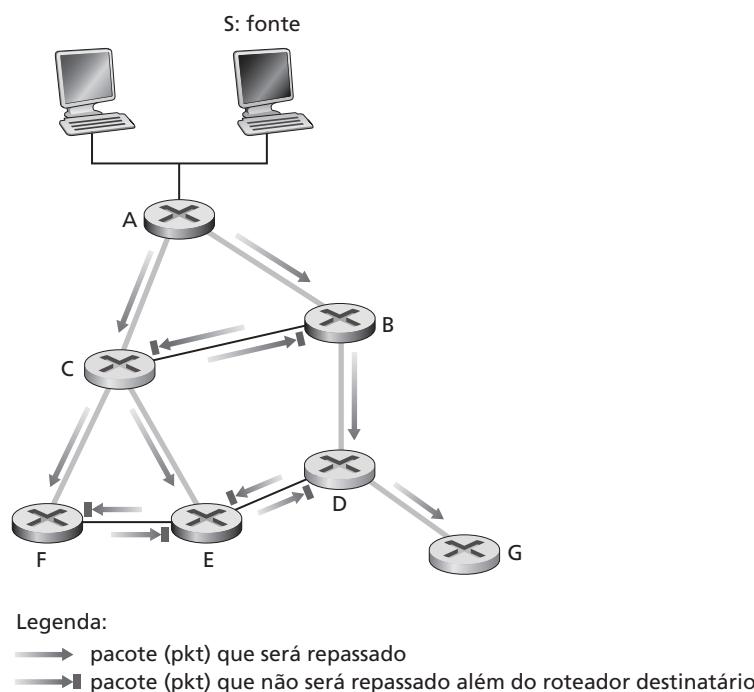


Figura 4.50 Repasse pelo caminho inverso, o caso do multicast



Roteamento multicast na Internet

O primeiro protocolo de roteamento *multicast* usado na Internet foi o **Protocolo de Roteamento Multicast por Vetor de Distâncias** (*Distance-Vector Multicast Routing Protocol* — DVMRP) [RFC 1075]. O DVMRP implementa árvores específicas de fonte com repasse de caminho inverso e poda. Esse protocolo usa um algoritmo RPF com poda, conforme discutido acima. Talvez o protocolo de roteamento multicast da Internet mais utilizado seja o protocolo de roteamento **PIM** (*multicast* independente de protocolo), o qual reconhece absolutamente dois cenários de distribuição multicast. No modo denso [RFC 3973], os membros do grupo multicast são densamente localizados; ou seja, muitos ou a maioria dos roteadores na área precisam estar envolvidos nos datagramas de roteamento multicast. O PIM de modo denso é uma técnica de repasse de caminho inverso do tipo inundar e podar semelhante em espírito ao DVMRP.

No modo esparso [RFC 4601], o número de roteadores com membros do grupo é pequeno em relação ao número total de roteadores; os membros do grupo são amplamente dispersos. O PIM de modo esparso usa pontos de encontro para definir a árvore de distribuição multicast. No SSM (*multicast* de fonte específica) [RFC 3569, RFC 4607], somente um único remetente pode enviar tráfego para a árvore *multicast*, simplificando consideravelmente a construção e manutenção da árvore.

Quando o PIM e o DVMRP são usados dentro de um domínio, o operador de rede pode configurar rotas IP *multicast* dentro do domínio, da mesma forma que protocolos de roteamento *unicast* intradomínio como RIP, IS-IS e OSPF podem ser configurados. Mas o que acontece quando as rotas multicast são requisitadas entre domínios diferentes? Existe um *multicast* equivalente do protocolo BGP interdomínio? A resposta é (literalmente) sim. [RFC 4271] define as extensões de protocolos múltiplos para o BGP que lhe permite carregar informações sobre roteamento para outros protocolos, incluindo informações *multicast*. O protocolo MSDP (*Multicast Source Discovery Protocol*) [RFC 3618, RFC 4611] pode ser usado para conectar os pontos de encontro em domínios de diferentes PIMs de modo esparso. Uma visão geral interessante sobre o atual estado do roteamento multicast na Internet pode ser encontrada em [RFC 5110].

Vamos finalizar nossa discussão sobre *multicast IP* observando que ele ainda tem de decolar de forma significativa. Para obter abordagens interessantes sobre o modelo de serviço *multicast* da Internet e questões sobre implementação, consulte [Diot, 2000; Sharma, 2003]. Não obstante, apesar da falta de emprego largo, o *multicast* em nível de rede está longe de “morrer”. O tráfego *multicast* foi conduzido por muitos anos na Internet 2, assim como as redes com as quais ela se interconecta [Internet2 Multicast, 2009]. No Reino Unido, a BBC está envolvida em testes de distribuição de conteúdo através do IP *multicast* [BBC Multicast, 2009]. Ao mesmo tempo, o *multicast* a nível de aplicação, como vimos em PPLive, no Capítulo 2, e outros sistemas *peer-to-peer*, como o *End System Multicast* [ESM, 2007], proveem distribuição de conteúdo *multicast* entre pares que utilizam protocolos *multicast* da camada de aplicação (em vez da camada de rede). Os serviços *multicast* futuros serão primeiramente implementados na camada de rede (no núcleo da rede) ou na camada de aplicação (na aresta da rede)? Enquanto a atual mania de distribuição de conteúdo *peer-to-peer* influencia a favor da camada de aplicação *multicast* pelo menos em um futuro próximo, a evolução continua a caminhar no IP *multicast*, e às vezes, no fim das contas, devagar se vai ao longe.

4.8 Resumo

Neste capítulo, iniciamos nossa viagem rumo ao núcleo da rede. Aprendemos que a camada de rede envolve todos os roteadores da rede. Por causa disso, os protocolos de camada de rede estão entre os mais desafiadores da pilha de protocolos.

Aprendemos que um roteador pode precisar processar milhares de fluxos de pacotes entre diferentes pares fonte-destino ao mesmo tempo. Para que um roteador possa processar um número tão grande de fluxos, os projetistas de redes aprenderam, com o passar dos anos, que as tarefas do roteador devem ser o mais simples possível. Muitas medidas podem ser tomadas para facilitar a tarefa do roteador, incluindo a utilização de uma camada de rede de datagramas, em vez de uma camada de rede de circuitos virtuais, a utilização de um cabeçalho aprimorado

de tamanho fixo (como no IPv6), a eliminação da fragmentação (também feita no IPv6) e o fornecimento de um único serviço de melhor esforço. Talvez a sutileza mais importante aqui seja não monitorar fluxos individuais, mas, em vez disso, tomar decisões de roteamento tendo exclusivamente como base endereços de destino hierarquicamente estruturados nos pacotes. É interessante notar que o serviço dos correios vem usando essa mesma abordagem há muitos anos.

Neste capítulo, examinamos também os princípios subjacentes de algoritmos de roteamento. Aprendemos como algoritmos de roteamento fazem a abstração da rede de computadores em um grafo com nós e enlaces. Com essa abstração, podemos pesquisar a rica teoria do roteamento de caminho mais curto em grafos, que foi desenvolvida nos últimos 40 anos nas comunidades de pesquisa operacional e de algoritmos. Vimos que há duas abordagens amplas: uma centralizada (global), em que cada nó obtém o mapeamento completo da rede e aplica independentemente o algoritmo de roteamento de caminho mais curto, e uma descentralizada, em que nós individuais têm apenas um quadro parcial da rede inteira e, mesmo assim, trabalham em conjunto para entregar pacotes ao longo das rotas mais curtas. Também estudamos como a hierarquia é utilizada para lidar com o problema da escala, dividindo redes de grande porte em domínios administrativos independentes denominados sistemas autônomos (ASs). Cada AS roteia independentemente seus datagramas pelo sistema, exatamente como cada país distribui sua correspondência postal pelo seu território. Aprendemos como abordagens centralizadas, descentralizadas e hierárquicas estão incorporadas nos principais protocolos de roteamento da Internet: RIP, OSPF e BGP. Concluímos nosso estudo de algoritmos de roteamento considerando roteamento *broadcast* e *multicast*.

Agora que concluímos nosso estudo da camada de rede, nossa jornada segue rumo a um nível mais baixo da pilha de protocolos, isto é, à camada de enlace. Como a camada de rede, a camada de enlace é também parte do núcleo da rede. Mas veremos no próximo capítulo que a camada de enlace tem a tarefa muito mais localizada de movimentar pacotes entre nós no mesmo enlace ou LAN. Embora essa tarefa possa à primeira vista parecer trivial quando comparada às tarefas da camada de rede, veremos que a camada de enlace envolve uma série de questões importantes e fascinantes que podem nos manter ocupados por muito tempo.

Exercícios de fixação

Capítulo 4 Questões de revisão

Seções 4.1 a 4.2

- Vamos rever um pouco da terminologia usada neste livro. Lembre-se de que o nome de um pacote de camada de transporte é *segmento* e que o nome de um pacote de camada de enlace é *quadro*. Qual é o nome de um pacote de camada de rede? Lembre-se de que roteadores e comutadores de camada de enlace são denominados *comutadores de pacotes*. Qual é a diferença fundamental entre um roteador e um comutador de camada de enlace? Lembre-se de que usamos o termo *roteadores* tanto para redes de datagramas quanto para redes de CVs.
- Quais são as duas funções mais importantes de camada de rede em uma rede de datagramas? Quais são as três funções mais importantes de camada de rede em uma rede com circuitos virtuais?
- Qual é a diferença entre rotear e repassar (transmitir)?
- Os roteadores de redes de datagramas e de redes de circuitos virtuais usam tabelas de repasse? Caso

usem, descreva as tabelas de repasse para ambas as classes de redes.

- Descreva alguns serviços hipotéticos que a camada de rede poderia oferecer a um pacote individual. Faça o mesmo para um fluxo de pacotes. Alguns dos serviços hipotéticos que você descreveu são fornecidos pela camada de rede da Internet? Alguns deles são fornecidos pelo modelo de serviço ATM CBR? Alguns são fornecidos pelo modelo de serviço ATM ABR?
- Cite algumas aplicações que poderiam se beneficiar do modelo de serviço ATM CBR.

Seção 4.3

- Discuta por que cada porta de entrada em um roteador de alta velocidade armazena uma cópia-sombra da tabela de repasse.

8. Três tipos de elementos de comutação são discutidos na Seção 4.3. Cite e descreva brevemente cada tipo.
9. Descreva como pode ocorrer perda de pacotes em portas de entrada. Descreva como a perda de pacotes pode ser eliminada em portas de entrada (sem usar buffers infinitos).
10. Descreva como pode ocorrer perda de pacotes em portas de saída.
11. O que é bloqueio HOL? Ele ocorre em portas de saída ou em portas de entrada?

Seção 4.4

12. Roteadores têm endereços IP? Em caso positivo, quantos endereços eles têm?
13. Qual é o equivalente binário de 32 bits para o endereço IP 223.1.3.27?
14. Visite um hospedeiro que usa DHCP para obter seu endereço IP, máscara de rede, roteador de default e endereço IP de seu servidor DNS local. Faça uma lista desses valores.
15. Suponha que haja três roteadores entre os hospedeiros da fonte e do destino. Ignorando a fragmentação, um datagrama IP enviado do hospedeiro da fonte até o hospedeiro do destino transitará por quantas interfaces? Quantas tabelas de repasse serão indexadas para deslocar o datagrama desde a fonte até o destino?
16. Suponha que uma aplicação gere blocos de 40 bytes de dados a cada 20 milissegundos e que cada bloco seja encapsulado em um segmento TCP e, em seguida, em um datagrama IP. Que porcentagem de cada datagrama será sobrecarga e que porcentagem será dados de aplicação?
17. Suponha que o Hospedeiro A envie ao Hospedeiro B um segmento TCP encapsulado em um datagrama IP. Quando o Hospedeiro B recebe o datagrama, como a camada de rede no Hospedeiro B sabe que deve passar o segmento (isto é, a carga útil do datagrama) para TCP e não para UDP ou qualquer outra coisa?
18. Suponha que você compre um roteador sem fio e o conecte a seu modem a cabo. Suponha também que seu ISP designe dinamicamente um endereço IP a seu dispositivo conectado (isto é, seu roteador sem fio). Suponha ainda que você tenha cinco PCs em casa e que usa 802.11 para conectá-los sem fio ao seu roteador também sem fio. Como são designados endereços IP aos cinco PCs? O roteador sem fio usa NAT?

19. Compare os campos de cabeçalho do IPv4 e do IPv6 e aponte suas diferenças. Eles têm algum campo em comum?
20. Afirma-se que, quando o IPv6 implementa túneis através de roteadores IPv4, o IPv6 trata os túneis IPv4 como protocolos de camada de enlace. Você concorda com essa afirmação? Explique sua resposta.

Seção 4.5

21. Compare e aponte as diferenças entre os algoritmos de estado de enlace e de vetor de distâncias.
22. Discuta como uma organização hierárquica da Internet possibilitou estender sua escala para milhões de usuários.
23. É necessário que todo sistema autônomo use o mesmo algoritmo de roteamento intra-AS? Justifique sua resposta.

Seção 4.6

24. Considere a Figura 4.37. Começando com a tabela original em D, suponha que D receba de A o seguinte anúncio:

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
Z	C	10
W	—	1
X	—	1
...

A tabela em D mudará? Em caso positivo, como mudará?

25. Compare os anúncios utilizados por RIP e OSPF e aponte suas diferenças.
26. Complete: anúncios RIP normalmente anunciam o número de saltos até vários destinos. Atualizações BGP, por outro lado, anunciam _____ aos diversos destinos.
27. Por que são usados protocolos inter-AS e intra-AS diferentes na Internet?
28. Por que considerações políticas não são tão importantes para protocolos intra-AS como o OSPF e o RIP, quanto para um protocolo de roteamento inter-AS como BGP?
29. Defina e aponte as diferenças entre os seguintes termos: *sub-rede*, *prefixo* e *rota BGP*.
30. Como o BGP usa o atributo NEXT-HOP? Como ele usa o atributo AS-PATH?
31. Descreva como um administrador de rede de um ISP de nível superior pode implementar política ao configurar o BGP.

Seção 4.7

32. Cite uma importante diferença entre a implementação da abstração *broadcast* por múltiplos *unicasts* e a de um grupo *broadcast* suportado por uma única rede (roteador).
33. Para cada uma das três abordagens gerais que estudamos para a comunicação *broadcast* (inundação não controlada, inundação controlada e *broadcast* de spanning tree), as seguintes declarações são verdadeiras ou falsas? Você pode admitir que não há perda de pacotes devido ao transbordamento do buffer e que todos os pacotes são entregues em um enlace na ordem em que foram enviados.
- a. Um nó pode receber várias cópias do mesmo pacote.
- b. Um nó pode repassar várias cópias de um pacote pelo mesmo enlace de saída.
34. Quando um hospedeiro se junta a um grupo *multicast*, ele deve mudar seu endereço IP para o endereço do grupo *multicast* ao qual está se juntando?
35. Quais são os papéis desempenhados pelo protocolo IGMP e por um protocolo de roteamento *multicast* de longa distância?
36. Qual é a diferença entre uma árvore compartilhada por um grupo e uma árvore de fonte no contexto do roteamento *multicast*?

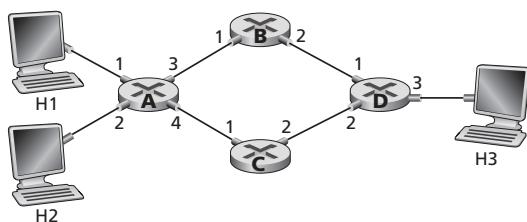


Problemas

1. Nesta questão, consideramos alguns dos prós e dos contras de redes de circuitos virtuais e redes de datagramas.
 - a. Suponha que os roteadores estivessem sujeitos a condições que os tivessem feito falhar com bastante frequência. Essas ações favorecem uma arquitetura de circuitos virtuais ou de datagramas?
 - b. Suponha que um nó de fonte e um destino solicitam que uma quantidade fixa de capacidade esteja sempre disponível em todos os roteadores no caminho entre o nó de fonte e de destino, para o uso exclusivo de fluxo de tráfego entre esse nós de fonte e de destino. Essas ações favorecem uma arquitetura de circuitos virtuais ou de datagramas? Por quê?
 - c. Suponha que os enlaces e os roteadores da rede nunca falhem e que os caminhos de roteamento usados entre as duplas de fonte/destino permanecem constantes. Nesse cenário, a arquitetura de circuitos virtuais ou de datagramas possui mais controle do tráfego?
2. Considere uma rede de circuitos virtuais. Suponha que o número do CV é um campo de 16 bits.
 - a. Qual é o número máximo de circuitos virtuais que pode ser carregado por um enlace?
 - b. Suponha que um nó central determine caminhos e números de CVs no estabelecimento da conexão. Suponha que o mesmo número de CV seja usado em cada enlace ao longo do caminho do CV. Descreva como o nó central poderia determinar o número do CV no estabelecimento da conexão. É possível haver um número de CVs ativos menor do que o máximo determinado na parte (a) e, mesmo assim, não haver nenhum número de CV livre em comum?
 - c. Suponha que sejam permitidos números diferentes de CVs em cada enlace ao longo do caminho de um CV. Durante o estabelecimento da conexão, após a determinação de um caminho fim a fim, descreva como os enlaces podem escolher seus números de CVs e configurar suas tabelas de repasse de uma maneira descentralizada, sem depender de um nó central.
3. Uma tabela de repasse bem básica em uma rede de CVs tem quatro colunas. O que significam os valores em cada uma dessas colunas? Uma tabela de repasse bem básica em uma rede de datagramas tem duas colunas. O que significam os valores em cada uma dessas colunas?
4. Considere a rede abaixo.
 - a. Suponha que esta rede seja uma rede de datagramas. Mostre a tabela de repasse no roteador A, de modo que todo o tráfego destinado ao hospedeiro H3 seja encaminhado pela interface 3.
 - b. Suponha que esta rede seja uma rede de datagramas. Você consegue compor uma tabela de repasse no roteador A, de modo que todo o tráfego de H1 destinado ao hospedeiro H3 seja encaminhado pela interface 3, enquanto todo o tráfego de H2 destinado ao hospedeiro H3 seja encaminhado pela interface 4? (Dica: esta é uma pergunta capciosa.)
 - c. Suponha, agora, que esta rede seja uma rede de circuitos virtuais e que haja uma chamada em andamento entre H1 e H3, e outra chamada em

andamento entre H2 e H3. Elabore uma tabela de repasse no roteador A, de modo que todo o tráfego de H1 destinado ao hospedeiro H3 seja encaminhado pela interface 3, enquanto todo o tráfego de H2 destinado ao hospedeiro H3 seja encaminhado pela interface 4.

- d. Admitindo o mesmo cenário de “c”, elabore tabelas de repasse em nós B, C e D.



5. Considere uma rede de circuito virtual cujo campo de número de CV tenha 2 bits. Suponha que a rede queira estabelecer um circuito virtual por quatro enlaces: enlace A, enlace B, enlace C e enlace D. Suponha que, nesse momento, cada um desses enlaces esteja carregando dois outros circuitos virtuais e que os números desses outros CVs são os seguintes:

Enlace A	Enlace B	Enlace C	Enlace D
00	01	10	11
01	10	11	00

Ao responder às perguntas a seguir, tenha em mente que cada um dos CVs existentes pode estar atravessando apenas um dos quatro enlaces.

- a. Se cada CV tiver de usar o mesmo número de CV em todos os enlaces ao longo de seu caminho, qual número de CV poderia ser designado ao novo CV?
- b. Se fosse permitido que cada CV tivesse um número de CV diferente nos diferentes enlaces ao longo de seu caminho (de modo que a tabela de repasse tenha de realizar tradução de número de CV), quantas combinações diferentes de quatro números de CVs (um para cada um dos quatro enlaces) poderiam ser usadas?
6. No texto usamos o termo *serviço orientado para conexão* para descrever a camada de transporte e *serviço de conexão* para a camada de rede. Por que essa sutileza na terminologia?
7. Na Seção 4.3 observamos que não pode haver nenhuma fila de entrada se o elemento de comutação for n vezes mais rápido do que as taxas das linhas de entrada, admitindo que n linhas de entrada tenham todas a mesma velocidade de linha. Explique (com palavras) por que isso tem de ser assim.

8. Considere o comutador abaixo. Suponha que todos os datagramas possuam o mesmo comprimento, que o comutador opere de uma maneira segmentada e sincrônica, e que em um intervalo de tempo (*time slot*) um datagrama possa ser transferido de uma porta de entrada para uma porta de saída. A malha de comutação é um crossbar no qual, no máximo, um datagrama possa ser transferido para uma determinada porta de saída em um intervalo de tempo, mas portas de saída diferentes podem receber datagramas de portas de entrada diferentes em um único intervalo de tempo. Qual é o número mínimo de intervalos de tempo necessário para transferir os pacotes mostrados das portas de entrada para suas portas de saída, admitindo qualquer ordem de sincronização de fila que você quiser (ou seja, não é necessário o bloqueio HOL)? Qual é o maior número de intervalos necessários, admitindo uma ordem de sincronização de pior caso e que uma fila de entrada não vazia nunca fica parada?



9. Considere uma rede de datagramas que usa endereços de hospedeiros de 32 bits. Suponha que um roteador tenha quatro enlaces, numerados de 0 a 3, e que os pacotes têm de ser repassados para as interfaces de enlaces como segue:

Faixa do endereço de destino **Interface de enlace**

11100000 00000000 00000000
00000000

até 0

11100000 00111111 11111111
11111111

11100000 01000000 00000000
00000000

até 1

11100000 01000000 11111111
11111111

11100000 01000001 00000000
00000000

até 2

11100001 01111111 11111111
11111111

senão 3

- a. Elabore uma tabela de repasse que tenha quatro registros, use compatibilização com o prefixo mais longo e repasse pacotes para as interfaces de enlace corretas.
- b. Descreva como sua tabela de repasse determina a interface de enlace apropriada para datagramas com os seguintes endereços:

11001000 10010001 01010001 01010101

11100001 01000000 11000011 00111100

11100001 10000000 00010001 01110111

10. Considere uma rede de datagramas que utiliza endereços de 8 bits. Suponha que um roteador utilize compatibilização com o prefixo mais longo e tenha a seguinte tabela de repasse:

Prefixo a ser comparado	Interface
00	0
010	1
011	2
10	2
11	3

Para cada uma das quatro interfaces, forneça a faixa associada de endereços de hospedeiros de destino e o número de endereços na faixa.

11. Considere uma rede de datagramas que usa endereços de hospedeiros de 8 bits. Suponha que um roteador use compatibilização com o prefixo mais longo e tenha a seguinte tabela de repasse:

Prefixo a ser comparado	Interface
1	0
10	1
111	2
senão	3

Para cada uma das quatro interfaces, forneça a faixa associada de endereços de hospedeiros de destino e o número de endereços na faixa.

12. Considere um roteador que interconecta três sub-redes: Sub-rede 1, Sub-rede 2 e Sub-rede 3. Suponha que todas as interfaces de cada uma dessas três sub-redes tenha de ter o prefixo 223.1.17/24. Suponha também que a Sub-rede 1 tenha de suportar até 63 interfaces, a Sub-rede 2 tenha de suportar até 95 interfaces e a Sub-rede 3, 16 interfaces. Dê três endereços de rede (da forma a.b.c.d/x) que satisfaçam essas limitações.

13. Na Seção 4.2.2 é dado um exemplo de tabela de repasse (usando compatibilização de prefixo mais longo). Reescreva a tabela usando a notação a.b.c.d/x em vez da notação de cadeia binária.

14. No Problema 9, solicitamos que você elaborasse uma tabela de repasse (usando compatibilização de prefixo mais longo). Reescreva a tabela usando a notação a.b.c.d/x em vez da notação de cadeia binária.

15. Considere uma sub-rede com prefixo 128.119.40.128/26. Dê um exemplo de um endereço IP (na forma xxx.xxx.xxx.xxx) que possa ser designado para essa rede. Suponha que um ISP possua o bloco de endereços na forma 128.119.40.64/25. Suponha que ele queira criar quatro sub-redes a partir desse bloco, e que cada bloco tenha o mesmo número de endereços IP. Quais são os prefixos (na forma a.b.c.d/x) para essas quatro sub-redes?

16. Considere a topologia mostrada na Figura 4.17. Começando pela parte superior e prosseguindo em sentido horário, denomine as três sub-redes como Redes A, B, e C. Denomine as sub-redes sem hospedeiros como Redes D, E, e F.

- a. Designe endereços de rede a cada uma dessas seis sub-redes, com as seguintes restrições: todos os endereços deverão ser alocados a partir de 214.97.254/23; a Sub-rede A deve ter endereços suficientes para suportar 250 interfaces; a Sub-rede B deve ter endereços suficientes para suportar 120 interfaces e a Sub-rede C deve ter endereços suficientes para suportar 120 interfaces. É claro que cada uma das sub-redes D, E e F devem poder suportar duas interfaces. Para cada sub-rede, a designação deve tomar a forma a.b.c.d/x ou a.b.c.d/x – e.f.g.h/y.

- b. Usando a resposta que você deu para a parte (a), elabore as tabelas de repasse (usando compatibilização de prefixo mais longo) para cada um dos três roteadores.

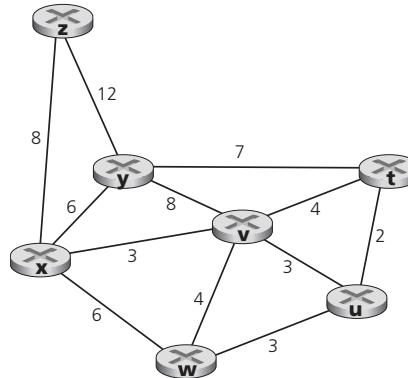
17. Considere enviar um datagrama de 2.400 bytes por um enlace que tem uma MTU de 700 bytes. Suponha que o datagrama original esteja marcado com o número de identificação 422. Quantos fragmentos são gerados? Quais são os valores em vários campos dos datagramas IP gerados em relação à fragmentação?

18. Suponha que, entre o Hospedeiro de origem A e o Hospedeiro destinatário B, os datagramas estejam limitados a 1.500 bytes (incluindo cabeçalho). Admitindo um cabeçalho IP de 20 bytes, quantos datagramas seriam necessários para enviar um arquivo MP3 de 5 milhões de bytes? Explique como você obteve a resposta.

19. Considere a configuração de rede da Figura 4.22. Suponha que o ISP designe ao roteador o endereço

24.34.112.235 e que o endereço de rede da rede residencial seja 192.168.1/24.

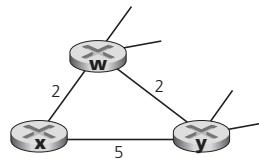
- Designe endereços a todas as interfaces na rede residencial.
 - Suponha que haja duas conexões TCP em curso em cada hospedeiro, todas para a porta 80 no hospedeiro 128.119.40.86. Forneça os seis registros correspondentes na tabela de tradução NAT.
20. Suponha que você esteja interessado em detectar o número de hospedeiros por trás da NAT. Você observa que a camada IP traz um número de identificação, de modo sequencial, em cada pacote IP. O número de identificação do primeiro pacote IP gerado por um hospedeiro é aleatório, e os números de identificação dos pacotes IP subsequentes são determinados sequencialmente. Admita que todos os pacotes IP gerados por hospedeiros por trás da NAT sejam enviados para o mundo exterior.
- Com base nessa observação e admitindo que você pode analisar todos os pacotes enviados para fora pela NAT, você pode descrever uma técnica simples que detecte o número de hospedeiros únicos por trás da NAT? Justifique sua resposta.
 - Se os números de identificação não são determinados de maneira sequencial, e sim aleatória, sua técnica funcionaria? Justifique sua resposta.
21. Neste problema estudaremos o impacto das NATs sobre aplicações P2P. Suponha que um parceiro com nome de usuário Arnold descubra, por meio de consulta, que um parceiro com nome de hospedeiro Bernard tem um arquivo que ele, Arnold, quer descarregar. Suponha também que Bernard e Arnold estejam por trás de uma NAT. Tente elaborar uma técnica que permita que Arnold estabeleça uma conexão TCP com Bernard sem a configuração da NAT de aplicação específica. Se você tiver dificuldade na elaboração dessa técnica, discuta o porquê.
22. Considerando a Figura 4.27, enumere os caminhos de *y* a *u* que não tenham nenhum laço.
23. Repita o problema 22 considerando os caminhos de *z* a *z*, *z* a *u* e *z* a *w*.
24. Considere a seguinte rede. Com os custos de enlace indicados, use o algoritmo do caminho mais curto de Dijkstra para calcular o caminho mais curto de *x* até todos os nós da rede. Mostre como o algoritmo funciona calculando uma tabela semelhante à Tabela 4.3.



25. Considere a rede mostrada no Problema 24. Usando o algoritmo de Dijkstra e mostrando seu trabalho usando uma tabela semelhante à Tabela 4.3, faça o seguinte:
- Calcule o caminho mais curto de *t* até todos os nós da rede.
 - Calcule o caminho mais curto de *u* até todos os nós da rede.
 - Calcule o caminho mais curto de *v* até todos os nós da rede.
 - Calcule o caminho mais curto de *w* até todos os nós da rede.
 - Calcule o caminho mais curto de *y* até todos os nós da rede.
 - Calcule o caminho mais curto de *z* até todos os nós da rede.
26. Considere a rede mostrada a seguir e admita que cada nó inicialmente conheça os custos até cada um de seus vizinhos. Considere o algoritmo de vetor de distâncias e mostre os registros na tabela de distâncias para o nó *z*.
- O diagrama mostra uma rede com 5 nós rotulados com letras maiúsculas. As distâncias entre os nós são as seguintes:

 - u* a *v*: 1
 - u* a *y*: 2
 - v* a *z*: 6
 - v* a *x*: 3
 - z* a *x*: 2
 - x* a *y*: 3
27. Considere uma topologia geral (isto é, não a rede específica mostrada anteriormente) e uma versão síncrona do algoritmo de vetor de distâncias. Suponha que, a cada iteração, um nó troque seus vetores de distâncias com seus vizinhos e receba os vetores de distâncias deles. Supondo que o algoritmo comece com cada nó conhecendo apenas os custos até seus vizinhos imediatos, qual é o número máximo de iterações requeridas até que o algoritmo distribuído converja? Justifique sua resposta.

28. Considere o fragmento de rede mostrado a seguir. x tem apenas dois vizinhos ligados a ele: w e y . w tem um caminho de custo mínimo até o destino u (não mostrado) de 5 e y tem um caminho de custo mínimo u de 6. Os caminhos completos de w e de y até u (e entre w e y) não são mostrados. Todos os valores dos custos de enlace na rede são números inteiros estritamente positivos.

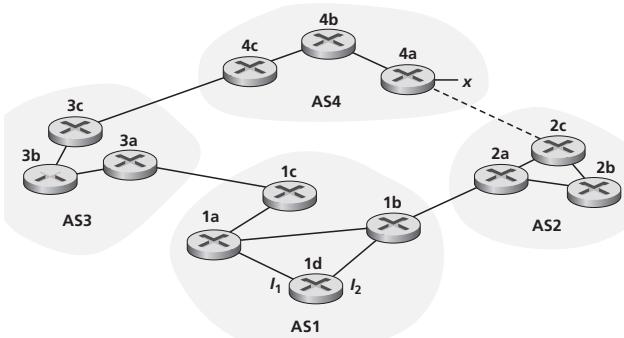


- a. Dê os vetores de distâncias de x para os destinos w, y e u .
 - b. Dê uma mudança de custo de enlace para $c(x, w)$ ou para $c(x, y)$ tal que x informará a seus vizinhos um novo caminho de custo mínimo até u como resultado da execução do algoritmo de vetor de distâncias.
 - c. Dê uma mudança de custo de enlace para $c(x, w)$ ou para $c(x, y)$ tal que x não informará a seus vizinhos um novo caminho de custo mínimo até u como resultado da execução do algoritmo de vetor de distâncias.
29. Considere a topologia de três nós mostrada na Figura 4.30. Em vez de ter os custos de enlace da Figura 4.30, os custos de enlace são: $c(x, y) = 3, c(y, z) = 6, c(z, x) = 4$. Calcule as tabelas de distâncias após a etapa de inicialização e após cada iteração de uma versão síncrona do algoritmo de vetor de distâncias (como fizemos em nossa discussão anterior da Figura 4.30).
30. Considere o problema da contagem até o infinito no roteamento de vetor de distâncias. Esse problema ocorrerá se reduzirmos o custo de um enlace? Por quê? E se conectarmos dois nós que não possuem enlace?
31. Demonstre que, para o algoritmo de vetor de distância na Figura 4.30, cada valor no vetor de distância $D(x)$ é não crescente e, consequentemente, se estabilizará em um número finito de etapas.
32. Considere a Figura 4.31. Suponha que haja outro roteador w , conectado aos roteadores y e z . Os custos de todos os enlaces são: $c(x, y) = 4, c(x, z) = 50, c(y, w) = 1, c(z, w) = 1, c(y, z) = 3$. Suponha que a reversão envenenada seja utilizada no algoritmo de roteamento de vetor de distância.
- a. Quando o roteamento de vetor de distância é estabilizado, os roteadores w, y e z informam uns aos outros sobre suas distâncias para x . Que

valores de distância eles podem informar uns aos outros?

- b. Agora suponha que o custo do enlace entre x e y aumente para 60. Haverá um problema de contagem até o infinito mesmo se a reversão envenenada for utilizada? Por quê? Por que não? Se houver um problema de contagem até o infinito, então quantas iterações são necessárias para o roteamento de vetor de distância alcançar um estágio estável novamente? Justifique sua resposta.

33. Descreva como laços no caminho podem ser detectados com BGP.
34. Um roteador BGP sempre escolherá uma rota sem laços com o menor comprimento de AS-path? Justifique sua resposta.
35. Considere a rede abaixo. Suponha que AS3 e AS2 estejam rodando o OSPF para seu protocolo de roteamento intra-AS. Suponha que AS1 e AS4 estejam rodando o RIP para seu protocolo de roteamento intra-AS. Suponha que o eBGP e o iBGP sejam usados para o protocolo de roteamento intra-AS. Inicialmente, suponha que não haja enlace físico entre AS2 e AS4.
- a. O roteador 3c sabe sobre o prefixo x por qual protocolo de roteamento: OSPF, RIP, eBGP ou iBGP?
 - b. O roteador 3a sabe sobre o prefixo x por qual protocolo de roteamento?
 - c. O roteador 1c sabe sobre o prefixo x por qual protocolo de roteamento?
 - d. O roteador 1d sabe sobre o prefixo x por qual protocolo de roteamento?



36. Referindo-se ao problema anterior, uma vez que o roteador 1d sabe sobre x , ele inserirá uma entrada (x, l) em sua tabela de repasse.
- a. l será igual a l_1 ou l_2 para essa entrada? Justifique a resposta em uma frase.
 - b. Agora suponha que haja um enlace físico entre AS2 e AS4, ilustrado pela linha pontilhada. Suponha que o roteador 1d saiba que x é acessível por

- meio de AS2 e de AS3. l será definido para l_1 ou l_2 ? Justifique a resposta em uma frase.
- c. Agora suponha que haja outro AS, denominado AS5, que fica no caminho entre AS2 e AS4 (não ilustrado no diagrama). Suponha que o roteador 1d saiba que x é acessível por meio de AS2 AS5 AS4, bem como de AS3 AS4. l será definido para l_1 ou l_2 ? Justifique a resposta em uma frase.
37. Considere a seguinte rede. O ISP B provê serviço nacional de backbone ao ISP regional A. O ISP C provê serviço nacional de backbone ao ISP regional D. Cada ISP consiste em um AS. Usando BGP, B e C formam pares entre si em dois lugares. Considere tráfego na direção de A a D. B preferiria passar esse tráfego para C na Costa Oeste (de modo que C teria de absorver o custo de carregar o tráfego através do país), enquanto C preferiria receber o tráfego via seu ponto de formação de par com B na Costa Leste (de modo que B carregaria o tráfego através do país). Qual mecanismo BGP C poderia usar de modo que B entregasse o tráfego de A a D em seu ponto de formação de par na Costa Leste? Para responder a essa pergunta, você precisará estudar muito bem a especificação do BGP.
-
38. Na Figura 4.42, considere a informação de caminho que chega às sub-redes stub W, X e Y. Com base na informação disponível em W e X, quais são as respectivas visões da topologia da rede? Justifique sua resposta. A topologia vista de Y é mostrada a seguir:
-
39. Considere que a Figura 4.42.B nunca encaminharia tráfego destinado a Y através de X baseado no roteamento BGP. Mas existem muitas aplicações conhecidas para as quais os pacotes de dados vão primeiro para X e depois para Y. Identifique tal aplicação, e descreva como os pacotes de dados percorrem um caminho não determinado pelo roteamento BGP.
40. Na Figura 4.42, suponha que haja outra rede stub V cliente de ISP A. Suponha que B e C tenham uma relação de interconexão, e que A seja cliente de B e de C. Suponha, ainda, que A gostaria de ter o tráfego destinado para W vindo apenas de B, e o tráfego destinado para V vindo de B ou C. Como A deveria anunciar suas rotas para B e C? Quais rotas C recebe?
41. Considere a rede de sete nós (com nós rotulados de t a z) do Problema 4. Mostre a árvore de custo mínimo com raiz em z que inclua (como hospedeiros finais) os nós u , v , w e y . Justifique informalmente por que sua árvore é uma árvore de custo mínimo.
42. Considere as duas abordagens básicas identificadas para fazer broadcast: emulação de unicast e broadcast de camada de rede (isto é, assistido por roteador) e suponha que seja usado broadcast de spanning tree para fazer broadcast de camada de rede. Considere um único remetente e 32 destinatários. Suponha que o remetente esteja conectado aos destinatários por uma árvore binária de roteadores. Qual é o custo para enviar um pacote broadcast nos casos da emulação de unicast e de broadcast de camada de rede para essa topologia? Aqui, cada vez que um pacote (ou cópia de um pacote) é enviado por um único enlace, ele incorre em uma unidade de custo. Qual topologia para interconectar o remetente, os destinatários e os roteadores fará com que os custos da emulação de unicast e do broadcast verdadeiro de camada de rede fiquem o mais longe possível um do outro? Você pode escolher quantos roteadores quiser.
43. Considere a operação do algoritmo de repasse de caminho inverso (RPF) na Figura 4.44. Usando a mesma topologia, descubra um conjunto de caminhos de todos os nós até o nó de fonte A (e indique esses caminhos em um grafo usando linhas grossas como as da Figura 4.44 de modo que, se esses caminhos forem os caminhos de menor custo, então o nó B receberia uma cópia das mensagens de broadcast de A dos nós A, C e D sob RPF).
44. Considere a topologia ilustrada na Figura 4.44. Suponha que todos os enlaces tenham custo unitário e que o nó E é a fonte de broadcast. Usando setas como as mostradas na Figura 4.44, indique enlaces pelos quais pacotes serão repassados usando RPF e enlaces pelos quais pacotes não serão repassados, dado que o nó E é a fonte.
45. Repita o Problema 44 utilizando o gráfico do Problema 24. Admita que z na fonte de broadcast e que os custos do enlace são mostrados no Problema 22.
46. Considere a topologia mostrada na Figura 4.46 e suponha que cada enlace tenha preço unitário. Suponha que o nó C seja escolhido como o centro de um algoritmo de roteamento multicast baseado em centro.

- Admitindo que cada roteador conectado use seu caminho de menor custo até o nó C para enviar mensagens de adesão a C , desenhe a árvore de roteamento baseada no centro resultante. Essa árvore é uma árvore de custo mínimo? Justifique sua resposta.
47. Repita o Problema 46, usando o gráfico do Problema 24. Admita que o nó central seja v .
48. Na Seção 4.5.1 estudamos o algoritmo de roteamento de estado de enlace de Dijkstra para calcular os caminhos *unicast* que são, individualmente, os caminhos de menor custo da fonte até todos os destinos. Poderíamos imaginar que a união desses caminhos forme uma **árvore unicast de caminho de menor custo** (ou uma árvore *unicast* de caminho mais curto, se todos os custos de enlaces fossem idênticos). Construindo um exemplo que nega essa afirmação (um contraexemplo), mostre que a árvore de caminho de menor custo *nem sempre* é o mesmo que uma *spanning tree* mínima.
49. Considere uma rede na qual todos os nós estão conectados a três outros nós. Em uma única etapa de tempo, um nó pode receber de seus vizinhos todos os pacotes transmitidos por *broadcast*, duplicar os pacotes e enviá-los a todos os seus vizinhos (exceto ao nó que enviou um dado pacote). Na próxima etapa, nós vizinhos podem receber, duplicar e repassar esses pacotes e assim por diante. Suponha que seja utilizada a inundação não controlada para prover *broadcast* a essa rede. Na etapa de tempo t , quantas cópias do pacote *broadcast* serão transmitidas, admitindo que, durante a etapa 1, um único pacote *broadcast* é transmitido pelo nó da fonte a seus três vizinhos?
50. Vimos na Secção 4.7 que não há nenhum protocolo de camada de rede que possa ser usado para identificar os hospedeiros que participam de um grupo *multicast*. Isso posto, como aplicações *multicast* podem aprender as identidades dos hospedeiros que estão participando de um grupo *multicast*?
51. Projete (dê uma descrição fictícia em pseudocódigo) um protocolo de nível de aplicação que mantenha os endereços de hospedeiros para todos os hospedeiros participantes de um grupo *multicast*. Identifique especificamente o serviço de rede (*unicast* ou *multicast*) que é usado por seu protocolo e indique se seu protocolo está enviando mensagens dentro da banda ou fora da banda (com relação ao fluxo de dados de aplicação entre os participantes do grupo *multicast*) e por quê.
52. Qual é o tamanho do espaço de endereço *multicast*? Suponha agora que dois grupos *multicast* escolham aleatoriamente um endereço *multicast*. Qual é a probabilidade de que escolham o mesmo endereço? Suponha agora que mil grupos *multicast* estejam em operação ao mesmo tempo e escolham seus endereços de grupo *multicast* aleatoriamente. Qual é a probabilidade de que uns interfiram nos outros?



Questões dissertativas

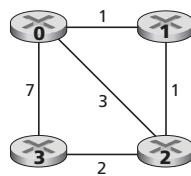
1. Descubra três empresas que estão vendendo roteadores de alta velocidade. Compare os roteadores mais potentes que elas comercializam. Como você definiu “mais potentes”?
2. Use o serviço whois da American Registry for Internet Numbers (<http://www.arin.net/whois>) para determinar os blocos de endereços IP para três universidades. Os serviços whois podem ser usados para determinar com certeza a localização de um endereço IP específico?
3. É possível escrever o programa ping cliente (usando mensagens ICMP) em Java? Justifique sua resposta.
4. Na Seção 4.4, dissemos que a disponibilização do IPv6 tem sido lenta. Por quê? O que é necessário para acelerar sua disponibilização?
5. Discuta alguns dos problemas que as NATs criam para a segurança Ipsec (veja [Phifer, 2000]).
6. Pesquise sobre o protocolo UPnP. Descreva especificamente as mensagens que um hospedeiro utiliza para reconfigurar uma NAT.
7. Suponha que os ASs X e Z não estejam diretamente conectados, mas, em vez disso, estejam conectados pelo AS Y. Suponha também que X tenha um acordo de interligação com Y e que Y tenha um acordo de interligação com Z. Finalmente, suponha que Z queira carregar todo o tráfego de Y, mas não queira carregar o tráfego de X. O BGP permite que Z implemente essa política?
8. Na Seção 4.7, identificamos uma série de aplicações *multicast*. Quais dessas aplicações se ajustam bem ao modelo minimalista de serviço *multicast* da Internet? Por quê? Quais aplicações não se ajustam particularmente bem a esse modelo de serviço?



Tarefas de programação

Nesta tarefa de programação, você escreverá um conjunto ‘distribuído’ de procedimentos que implementam um roteamento de vetor de distâncias assíncrono distribuído para a rede mostrada a seguir.

Você deve escrever as seguintes rotinas que “rodarão” assincronamente dentro do ambiente emulado fornecido para essa tarefa. Para o nó 0, você escreverá as rotinas:



rtinit0(). Essa rotina será chamada uma vez no início da emulação. *rtinit0()* não tem argumentos. Você deve inicializar sua tabela de distâncias no nó 0 para refletir os custos diretos de 1, 3 e 7 até os nós 1, 2 e 3, respectivamente. Na figura anterior, todos os enlaces são bidirecionais e os custos em ambas as direções são idênticos. Após inicializar a tabela de distâncias e quaisquer outras estruturas de dados necessárias às rotinas de seu nó 0, este deve então enviar a seus vizinhos diretamente ligados (nesse caso, 1, 2 e 3) o custo de seus caminhos de custo mínimo para todos os outros nós da rede. Essa informação do custo mínimo é enviada aos nós vizinhos em um pacote de atualização de roteamento denominado rotina *tolayer2()*, como descrita na tarefa completa. O formato do pacote de atualização de roteamento também está descrito na tarefa completa.

*rtupdate0(struct rtpkt *rcvdpkt)*. Essa rotina será chamada quando o nó 0 receber um

pacote de roteamento que foi enviado a ele por um de seus vizinhos diretamente conectados. O parâmetro **rcvdpkt* é um indicador para o pacote que foi recebido. *rtupdate0()* é o coração do algoritmo de vetor de distâncias. Os valores que ele recebe em um pacote de atualização de roteamento de algum outro nó *i* contêm os custos correntes do caminho mais curto de *i* para todos os outros nós da rede. *rtupdate0()* usa esses valores recebidos para atualizar sua própria tabela de distâncias (como especificada pelo algoritmo de vetor de distâncias). Se seu próprio custo mínimo até um outro nó mudar como resultado da atualização, o nó 0 informará essa mudança no custo mínimo a seus vizinhos diretamente conectados enviando a eles um pacote de roteamento. Lembre-se de que no algoritmo de vetor de distâncias apenas os nós conectados diretamente trocarão pacotes de roteamento. Assim, os nós 1 e 2 vão se comunicar, mas os nós 1 e 3, não.

Rotinas semelhantes são definidas para os nós 1, 2 e 3. Assim, você escreverá oito procedimentos ao todo: *rtinit0()*, *rtinit1()*, *rtinit2()*, *rtinit3()*, *rtupdate0()*, *rtupdate1()*, *rtupdate2()*, *rtupdate3()*. Juntas, essas rotinas implementarão um cálculo assíncrono, distribuído, das tabelas de distâncias para a topologia e os custos mostrados na figura apresentada anteriormente.

No site http://www.aw.com/kurose_br você poderá encontrar todos os detalhes da tarefa de programação, bem como o código em C de que precisará para criar o ambiente hardware/software simulado. Também está disponível uma versão da tarefa em Java.



Wireshark Lab

No Companion Website deste livro, www.aw.com/kurose_br, você encontrará duas tarefas de laboratório Wireshark, em inglês. A primeira examina a operação do

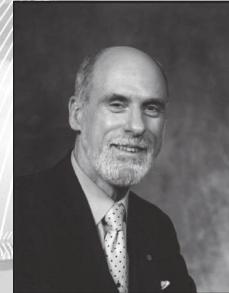
protocolo IP e, em particular, o formato do datagrama IP. A segunda explora a utilização do protocolo ICMP nos comandos ping e traceroute.



Entrevista

Vinton G. Cerf

Vinton G. Cerf é vice-presidente e evangelista-chefe do Google. Ele trabalhou por mais 16 anos na MCI, ocupando diversos cargos, sendo o último como vice-presidente sênior de Estratégia de Tecnologia. É muito conhecido pela coautoria dos protocolos TCP/IP e da arquitetura da Internet. Durante sua gestão de 1976 a 1982 na Advanced Research Projects Agency do Departamento de Defesa dos Estados Unidos (DARPA), desempenhou um papel fundamental na liderança do desenvolvimento de pacotes de dados da Internet e relacionados com a Internet e de técnicas de segurança. Em 2005, ele recebeu a Medalha Presidencial de Liberdade dos EUA, e a Medalha Nacional de Tecnologia dos EUA em 1997. Ele é bacharel em Matemática pela Stanford University e Doutor em ciência da computação pela UCLA.



O que o fez se decidir pela especialização em rede?

Eu trabalhava como programador na UCLA no final da década de 1960 com o patrocínio da Advanced Research Projects Agency do Departamento de Defesa dos Estados Unidos (na época conhecida como ARPA e hoje, como DARPA). Meu trabalho era desenvolvido no laboratório do professor Leonard Kleinrock no Network Measurement Center da recém-criada ARPAnet. O primeiro nó da ARPAnet foi instalado na UCLA no dia 1º de setembro de 1969. Eu era responsável pela programação de um computador utilizado para coletar informações de desempenho da ARPAnet e passar essas informações para comparação com modelos matemáticos e previsões de desempenho da rede.

Eu e vários outros estudantes de pós-graduação éramos responsáveis pelo trabalho com o que era conhecido como protocolos de nível de hospedeiro da ARPAnet — os procedimentos e formatos que permitiam a interação dos muitos tipos diferentes de computadores na rede. Era uma exploração fascinante de um novo mundo (para mim) de computação distribuída e comunicação.

Quando começou a projetar o IP, você imaginava que esse protocolo tornar-se-ia tão predominante quanto é hoje?

Quando Bob Kahn e eu começamos a trabalhar nisso, em 1973, acho que estávamos muito mais preocupados com a questão central: como fazer com que redes de pacotes heterogêneas interagissemumas com as outras, admitindo que não poderíamos modificá-las. Esperávamos descobrir um modo que permitisse que um conjunto arbitrário de redes de comutação de pacotes fosse interligado de maneira transparente, de modo que os computadores componentes das redes pudesse se comunicar fim a fim sem precisar de nenhuma tradução entre eles. Acho que sabíamos que estávamos lidando com uma tecnologia poderosa e expansível, mas duvido que tivéssemos uma ideia muito clara do que seria o mundo com centenas de milhões de computadores todos interligados com a Internet.

Em sua opinião, qual é o futuro das redes e da Internet? Quais são os grandes obstáculos/desafios que estão no caminho do seu desenvolvimento?

Acredito que a Internet, em particular, e as redes, em geral, continuarão a proliferar. Hoje já existem evidências convincentes de que haverá bilhões de dispositivos habilitados para a Internet, entre eles equipamentos como telefones celulares, refrigeradores, PDAs, servidores residenciais, televisões, bem como a costumeira coleção de





laptops, servidores e assim por diante. Entre os grandes desafios estão o suporte para a mobilidade, a duração das baterias, a capacidade dos enlaces de acesso à rede e a escalabilidade ilimitada do núcleo ótico da rede. A tarefa com a qual estou profundamente envolvido no Jet Propulsion Laboratory é o projeto de uma extensão interplanetária da Internet. Precisaremos descobrir um atalho para passar do IPv4 [endereços de 32 bits] para o IPv6 [128 bits]. A lista é comprida!

Quais pessoas o inspiraram profissionalmente?

Meu colega Bob Kahn; o orientador de minha tese, Gerald Estrin; meu melhor amigo, Steve Crocker (nós nos conhecemos na escola secundária e ele me apresentou aos computadores em 1960!); e os milhares de engenheiros que continuam a desenvolver a Internet.

Você pode dar algum conselho aos estudantes que estão ingressando no campo das redes/Internet?

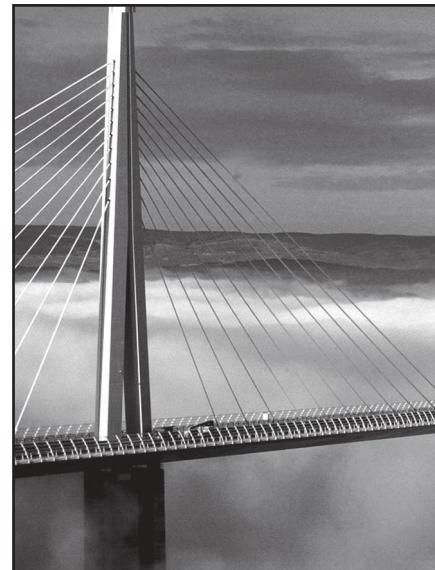
Não limitem seu pensamento aos sistemas existentes — imaginem o que poderia ser possível e então comecem a trabalhar para descobrir um meio de sair do estado atual das coisas e chegar lá. Ousem sonhar; eu e alguns colegas do Jet Propulsion Laboratory estamos trabalhando no projeto de uma extensão interplanetária da Internet terrestre. A implementação dessa rede pode levar décadas, uma missão por vez, mas, usando uma paráfrase: “O homem deve tentar alcançar o que está fora do seu alcance; senão, para que existiriam os céus?”





Capítulo 5

A camada de enlace e redes locais



No capítulo anterior, aprendemos que a camada de rede fornece um serviço de comunicação entre dois hospedeiros. Como mostra a Figura 5.1, esse caminho de comunicação consiste em uma série de enlaces de comunicação que tem início no hospedeiro de origem, passa por uma série de roteadores e termina no hospedeiro de destino. À medida que continuamos a descer a pilha de protocolos, da camada de rede até a camada de enlace, é natural que imaginemos como pacotes são enviados pelos enlaces individuais dentro do caminho de comunicação fim a fim. Como os datagramas de camada de rede são encapsulados nos quadros de camada de enlace para transmissão por um único enlace? Protocolos de camada de enlace podem prover transferência confiável de dados de roteador a roteador? Protocolos de camada de enlace diferentes podem ser usados nos diferentes enlaces ao longo do caminho de comunicação? Neste capítulo responderemos a essas e a outras perguntas.

Ao discutir a camada de enlace, descobriremos que há dois tipos de canais de camada de enlace completamente diferentes. O primeiro tipo são os canais de *broadcast*, que são comuns em redes locais (LANs), LANs sem fio, redes por satélite e redes de acesso híbridas de cabo coaxial e de fibra (HFC). No caso do canal de *broadcast*, muitos hospedeiros estão conectados ao mesmo canal de comunicação e é preciso um protocolo de acesso ao meio para coordenar transmissões e evitar colisões entre quadros transmitidos. O segundo tipo de canal de camada de enlace é o enlace de comunicação ponto a ponto, tal como o existente entre dois roteadores ou entre um modem discado residencial e um roteador ISP. Coordenar o acesso a um enlace ponto a ponto é trivial, mas há ainda questões importantes referentes a enquadramento, transferência confiável de dados, detecção de erros e controle de fluxo.

Neste capítulo estudaremos diversas tecnologias importantes de camada de enlace. Examinaremos minuciosamente a Ethernet, de longe a tecnologia predominante para LANs com fio. Estudaremos também o protocolo ponto a ponto (*point-to-point protocol* — PPP), o preferido para modems discados em hospedeiros residenciais. Embora certamente sejam tópicos de camada de enlace, as redes Wi-Fi e, mais geralmente, as LANs sem fio serão estudadas no Capítulo 6, dedicado às redes de computadores sem fio e à mobilidade.

5.1 Camada de enlace: introdução e serviços

Vamos começar com um pouco de terminologia útil. Achamos que, neste capítulo, é conveniente que nos refiramos aos hospedeiros e roteadores apenas como **nós**, já que, como veremos em breve, não estaremos particularmente preocupados em saber se um nó é um roteador ou um hospedeiro. Também nos referiremos aos canais

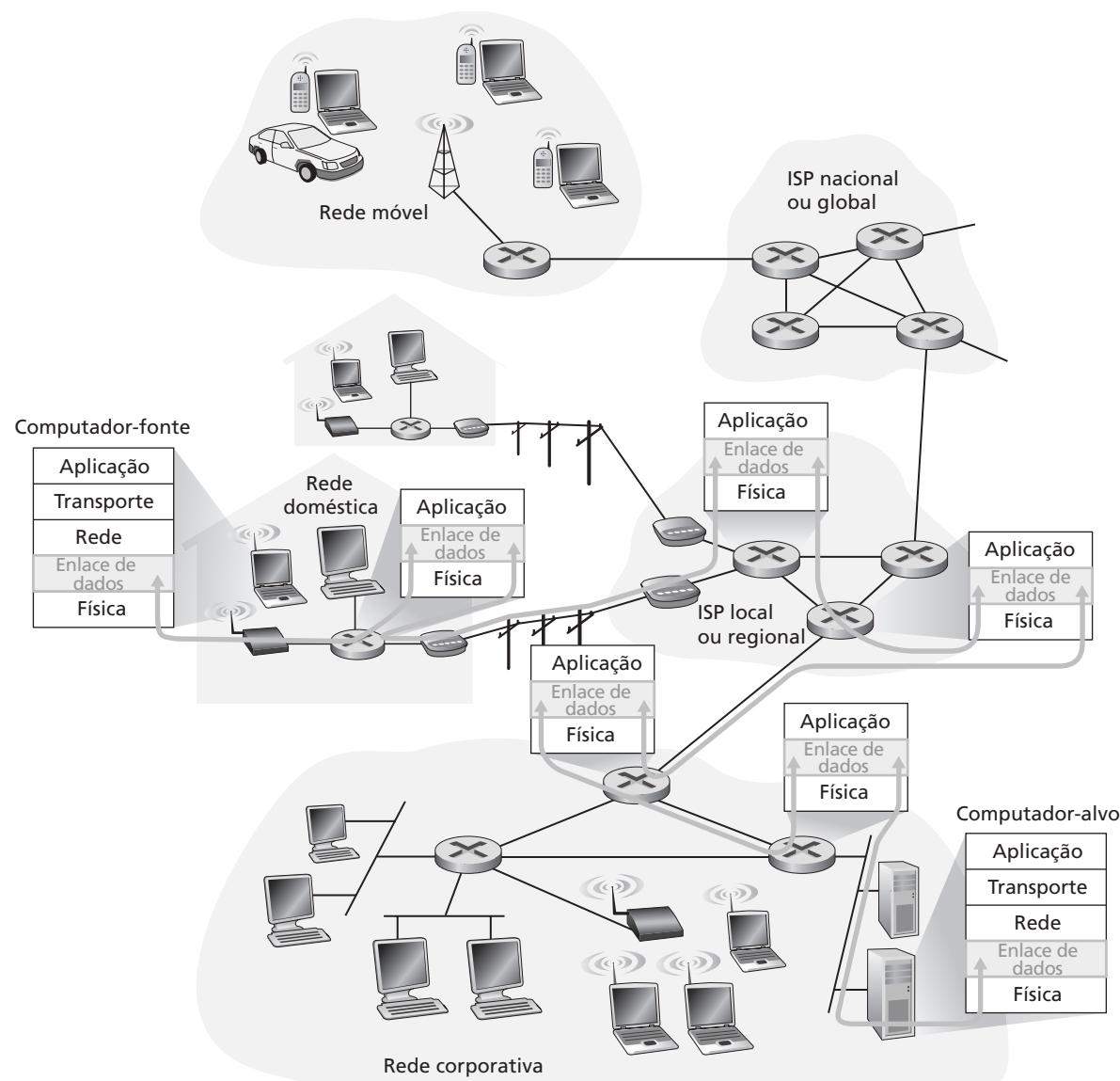


Figura 5.1 A camada de enlace

de comunicação que conectam nós adjacentes ao longo dos caminhos de comunicação como **enlaces**. Para levar um datagrama de um hospedeiro de origem até um hospedeiro de destino, o datagrama tem de ser transportado sobre cada um dos enlaces individuais existentes no caminho fim a fim. Considerando um dado enlace, um nó transmissor encapsula o datagrama em um quadro de camada de enlace e transmite o quadro para dentro do enlace, e um nó receptor recebe o quadro e extraí o datagrama.

5.1.1 Os serviços fornecidos pela camada de enlace

Um protocolo de camada de enlace é usado para transportar um datagrama por um enlace individual. O **protocolo de camada de enlace** define o formato dos pacotes trocados entre os nós nas extremidades do enlace, bem como as ações realizadas por esses nós ao enviar e receber os pacotes. Lembre-se de que no Capítulo 1 dissemos que as unidades de dados trocadas pelo protocolo de camada de enlace são denominadas **quadros** e que cada quadro de camada de enlace caracteristicamente encapsula um datagrama de camada de rede. Como veremos adiante, entre as ações realizadas por um protocolo de camada de enlace ao enviar e receber quadros, estão

d detecção de erros, retransmissão, controle de fluxo e acesso aleatório. Como exemplos de protocolos de camada de enlace temos os protocolos Ethernet, 802.11 para LANs sem fio (também conhecidas como Wi-Fi), token ring e PPP. Analisaremos muitos desses protocolos detalhadamente na segunda metade deste capítulo.

Enquanto a camada de rede tem como tarefa movimentar segmentos da camada de transporte fim a fim, desde o hospedeiro de origem até o hospedeiro de destino, um protocolo de camada de enlace é encarregado de movimentar datagramas de camada de rede nó a nó por um único *enlace* no caminho. Uma característica importante da camada de enlace é que um datagrama pode ser transportado por diferentes protocolos de enlace nos diferentes enlaces no caminho. Por exemplo, um datagrama pode ser transportado pelo protocolo Ethernet no primeiro enlace, pelo PPP no último enlace e por um protocolo WAN de camada de enlace nos enlaces intermediários. É importante notar que os serviços fornecidos pelos protocolos de camada de enlace podem ser diferentes. Por exemplo, um protocolo de camada de enlace pode prover ou não entrega confiável. Assim, a camada de rede deve ser capaz de realizar sua tarefa fim a fim em face de um conjunto heterogêneo de serviços individuais de camada de enlace.

Para uma boa compreensão da camada de enlace e de como ela se relaciona com a camada de rede, vamos considerar uma analogia com um sistema de transporte. Considere um agente de viagens que está fazendo o planejamento de uma viagem para um turista desde Princeton, em Nova Jersey, até Lausanne, na Suíça. O agente de viagens decide que é mais conveniente para o turista pegar uma limusine de Princeton até o Aeroporto JFK, em seguida um avião do Aeroporto JFK até o Aeroporto de Genebra e, finalmente, um trem do Aeroporto de Genebra até a estação ferroviária de Lausanne. Assim que o agente fizer as três reservas, é responsabilidade da empresa de limusines levar o turista de Princeton ao Aeroporto JFK, é responsabilidade da companhia aérea levar o turista do Aeroporto JFK a Genebra, e é responsabilidade do trem suíço levar o turista de Genebra a Lausanne. Cada um dos três segmentos da viagem é ‘direto’ entre duas localidades ‘adjacentes’. Note que os três segmentos de transporte são administrados por empresas diferentes e usam meios de transporte completamente diferentes (limusine, avião e trem). Embora os meios de transporte sejam diferentes, cada um deles fornece o serviço básico de levar passageiros de uma localidade a outra localidade adjacente. Nessa comparação com o transporte, o turista é análogo a um datagrama, cada segmento de transporte é análogo a um enlace de comunicação, o meio de transporte é análogo a um protocolo de camada de enlace e o agente de viagens é análogo a um protocolo de roteamento.

Embora o serviço básico de qualquer camada de enlace seja mover um datagrama de um nó até um nó adjacente por um único enlace de comunicação, os detalhes do serviço podem variar de um protocolo de camada de enlace para outro. Entre os possíveis serviços que podem ser oferecidos por um protocolo de camada de enlace, estão:

Enquadramento de dados. Quase todos os protocolos de camada de enlace encapsulam cada datagrama de camada de rede dentro de um quadro de camada de enlace antes de transmiti-lo pelo enlace. Um quadro consiste em um campo de dados no qual o datagrama da camada de rede é inserido e em uma série de campos de cabeçalho. (Um quadro pode incluir também campos de trailer [campos no final do quadro], contudo, vamos nos referir também aos campos de trailer como campos de cabeçalho.) A estrutura do quadro é especificada pelo protocolo de camada de enlace. Veremos diversos formatos de quadros diferentes quando examinarmos os protocolos de camada de enlace específicos na segunda metade deste capítulo.

Acesso ao enlace. Um protocolo de controle de acesso ao meio (*medium access control protocol* — MAC) especifica as regras segundo as quais um quadro é transmitido pelo enlace. Para enlaces ponto a ponto que têm um único remetente em uma extremidade do enlace e um único receptor na outra extremidade, o protocolo MAC é simples (ou inexistente) — o remetente pode enviar um quadro sempre que o enlace estiver ocioso. O caso mais interessante é quando vários nós compartilham um único enlace de broadcast — o denominado problema de acesso múltiplo. Aqui, o protocolo MAC serve para coordenar as transmissões de quadros dos muitos nós; estudaremos protocolos MAC detalhadamente na Seção 5.3.

Entrega confiável. Quando um protocolo de camada de enlace fornece serviço confiável de entrega, ele garante que vai transportar cada datagrama da camada de rede pelo enlace sem erro. Lembre-se de que certos protocolos de camada de transporte (como o TCP) também fornecem um serviço confiável de entrega. Semelhante ao que acontece com um serviço confiável de entrega de camada de transporte,

consegue-se um serviço confiável de entrega de camada de enlace com reconhecimentos e retransmissões (veja a Seção 3.4). Um serviço confiável de entrega de camada de enlace é muito usado por enlaces que costumam ter altas taxas de erros, como é o caso de um enlace sem fio, com a finalidade de corrigir um erro localmente, no enlace no qual o erro ocorre, em vez de forçar uma retransmissão fim a fim dos dados por um protocolo de camada de transporte ou de aplicação. Contudo, a entrega confiável de camada de enlace pode ser considerada uma sobrecarga desnecessária para enlaces de baixa taxa de erros, incluindo enlaces de fibra, enlaces coaxiais e muitos enlaces de pares de fios trançados de cobre. Por essa razão, muitos protocolos de camada de enlace com fio não fornecem um serviço de entrega confiável.

Controle de fluxo. Os nós de cada lado de um enlace têm uma capacidade limitada de armazenar quadros. Este é um problema potencial, pois um nó receptor pode receber quadros a uma velocidade maior do que sua capacidade de processá-los. Sem controle de fluxo, o buffer do receptor pode transbordar e quadros podem ser perdidos. Semelhante à camada de transporte, um protocolo de camada de enlace pode fornecer controle de fluxo para evitar que o nó remetente de um lado de um enlace congestione o nó receptor do outro lado do enlace.

Detecção de erros. O hardware da camada de enlace de um nó receptor pode decidir incorretamente que um bit de um quadro é 0 quando foi transmitido como 1 e vice-versa. Esses erros de bits são introduzidos por atenuação de sinal e ruído eletromagnético. Como não há necessidade de repassar um datagrama que tem um erro, muitos protocolos de camada de enlace oferecem um mecanismo para detectar a presença de tais erros. Isso é feito obrigando o nó transmissor a enviar bits de detecção de erros no quadro e obrigando o nó receptor a realizar uma verificação de erros. Lembre-se de que dissemos nos capítulos 3 e 4 que as camadas de transporte e de rede da Internet também fornecem um serviço limitado de detecção de erros — a soma de verificação da Internet. A detecção de erros na camada de enlace geralmente é mais sofisticada e é implementada em hardware.

Correção de erros. A correção de erros é semelhante à detecção de erros, exceto que um receptor não somente detecta quando ocorreram os erros no quadro, mas também determina exatamente em que lugar do quadro os erros ocorreram (e, então, os corrige). Alguns protocolos fornecem correção de erros na camada de enlace apenas para o cabeçalho do pacote, e não para o pacote inteiro. Examinaremos a detecção e a correção de erros na Seção 5.2.

Half-duplex e full-duplex. Com transmissão *full-duplex*, os nós em ambas as extremidades de um enlace podem transmitir pacotes ao mesmo tempo. Com transmissão *half-duplex* um nó não pode transmitir e receber pacotes ao mesmo tempo.

Como dissemos antes, muitos serviços fornecidos pela camada de enlace apresentam grandes paralelos com os serviços fornecidos na camada de transporte. Por exemplo, tanto a camada de enlace como a de transporte podem fornecer entrega confiável. Embora os mecanismos usados para prover entrega confiável nas duas camadas sejam semelhantes (veja a Seção 3.4), os dois serviços de entrega confiável não são iguais. Um protocolo de transporte fornece entrega confiável de segmentos entre dois processos que operam fim a fim na rede; um protocolo de enlace confiável fornece serviço de entrega confiável entre dois nós ligados por um único enlace. De modo semelhante, tanto os protocolos de camada de enlace como os de camada de transporte podem fornecer controle de fluxo e detecção de erros; novamente, o controle de fluxo em um protocolo de camada de transporte é fornecido fim a fim, enquanto em um protocolo de camada de enlace é fornecido entre nós adjacentes.

5.1.2 Onde a camada de enlace é implementada?

Antes de mergulharmos em nosso detalhado estudo sobre a camada de enlace, vamos considerar a questão de *onde* a camada de enlace é implementada. Daqui em diante focaremos em um sistema final, visto que já aprendemos no Capítulo 4 como a camada de enlace é implementada em uma placa de linha de um roteador. A camada de enlace de um computador deve ser implementada em um software ou em um hardware? Deve ser implementada em uma placa ou chip separado e como ocorre a interface com o resto do hardware de um hospedeiro e com os componentes de sistemas operacionais?

A Figura 5.2 mostra a arquitetura de um típico hospedeiro. Na maior parte, a camada de enlace é implementada em um **adaptador de rede**, por vezes também conhecido como **controlador de interface de rede (NIC)**. No núcleo do adaptador de rede está o controlador da camada de enlace, normalmente um único chip de sistema especial, que implementa vários serviços da camada de enlace (enquadramento, acesso ao enlace, controle de fluxo, reconhecimento de erros etc.) como visto na sessão anterior. Dessa forma muito da funcionalidade do controlador da camada de enlace é implementado em hardware. Por exemplo, o controlador da Intel 8254x [Intel 2009] implementa os protocolos Ethernet, os quais estudaremos na seção 5.5; o controlador Atheros AR5006 [Atheros 2009] implementa os protocolos Wi Fi 802.11 que estudaremos na seção 6.3. Até o final dos anos 90, a maioria dos adaptadores de rede eram placas fisicamente separadas (como a placa PCM CIA ou uma placa plug-in que se encaixa em um slot para cartão PCI de um computador), agora cada vez mais adaptadores de rede estão sendo integrados na placa-mãe do hospedeiro — uma configuração chamada LAN-on-motherboard.

No lado transmissor, o controlador separa um datagrama que foi criado e o armazenado na memória do hospedeiro por camadas mais altas da pilha de protocolos, encapsula o datagrama em uma camada de enlace (preenchendo os vários campos do quadro), e então transmite o quadro para um enlace de comunicação, seguindo o protocolo de acesso ao enlace. No lado receptor, um controlador recebe todo o quadro e extrai o datagrama da camada de rede. Se a camada de enlace efetuar uma verificação de erros, é o controlador transmissor que estabelece os bits de detecção de erros no cabeçalho de quadro e é o controlador receptor que executa a verificação de erros. Se a camada de enlace tiver seu fluxo controlado, os controles de transmissão e de recepção trocam informações de controle de fluxo para que o controlador de envio envie quadros em uma velocidade que o receptor possa lidar.

A Figura 5.2 mostra um adaptador de rede conectado ao barramento do computador (por exemplo, PCI ou barramento PCI-X), que se parece muito com qualquer outro dispositivo de entrada/saída dos outros componentes do computador. A Figura 5.2 mostra também que, enquanto a maior parte da camada de enlace é implementada em hardware no cartão de interface, parte da camada de enlace é implementada em software que é executada na CPU do hospedeiro. Os componentes do software da camada de enlace tipicamente implementam uma funcionalidade camada de enlace de um nível maior, como receber um datagrama da camada de rede, monitorando informações de endereçamento da camada de enlace e ativando o controle de hardware. No lado receptor, o software da camada de enlace responde a interrupções do controlador (por exemplo, devido ao recebimento de um ou mais quadros), lida com condições de erro e passa o datagrama para a camada de rede. Assim, a camada de enlace é uma combinação de hardware e software — o lugar na pilha de protocolos, onde software encontra o

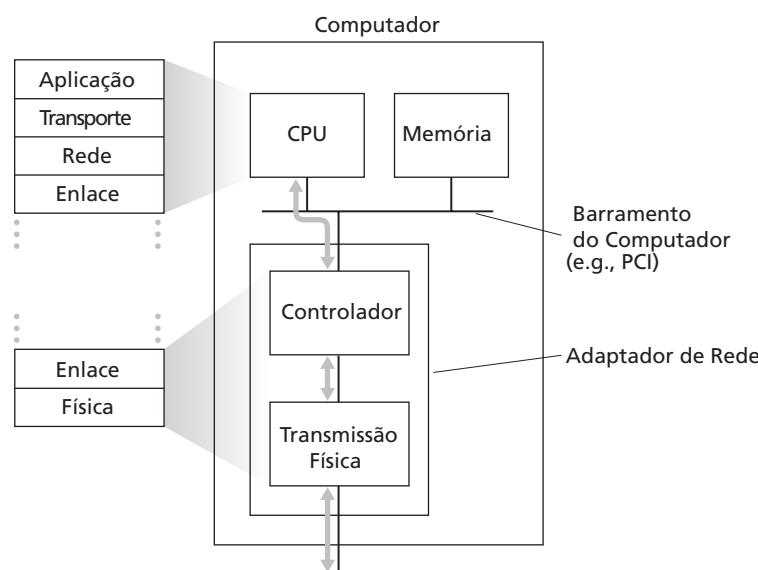


Figura 5.2 Adaptador de rede: seu relacionamento com o resto dos componentes do hospedeiro e a funcionalidade da pilha de protocolos

hardware. [Intel 2009] fornece um panorama legível (assim como descrições detalhadas) do controlador 8254x do ponto de vista de uma programação de software.

A Figura 5.3 mostra os adaptadores transmissores e receptores. Com a funcionalidade principal da camada de enlace sendo implementada pelo controlador, os adaptadores são unidades semiautônomas, sendo seu trabalho transferir um quadro de um adaptador ao outro. Vários pesquisadores já investigaram a possibilidade de acrescentar mais funcionalidade (além do processamento de camada de enlace) aos adaptadores de rede. O controlador 8254x, por exemplo, pode calcular a soma de verificação TCP/UDP e a soma de verificação do cabeçalho IP em hardware. É a funcionalidade da camada de rede da camada de transporte sendo implementada pelo controlador da camada de enlace. Apesar de parecer uma violação ultrajante do princípio de disposição em camadas, a vantagem é que as somas de verificação podem ser calculadas com mais rapidez do que em software, até o ponto que alguns podem ficar tentados a ignorar essa violação de princípios. [Mogul 2003] fornece uma discussão interessante dos prós e contras do desempenho de processar o TCP em um adaptador. [Kim 2005] investiga o desempenho de uma funcionalidade de maior nível (HTTP em cache) no adaptador.

5.2 Técnicas de detecção e correção de erros

Na seção anterior, observamos que **detecção e correção de erros no nível de bits** — detecção e correção da corrupção de bits em um quadro de camada de enlace enviado de um nó para outro nó vizinho fisicamente ligado a ele — são dois serviços frequentemente fornecidos pela camada de enlace. Vimos no Capítulo 3 que serviços de detecção e correção de erros também são frequentemente oferecidos na camada de transporte. Nesta seção, examinaremos algumas das técnicas mais simples que podem ser usadas para detectar e, em alguns casos, corrigir esses erros de bits. Como esse assunto — em especial, a teoria e a implementação dessas técnicas — é tratado detalhadamente por muitos autores (como [Schwartz, 1980] ou [Bertsekas, 1991]), nossa abordagem será necessariamente breve. Nossa meta é desenvolver uma visão intuitiva das capacidades que as técnicas de detecção e correção de erros fornecem e ver como algumas técnicas simples funcionam e são usadas na camada de enlace.

A Figura 5.4 ilustra o cenário de nosso estudo. No nó remetente, para que os dados, D , fiquem protegidos contra erros de bits, eles são aumentados com bits de detecção e de correção (*error detection-and-correction bits* — EDC). Em geral, os dados que devem ser protegidos incluem não somente o datagrama passado para baixo a partir da camada de rede para transmissão pelo enlace, mas também informações de endereçamento de camada de enlace, números de sequência e outros campos do cabeçalho do quadro de enlace. Tanto D como EDC são enviados ao nó receptor em um quadro de enlace. No nó receptor, são recebidas sequências D' e EDC' . Note que D' e EDC' podem ser diferentes dos D e EDC originais, como resultado de alterações nos bits em trânsito.

O desafio do receptor é determinar se D' é ou não igual ao D original, uma vez que recebeu apenas D' e EDC' . A exata sintaxe da decisão do receptor na Figura 5.4 (perguntamos se um erro foi detectado, e não se um erro foi cometido!) é importante. Técnicas de detecção e correção de erros permitem que o receptor detecte a ocorrência de erros de bits às vezes, mas não sempre. Mesmo com a utilização de bits de detecção de erros, ainda há a possibilidade de ocorrência de **erros de bits não detectados**, isto é, o receptor pode não perceber que a informação recebida contém erros de bits. Em consequência, o receptor poderá entregar um datagrama corrompido à camada de rede ou não perceber o conteúdo de um campo no cabeçalho do quadro foi corrompido. Assim, é preciso

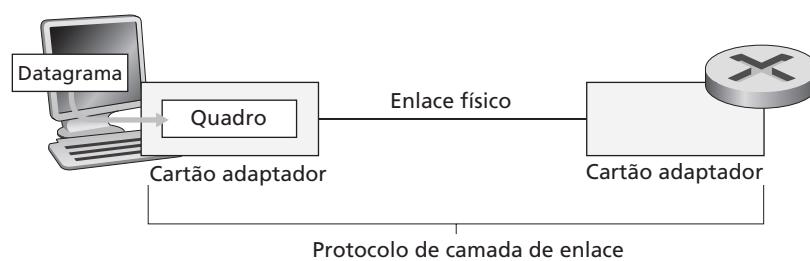


Figura 5.3 Adaptadores de rede em comunicação: um datagrama de uma camada de rede revestido por um quadro de camada de enlace

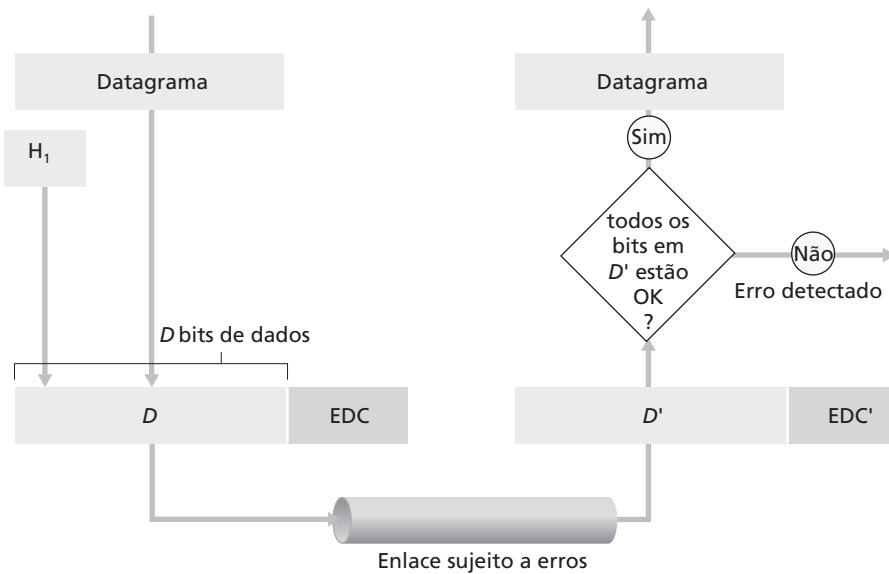


Figura 5.4 Cenário de detecção e correção de erros

escolher um esquema de detecção de erros para o qual a probabilidade de que essas ocorrências aconteçam seja pequena. Em geral, técnicas mais sofisticadas de detecção e correção de erros (isto é, as que têm uma probabilidade menor de permitir erros de bits não detectados) ficam sujeitas a uma sobrecarga maior — é preciso mais processamento para computar e transmitir um número maior de bits de detecção e correção de erros.

Vamos examinar agora três técnicas de detecção de erros nos dados transmitidos — verificações de paridade (para ilustrar as ideias básicas da detecção e correção de erros), métodos de soma de verificação (que são mais empregados na camada de transporte) e verificações de redundância cíclica (CRCs) (que são normalmente empregadas na camada de enlace, nos adaptadores).

5.2.1 Verificações de paridade

Talvez a maneira mais simples de detectar erros seja utilizar um único **bit de paridade**. Suponha que a informação a ser enviada, D na Figura 5.5, tenha d bits. Em um esquema de paridade par, o remetente simplesmente inclui um bit adicional e escolhe o valor desse bit de modo que o número total de '1' nos bits $d + 1$ (a informação original mais um bit de paridade) seja par. Em esquemas de paridade ímpar, o valor do bit de paridade é escondido de modo que haja um número ímpar de '1'. A Figura 5.5 ilustra um esquema de paridade par com o bit de paridade armazenado em um campo separado.

A operação do receptor também é simples com um único bit de paridade. O receptor precisa apenas contar quantos '1' há nos $d + 1$ bits recebidos. Se, utilizando um esquema de paridade par, for encontrado um número ímpar de bits de valor 1, o receptor saberá que ocorreu pelo menos um erro de bit. Mais precisamente, ele saberá que ocorreu algum número ímpar de erros de bit.

Mas o que acontecerá se ocorrer um número par de erros de bit? É bom que você se convença de que isso resultaria em um erro não detectado. Se a probabilidade de erro de bits for pequena e se for razoável admitir que os erros ocorrem independentemente entre um bit e o bit seguinte, a probabilidade de haver vários erros de bits em um pacote seria bastante pequena. Nesse caso, um único bit de paridade poderia ser suficiente. Contudo, medições demonstraram que, em vez de acontecerem independentemente, os erros frequentemente se aglomeram em 'rajadas'. Sob a condição de rajada de erros, a probabilidade de haver erros não detectados em um quadro protegido por um esquema de paridade de bit único pode se aproximar de 50 por cento [Spragins, 1991]. Claro que é necessário um esquema de detecção de erros mais robusto (e, felizmente, é o que se usa na prática!). Mas,

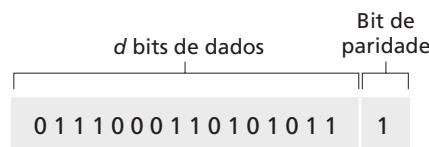


Figura 5.5 Paridade par com um bit de paridade

antes de examinarmos os esquemas de detecção de erros que são usados na prática, vamos considerar uma generalização simples da paridade de bit único que nos dará uma percepção das técnicas de correção de erros.

A Figura 5.6 mostra uma generalização bidimensional do esquema de paridade de bit único. Nessa figura, os d bits de D são divididos em i filas e j colunas. Um valor de paridade é calculado para cada fila e para cada coluna. Os $i + j + 1$ bits de paridade resultantes compreendem os bits de detecção de erros do quadro da camada de enlace.

Suponha agora que ocorra um erro de bit único nos d bits originais de informação. Com esse esquema de **paridade bidimensional**, tanto a paridade da coluna quanto a da fila que contiver o bit modificado estarão com erro. O receptor então não somente pode detectar que ocorreu um erro de um bit único, mas também usar os índices da fila e da coluna com erros de paridade para realmente identificar o bit que foi corrompido e *corrigir* aquele erro! A Figura 5.6 mostra um exemplo no qual o bit com valor 1 na posição (2, 2) está corrompido e mudou para um 0 — um erro que não somente é detectável, como também é corrigível no receptor. Embora nossa discussão tenha focalizado os d bits originais de informação, um erro único nos próprios bits de paridade também é detectável e corrigível. A paridade bidimensional também pode detectar (mas não corrigir!) qualquer combinação de dois erros em um pacote. Outras propriedades do esquema de paridade bidimensional são discutidas nos exercícios ao final deste capítulo.

A capacidade do receptor para detectar e corrigir erros é conhecida como **correção de erros de repasse** (*forward error correction* — FEC). Essas técnicas são comumente usadas na armazenagem de áudio e em equi-

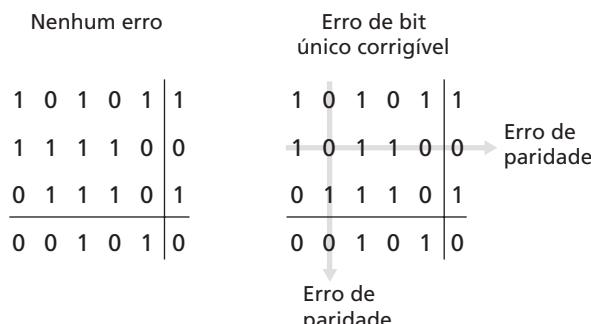
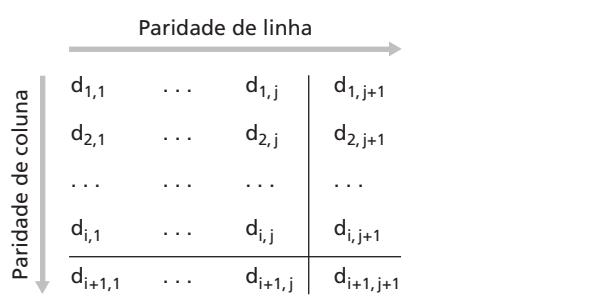


Figura 5.6 Paridade par bidimensional

pamentos de reprodução, como CDs de áudio. Em um ambiente de rede, as técnicas FEC podem ser usadas isoladamente ou em conjunto com as técnicas ARQ, que examinamos no Capítulo 3. As técnicas FEC são valiosas porque podem reduzir o número requerido de retransmissões do remetente. Talvez o mais importante seja que elas permitem imediata correção de erros no receptor. Isso evita ter de esperar pelo atraso de propagação da viagem de ida e volta que o remetente precisa para receber um pacote NAK e para que um pacote retransmitido se propague de volta ao receptor — uma vantagem potencialmente importante para aplicações de rede em tempo real [Rubenstein, 1998]. Pesquisas que examinaram a utilização da FEC em protocolos de controle de erros estão os de [Biersack, 1992; Nonnenmacher, 1998; Byers, 1998; Shacham, 1990].

5.2.2 Métodos de soma de verificação

Em técnicas de soma de verificação, os d bits de dados na Figura 5.5 são tratados como uma sequência de números inteiros de k bits. Um método simples de soma de verificação é somar esses inteiros de k bits e usar o total resultante como bits de detecção de erros. A **soma de verificação da Internet** é baseada nessa abordagem — bytes de dados são tratados como inteiros de 16 bits e somados. O complemento de 1 dessa soma então forma a soma de verificação da Internet que é carregada no cabeçalho do segmento. Como discutido na Seção 3.3, o receptor verifica a soma de verificação calculando os complementos de 1 da soma dos dados recebidos (inclusive a soma de verificação) e verificando se o resultado contém somente bits 1. Se qualquer um dos bits for 0, isso indicará um erro. O RFC 1071 discute detalhadamente o algoritmo da soma de verificação da Internet e sua implementação. Nos protocolos TCP/IP, a soma de verificação da Internet é calculada em todos os campos (incluindo os campos de cabeçalho e de dados). Em IP soma de verificação é calculada sobre o cabeçalho do IP (já que os segmentos UDP ou TCP têm sua própria soma de verificação). Em outros protocolos, o XTP, por exemplo [Strayer, 1992], uma soma de verificação é calculada para o cabeçalho e outra soma de verificação é calculada para o pacote inteiro.

Métodos de soma de verificação exigem relativamente pouca sobrecarga de pacote. Por exemplo, as somas de verificação em TCP e UDP utilizam apenas 16 bits. Contudo, oferecem proteção relativamente baixa contra erros em comparação com a verificação de redundância cíclica, discutida mais adiante, que é utilizada com frequência na camada de enlace. Uma pergunta que surge naturalmente neste ponto é: por que a soma de verificação é utilizada na camada de transporte e a verificação de redundância cíclica é utilizada na camada de enlace? Lembre-se de que a camada de transporte é normalmente implementada em software como parte do sistema operacional de um hospedeiro. Como a detecção de erros na camada de transporte é implementada em software, é importante que o esquema de detecção de erros seja simples e rápido como a soma de verificação. Por outro lado, a detecção de erro na camada de enlace é implementada em hardware dedicado, em adaptadores que podem rodar velocemente as mais complexas operações de CRC. Feldmeier [Feldmeier, 1995] apresenta técnicas de implementação rápida em software não somente para códigos de soma de verificação ponderada, mas também para CRC (veja a seguir) e outros códigos.

5.2.3 Verificação de redundância cíclica (CRC)

Uma técnica de detecção de erros usada amplamente nas redes de computadores de hoje é baseada em **códigos de verificação de redundância cíclica** (*cyclic redundancy check* — CRC). Códigos de CRC também são conhecidos como **códigos polinomiais**, já que é possível considerar a cadeia de bits a ser enviada como um polinômio cujos coeficientes são os valores 0 e 1 na cadeia de bits, sendo as operações na cadeia de bits interpretadas como aritmética polinomial.

Códigos de CRC funcionam como segue. Considere a parcela de d bits de dados, D , que o nó remetente quer enviar para o nó receptor. O remetente e o receptor devem, primeiramente, concordar com um padrão de $r + 1$ bits, conhecido como um **gerador**, que denominaremos G . Vamos exigir que o bit mais significativo (o da extrema esquerda) de G seja um 1. A ideia fundamental por trás dos códigos de CRC é mostrada na Figura 5.7. Para uma dada parcela de dados, D , o remetente escolherá r bits adicionais, R , e os anexará a D de modo que o

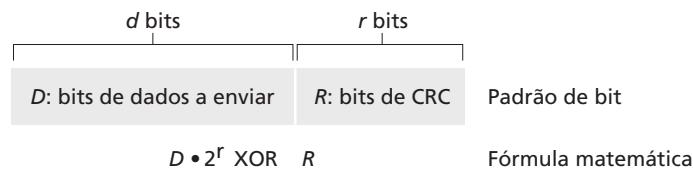


Figura 5.7 Códigos de CRC

padrão de $d + r$ bits resultante (interpretado como um número binário) seja divisível exatamente por G (por exemplo, sem nenhum remanescente), usando aritmética de módulo 2. Assim, o processo de verificação de erros com CRC é simples: o receptor divide os $d + r$ bits recebidos por G . Se o resto for diferente de zero, o receptor saberá que ocorreu um erro; caso contrário, os dados são aceitos como corretos.

Todos os cálculos de CRC são feitos por aritmética de módulo 2 sem ‘vai 1’ nas adições nem ‘empresta 1’ nas subtrações. Isso significa que a adição e a subtração são idênticas e ambas são equivalentes à operação *ou exclusivo* (XOR) dos bits dos operandos. Por exemplo,

$$\begin{array}{r} 1011 \text{ XOR } 0101 = 1110 \\ 1001 \text{ XOR } 1101 = 0100 \end{array}$$

Também de modo semelhante temos:

$$\begin{array}{r} 1011 - 0101 = 1110 \\ 1001 - 1101 = 0100 \end{array}$$

A multiplicação e a divisão são as mesmas da aritmética binária, exceto que, em qualquer adição ou subtração exigida, não se somam nem se emprestam casas. Como na aritmética binária comum, a multiplicação por 2^k desloca um padrão de bits para a esquerda por k casas. Assim, dados D e R , a quantidade $(D \cdot 2^r)$ XOR R dá como resultado o padrão de bit $d + r$ mostrado na Figura 5.7. Usaremos essa representação algébrica do padrão de bit $d + r$ da Figura 5.7 em nossa discussão a seguir.

Vamos agora voltar à questão crucial de como o remetente calcula R . Lembre-se de que queremos descobrir um R tal que exista um n que

$$(D \cdot 2^r) \text{ XOR } R = nG,$$

isto é, devemos escolher um R tal que G divida $(D \cdot 2^r)$ XOR sem resto. Se usarmos a operação *ou exclusivo*, isto é, adicionarmos binariamente, sem vai 1, em R em ambos os lados da equação anterior, teremos:

$$D \cdot 2^r = nG \text{ XOR } R.$$

Essa equação nos diz que, se dividirmos $D \cdot 2^r$ por G , o valor do resto será exatamente R . Em outras palavras, podemos calcular R como

$$R = \text{resto } \frac{D \cdot 2^r}{G}$$

A Figura 5.8 ilustra esses cálculos para o caso de $D = 101110$, $d = 6$ e $G = 1001$, $r = 3$. Os 9 bits transmitidos nesse caso são 101110 011. Você deve fazer esses cálculos e também verificar se, na verdade, $D \cdot 2^r = 101011 \cdot G$ XOR R .

Padrões internacionais foram definidos para geradores G de 8, 12, 16 e 32 bits. O padrão CRC 32 de 32 bits, que foi adotado em uma série de protocolos do IEEE da camada de enlace, usa um gerador igual a:

$$G_{\text{CRC-32}} = 100000100110000010001110110110111.$$

Cada um dos padrões de CRC pode detectar erros de rajada de menos do que $r + 1$ bits. (Isso significa que todos os erros de bits consecutivos de r bits ou menos serão detectados.) Além disso, sob hipóteses apropriadas, uma rajada de comprimento maior do que $r + 1$ bits é detectada com probabilidade de $1 - 0,5^r$. Cada um dos padrões de CRC também pode detectar qualquer número ímpar de erros de bits. A teoria por trás dos códigos de CRC e de códigos até mais poderosos ultrapassa o escopo deste livro. O livro de [Schwartz, 1980] oferece uma excelente introdução a esse tópico.

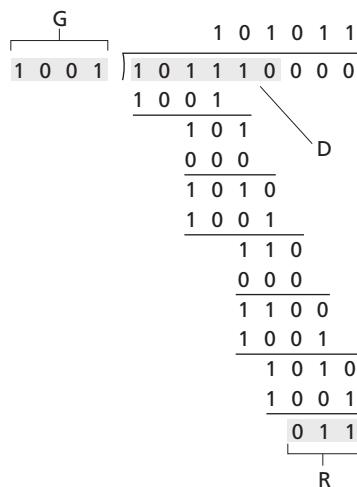


Figura 5.8 Um exemplo de cálculo de CRC

5.3 Protocolos de acesso múltiplo

Na introdução deste capítulo, observamos que há dois tipos de enlaces de redes: enlaces ponto a ponto e enlaces *broadcast*. Um **enlace ponto a ponto** consiste em um único remetente em uma extremidade do enlace e um único receptor na outra extremidade do enlace. Muitos protocolos de camada de enlace foram projetados para enlaces ponto a ponto; o PPP (protocolo ponto a ponto) e um controle de ligação de dados (HDCL) de alto nível são alguns protocolos que examinaremos mais adiante neste capítulo. O segundo tipo de enlace, o **enlace broadcast**, pode ter vários nós remetentes e receptores, todos conectados ao mesmo canal de transmissão único e compartilhado. O termo *broadcast* é usado aqui porque, quando qualquer um dos nós transmite um quadro, o canal propaga o quadro e cada um dos nós recebe uma cópia. A Ethernet e as LANs sem fio são exemplos de tecnologias de *broadcast* de camada de enlace. Nesta seção, vamos nos afastar um pouco dos protocolos específicos de camada de enlace e examinar, primeiramente, um problema de importância fundamental para a camada de enlace de dados: como coordenar o acesso de vários nós remetentes e receptores a um canal *broadcast* compartilhado — o **problema do acesso múltiplo**. Canais *broadcast* são muito usados em LANs, redes que estão geograficamente concentradas em um único edifício (ou empresa, ou campus universitário). Assim, no final da seção, examinaremos também como os canais de acesso múltiplo são usados em LANs.

Todos estamos familiarizados com a noção de *broadcast* — a televisão usa essa tecnologia desde que foi inventada. Mas a televisão tradicional é um *broadcast* unidirecional (isto é, um nó fixo que transmite para muitos nós receptores), ao passo que os nós em um canal *broadcast* de uma rede de computadores tanto podem enviar quanto receber. Talvez uma analogia humana mais apropriada para um canal *broadcast* seja um coquetel, no qual muitas pessoas se encontram em uma sala grande para falar e ouvir (e o ar fornece o meio de transmissão). Uma outra boa analogia é algo com que muitos leitores estão familiarizados — uma sala de aula, onde professor(es) e estudante(s) compartilham, de maneira semelhante, o mesmo e único meio de transmissão *broadcast*. Um problema fundamental em ambos os cenários é determinar quem fala (isto é, quem transmite pelo canal) e quando fala. Como seres humanos, desenvolvemos uma série elaborada de protocolos para compartilhar o canal *broadcast*:

“Dê a todos uma oportunidade de falar.”

“Não fale até que alguém fale com você.”

“Não monopolize a conversa.”

“Levante a mão se tiver uma pergunta a fazer.”

“Não interrompa uma pessoa quando ela estiver falando.”

“Não durma quando alguém estiver falando.”

Redes de computadores têm protocolos semelhantes — denominados **protocolos de acesso múltiplo** —, pelos quais os nós regulam sua transmissão pelos canais *broadcast* compartilhados. Como mostra a Figura 5.9, os protocolos de acesso múltiplo são necessários em uma ampla variedade de cenários de rede, que inclui redes locais com fio e sem fio e redes por satélite. Embora tecnicamente cada nó acesse o canal *broadcast* por meio de seu adaptador, nesta seção vamos nos referir ao nó como o dispositivo de envio e de recepção. Na prática, centenas ou até milhares de nós podem se comunicar diretamente por um canal *broadcast*.

Como todos os nós têm a capacidade de transmitir quadros, mais do que dois nós podem transmitir quadros ao mesmo tempo. Quando isso acontece, todos os nós recebem vários quadros ao mesmo tempo, isto é, os quadros transmitidos **colidem** em todos os receptores. Em geral, quando há uma colisão, nenhum dos nós receptores consegue perceber algum sentido nos quadros que foram transmitidos; de certo modo, os sinais dos quadros que colidem ficam inextricavelmente embaralhados. Assim, todos os quadros envolvidos na colisão são perdidos e o canal *broadcast* é desperdiçado durante o intervalo de colisão. É claro que, se muitos nós querem transmitir quadros frequentemente, muitas transmissões resultarão em colisões e grande parte da largura de banda do canal *broadcast* será desperdiçada.

Para assegurar que o canal *broadcast* realize trabalho útil quando há vários nós ativos, é preciso coordenar, de algum modo, as transmissões desses nós ativos. Essa tarefa de coordenação é de responsabilidade do protocolo de acesso múltiplo. Durante os últimos 40 anos, milhares de artigos e centenas de teses foram escritos sobre protocolos de acesso múltiplo; um levantamento abrangente dos primeiros 20 anos desse volume de trabalho pode ser encontrado em [Rom, 1990]. Além disso, a pesquisa ativa sobre protocolos de acesso múltiplo continua devido ao surgimento contínuo de novos tipos de enlaces, particularmente novos enlaces sem fio.

Durante anos, dezenas de protocolos de acesso múltiplo foram implementados em uma variedade de tecnologias de camada de enlace. Não obstante, podemos classificar praticamente qualquer protocolo de acesso múltiplo em uma das seguintes categorias: **protocolos de divisão de canal**, **protocolos de acesso aleatório** e **protocolos de revezamento**. Examinaremos essas categorias de protocolos de acesso múltiplo nas três subseções seguintes.

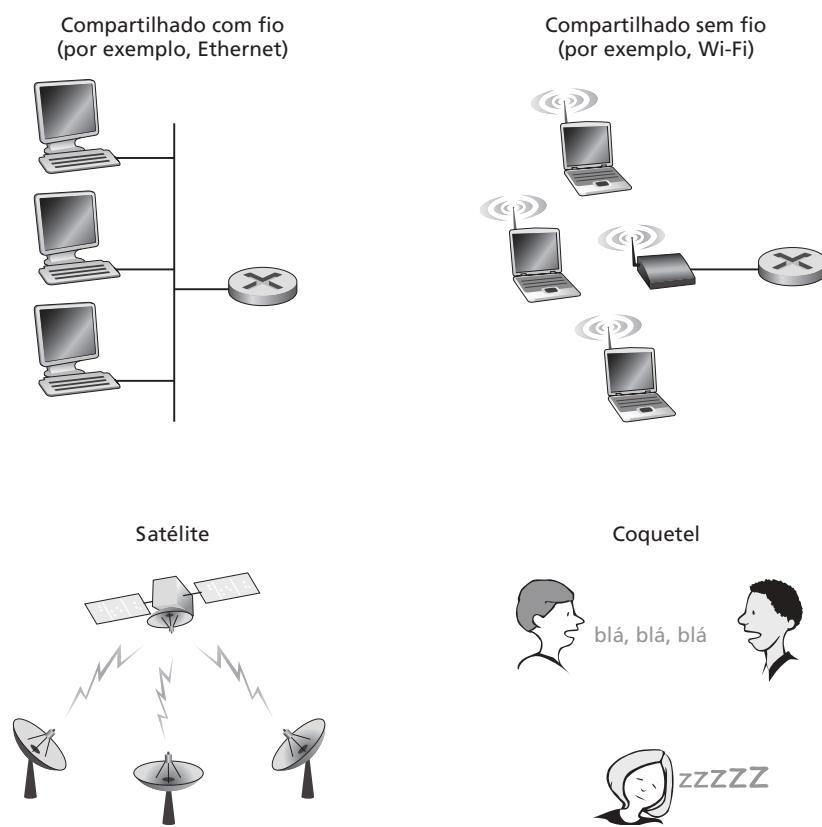


Figura 5.9 Vários canais de acesso múltiplo

Vamos concluir essa visão geral mencionando que, idealmente, um protocolo de acesso múltiplo para um canal *broadcast* com velocidade de R bits por segundo tem as seguintes características desejáveis:

1. Quando apenas um nó tem dados para enviar, esse nó tem uma vazão de R bps.
2. Quando M nós têm dados para enviar, cada um desses nós tem uma vazão de R/M bps. Isso não significa necessariamente que cada um dos M nós sempre terá uma velocidade instantânea de R/M , mas que cada nó deverá ter uma velocidade média de transmissão de R/M durante algum intervalo de tempo adequadamente definido.
3. O protocolo é descentralizado, isto é, não há nós mestres que possam falhar e derrubar o sistema inteiro.
4. O protocolo é simples para que sua implementação seja barata.

5.3.1 Protocolos de divisão de canal

Lembre-se de que na Seção 1.3 dissemos que a multiplexação por divisão de tempo (TDM) e a multiplexação por divisão de frequência (FDM) são duas técnicas que podem ser usadas para dividir a largura de banda de um canal *broadcast* entre todos os nós que compartilham esse canal. Como exemplo, suponha que o canal suporte N nós e que a velocidade de transmissão do canal seja R bps. O protocolo TDM divide o tempo em **quadros temporais** e depois divide cada quadro temporal em N **compartimentos**. (O quadro temporal TDM não deve ser confundido com a unidade de dados de camada de enlace trocada entre adaptadores remetentes e receptores, que também é denominada um quadro. Para reduzir a confusão, nesta seção vamos nos referir à unidade de dados trocada na camada de enlace como um pacote.) Cada compartimento é, então, atribuído a um dos N nós. Sempre que um nó tiver um pacote para enviar, ele transmite os bits do pacote durante o compartimento atribuído a ele no quadro temporal rotativo TDM. Normalmente, os tamanhos dos quadros são escolhidos de modo que um único quadro possa ser transmitido durante um compartimento. A Figura 5.10 mostra um exemplo de TDM simples de quatro nós.

Voltando à analogia do coquetel, um coquetel regulamentado por TDM permitiria que um dos convidados falasse durante um período de tempo fixo; em seguida, permitiria que um outro convidado falasse pelo mesmo período de tempo e assim por diante. Quando todos tivessem tido sua chance de falar, o modelo seria repetido.

O protocolo TDM é atraente, pois elimina colisões e é perfeitamente justo: cada nó ganha uma velocidade de transmissão dedicada de R/N bps durante cada quadro temporal. Contudo, ele tem duas desvantagens importantes. A primeira desvantagem é que um nó fica limitado a uma velocidade média de R/N bps, mesmo quando ele é o único nó com pacotes para enviar. A segunda desvantagem é que o nó deve sempre esperar sua vez na sequência de transmissão — de novo, mesmo quando ele é o único nó com um quadro a enviar. Imagine um convidado que é o único que tem algo a dizer (e imagine uma situação ainda mais rara, em que todos na festa querem ouvir o que aquela pessoa tem a dizer). É óbvio que o protocolo TDM seria uma má escolha para um protocolo de acesso múltiplo para essa festa em particular.

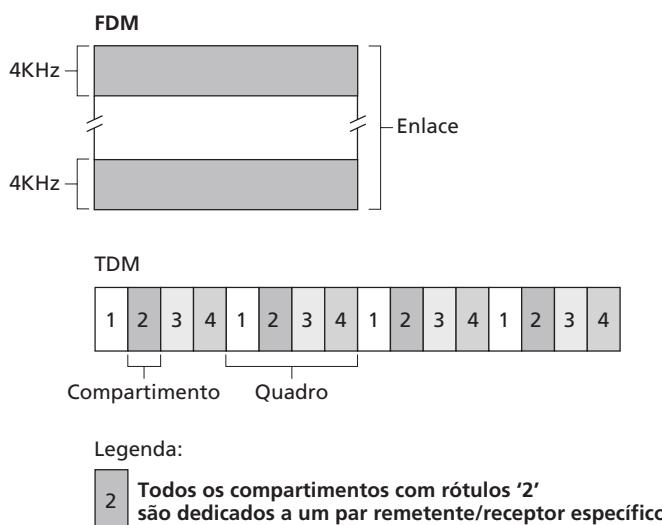


Figura 5.10 Um exemplo de TDM e FDM de quatro nós



Enquanto o protocolo TDM compartilha o canal *broadcast* no tempo, o protocolo FDM divide o canal de R bps em frequências diferentes (cada uma com uma largura de banda de R/N) e reserva cada frequência a um dos N nós, criando, desse modo, N canais menores de R/N bps a partir de um único canal maior de R bps. O protocolo FDM compartilha as vantagens do protocolo TDM — evita colisões e divide a largura de banda com justiça entre os N nós. Porém, também compartilha uma desvantagem principal com o protocolo TDM — um nó é limitado a uma largura de banda R/N , mesmo quando é o único nó que tem pacotes a enviar.

Um terceiro protocolo de divisão de canal é o protocolo de **acesso múltiplo por divisão de código** (*code division multiple access* — CDMA). Enquanto os protocolos TDM e FDM atribuem aos nós intervalos de tempo e frequências, respectivamente, o protocolo CDMA atribui um *código* diferente a cada nó. Então, cada nó usa seu código exclusivo para codificar os bits de dados que envia. Se os códigos forem escolhidos cuidadosamente, as redes CDMA terão a maravilhosa propriedade de permitir que nós diferentes transmitam simultaneamente e, ainda assim, consigam que seus receptores respectivos recebam corretamente os bits codificados pelo remetente (admitindo-se que o receptor conheça o código do remetente), a despeito das interferências causadas pelas transmissões dos outros nós. O CDMA vem sendo usado em sistemas militares há algum tempo (devido às suas propriedades antiinterferências) e agora está começando a se difundir para uso civil, em particular para canais de acesso múltiplo sem fio. Como a utilização do CDMA está muito ligada a canais sem fio, adiaremos a discussão de seus detalhes técnicos para o Capítulo 6. Por enquanto, basta saber que códigos CDMA, assim como compartimentos de tempo em TDM e frequências em FDM, podem ser alocados a usuários de canais de múltiplo acesso.

5.3.2 Protocolos de acesso aleatório

A segunda classe geral de protocolos de acesso múltiplo são os protocolos de acesso aleatório. Com um protocolo de acesso aleatório, um nó transmissor sempre transmite à taxa total do canal, isto é, R bps. Quando há uma colisão, cada nó envolvido nela retransmite repetidamente seu quadro (isto é, pacote) até que este passe sem colisão. Mas, quando um nó sofre uma colisão, ele nem sempre retransmite o quadro imediatamente. *Em vez disso, ele espera um tempo aleatório antes de retransmitir o quadro.* Cada nó envolvido em uma colisão escolhe atrasos aleatórios independentes. Como após uma colisão os tempos de atraso são escolhidos independentemente, é possível que um dos nós escolha um atraso suficientemente mais curto do que os atrasos dos outros nós em colisão e, portanto, consiga passar seu quadro discretamente para dentro do canal, sem colisão.

Há dezenas, se não centenas, de protocolos de acesso aleatório descritos na literatura [Rom, 1990; Bertsekas, 1991]. Nesta seção, descreveremos alguns dos mais usados — os protocolos ALOHA [Abramson, 1970; 1985] e os protocolos de acesso múltiplo com detecção de portadora (CSMA) [Kleinrock, 1975b]. Mais adiante, na Seção 5.5, examinaremos os detalhes da Ethernet [Metcalfe, 1976], uma variante popular e muito disseminada do protocolo CSMA.

Slotted Aloha

Vamos começar nosso estudo de protocolos de acesso aleatório com um dos mais simples desses protocolos, o slotted ALOHA. Em nossa descrição do slotted ALOHA, admitiremos o seguinte:

- Todos os quadros consistem em exatamente L bits.
- O tempo é dividido em intervalos de tamanho L/R segundos (isto é, um intervalo é igual ao tempo de transmissão de um quadro).
- Os nós começam a transmitir quadros somente no início dos intervalos.
- Os nós são sincronizados de modo que cada nó sabe onde os intervalos começam.
- Se dois ou mais nós colidirem em um intervalo, então todos os nós detectarão o evento de colisão antes do término do intervalo.

Seja p uma probabilidade, isto é, um número entre 0 e 1. O funcionamento do slotted ALOHA em cada nó é simples:

- Quando o nó tem um novo quadro para enviar, espera até o início do próximo intervalo e transmite o quadro inteiro no intervalo.
- Se não houver uma colisão, o nó transmitirá seu quadro com sucesso e, assim, não precisará considerar a retransmissão do quadro. (O nó pode preparar um novo quadro para transmitir, caso tenha algum.)

Se houver uma colisão, o nó a detectará antes do final do intervalo. Ele retransmitirá seu quadro em cada intervalo subsequente com probabilidade p até que o quadro seja transmitido sem colisão.

Por retransmissão com probabilidade p , queremos dizer que o nó efetivamente joga uma moeda viciada; coroa corresponde a ‘retransmitir’, o que ocorre com probabilidade p , enquanto cara corresponde a “pule o intervalo e jogue a moeda novamente no próximo intervalo”, o que ocorre com probabilidade $(1 - p)$. Todos os nós envolvidos na colisão jogam suas moedas independentemente.

Aparentemente o slotted ALOHA teria muitas vantagens. Diferentemente da partição de canal, esse protocolo permite que um único nó transmita continuamente à taxa total do canal, R , quando ele for o único nó ativo. (Diz-se que um nó é ativo quanto tem quadros a enviar.) O slotted ALOHA também é altamente descentralizado, porque cada nó detecta colisões e decide independentemente quando retransmitir. (No entanto, requer que os intervalos sejam sincronizados nos nós; em breve discutiremos uma versão sem intervalos do protocolo ALOHA [unslotted ALOHA], bem como protocolos CSMA — nenhum deles requer essa sincronização e, portanto, são totalmente descentralizados.) O slotted ALOHA é também um protocolo extremamente simples.

O slotted ALOHA funciona bem quando há somente um nó ativo, mas qual é sua eficiência quando há vários nós ativos? Nesse caso, há duas preocupações possíveis. A primeira, como mostra a Figura 5.11, é que, quando há vários nós ativos, uma certa fração dos intervalos terá colisões e, portanto, será ‘desperdiçada’. A segunda preocupação é que uma outra fração dos intervalos estará vazia porque todos os nós ativos evitarão transmitir como resultado da política probabilística de transmissão. Os únicos intervalos ‘não desperdiçados’ serão aqueles em que exatamente um nó transmite. Um intervalo em que exatamente um nó transmite é denominado um **intervalo bem-sucedido**. A **eficiência** de um protocolo de acesso múltiplo com intervalos é definida como a fração (calculada durante um longo tempo) de intervalos bem-sucedidos no caso em que há um grande número de nós ativos, cada qual tendo sempre um grande número de quadros a enviar. Note que, se não fosse usado nenhum tipo de controle de acesso e cada nó retransmitisse imediatamente após cada colisão, a eficiência seria zero. O slotted ALOHA claramente aumenta a eficiência para além de zero, mas em quanto?

Vamos agora esboçar a derivação da eficiência máxima do slotted ALOHA. Para manter a simplicidade dessa derivação, vamos modificar um pouco o protocolo e admitir que cada nó tenta transmitir um quadro em cada intervalo com probabilidade p . (Isto é, admitimos que cada nó sempre tenha um quadro para enviar e que cada nó transmita com probabilidade p tanto para um quadro novo como para um quadro que já sofreu uma colisão.) Suponha que haja N nós. Então, a probabilidade de que um dado intervalo seja um intervalo bem-sucedido é a probabilidade de que um dos nós transmita e os restantes $N - 1$ nós não transmitam. A probabilidade de que um dado nó transmita é p ; a probabilidade de que os nós restantes não transmitam é $(1 - p)^{N-1}$. Por conseguinte, a probabilidade de que um dado nó tenha sucesso é $p(1 - p)^{N-1}$. Como há N nós, a probabilidade de um nó arbitrário ter sucesso é $Np(1 - p)^{N-1}$.

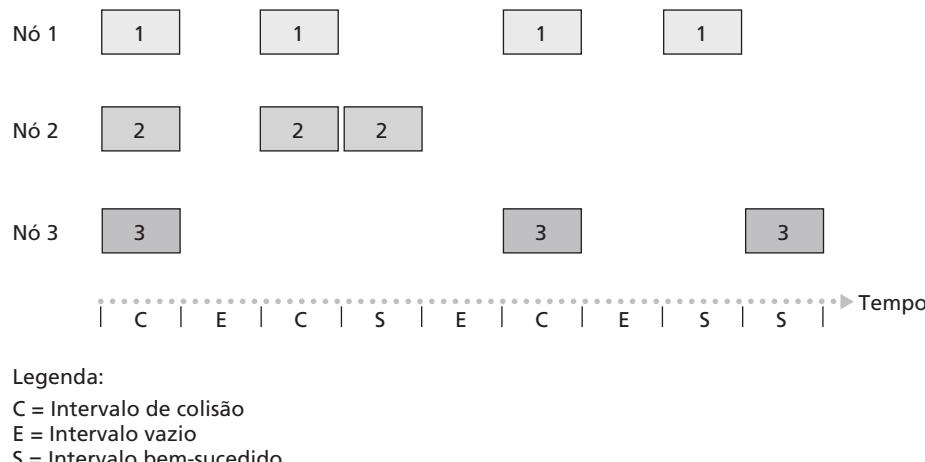


Figura 5.11 Nós 1, 2 e 3 colidem no primeiro intervalo. O nó 2 finalmente é bem-sucedido no quarto intervalo, o nó 1 no oitavo intervalo e o nó 3 no nono intervalo

Assim, quando há N nós ativos, a eficiência do slotted ALOHA é $Np(1 - p)^{N-1}$. Para obtermos a eficiência máxima para N nós ativos, temos de encontrar um p^* que maximize essa expressão. (Veja os exercícios ao final deste capítulo para um esboço geral dessa derivação.) E, para obtermos a eficiência máxima para um grande número de nós ativos, consideraremos o limite de $Np^*(1 - p^*)^{N-1}$ quando N tende ao infinito. (Novamente, veja os exercícios ao final deste capítulo.) Após esses cálculos, descobriremos que a eficiência máxima do protocolo é dada por $1/e = 0,37$. Isto é, quando um grande número de nós tem muitos quadros a transmitir, então (na melhor das hipóteses) apenas 37 por cento dos intervalos realiza um trabalho útil. Assim, a taxa efetiva de transmissão do canal não é R bps, mas apenas $0,37 R$ bps! Uma análise semelhante também demonstra que 37 por cento dos intervalos ficam vazios e 26 por cento apresentam colisões. Imagine o pobre administrador de rede que comprou um sistema slotted ALOHA de 100 Mbps esperando poder usar a rede para transmitir dados entre um grande número de usuários a uma taxa agregada de, digamos, 80 Mbps! Embora o canal tenha a capacidade de transmitir um dado quadro à taxa máxima do canal de 100 Mbps, no final, a vazão que se consegue com esse canal é de menos de 37 Mbps.

Aloha

O protocolo slotted ALOHA requer que todos os nós sincronizem suas transmissões para que comecem no início de um intervalo. O primeiro protocolo ALOHA [Abramson, 1970] era, na realidade, um protocolo sem intervalos e totalmente descentralizado. No ALOHA puro, quando um quadro chega pela primeira vez (isto é, um datagrama de camada de rede é passado para baixo a partir da camada de rede no nó remetente), o nó imediatamente transmite o quadro inteiro ao canal *broadcast*. Se um quadro transmitido sofrer uma colisão com uma ou mais transmissões, o nó retransmitirá imediatamente (após ter concluído a transmissão total do quadro que sofreu a colisão) o quadro com probabilidade p . Caso contrário, o nó esperará por um tempo de transmissão de quadro. Após essa espera, ele então retransmitirá o quadro com probabilidade p ou espera (permanecendo ocioso) por um outro tempo de quadro com probabilidade $1 - p$.

Para determinar a eficiência máxima do ALOHA puro, vamos focalizar um nó individual. Consideraremos as mesmas premissas que adotamos na análise do slotted ALOHA e tomaremos o tempo de transmissão do quadro como a unidade de tempo. A qualquer dado tempo, a probabilidade de que um nó esteja transmitindo um quadro é p . Suponha que esse quadro comece a transmitir no tempo t_0 . Como mostrado na Figura 5.12, para que esse quadro seja transmitido com sucesso, nenhum dos outros nós pode começar sua transmissão no intervalo de tempo $[t_0 - 1, t_0]$. Essa transmissão se sobreporia ao início da transmissão do quadro do nó i . A probabilidade de que todos os outros nós não iniciem uma transmissão nesse intervalo é $(1 - p)^{N-1}$. De maneira semelhante, nenhum outro nó pode iniciar uma transmissão enquanto o nó i estiver transmitindo, pois essa transmissão se sobreporia à parte final da transmissão do nó i . A probabilidade de que todos os outros nós não iniciem uma transmissão nesse intervalo é também $(1 - p)^{N-1}$. Assim, a probabilidade de que um dado nó tenha uma transmissão bem-sucedida é $p(1 - p)^{2(N-1)}$. Levando ao limite, como fizemos no caso do slotted ALOHA, descobrimos que a eficiência máxima do protocolo ALOHA puro é de apenas $1/(2e)$ — exatamente a metade da eficiência do slotted ALOHA. É este o preço que se paga por um protocolo ALOHA totalmente descentralizado.

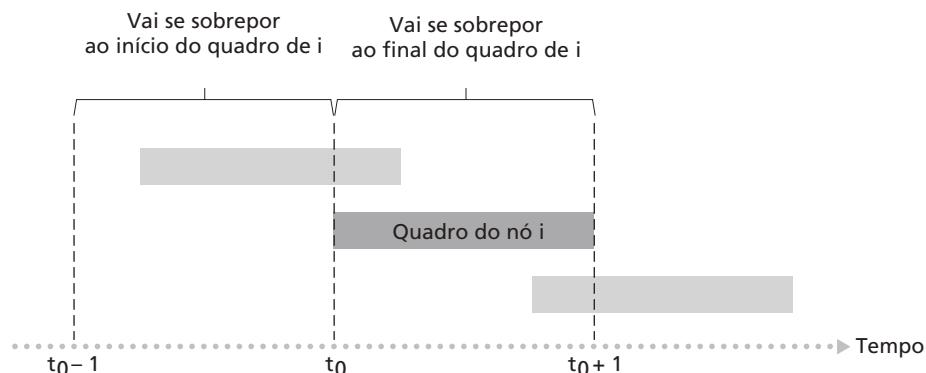


Figura 5.12 Transmissões interferentes no Aloha puro



História

Norm Abramson e a ALOHAnet

Norm Abramson, um doutor em engenharia, era apaixonado por surfe e interessado na comutação de pacotes. Essa combinação de interesses o levou à Universidade do Havaí em 1969. O Havaí é formado por muitas ilhas montanhosas, o que dificulta a instalação e a operação de redes terrestres. Quando não estava surfando, Abramson ficava pensando em como projetar uma rede que fizesse comutação de pacotes por rádio. A rede que ele projetou tinha um hospedeiro central e diversos nós secundários espalhados pelas ilhas havaianas. A rede tinha dois canais, cada um usando uma faixa de frequência diferente. O canal na direção dos nós secundários fazia broadcast de pacotes do hospedeiro central para os hospedeiros secundários; e o canal na direção contrária enviava pacotes dos hospedeiros secundários para o hospedeiro central. Além de enviar pacotes de informação, o hospedeiro central também enviava pelo canal na direção dos nós secundários um reconhecimento para cada pacote recebido com sucesso dos hospedeiros secundários.

Como os hospedeiros secundários transmitiam pacotes de maneira descentralizada, inevitavelmente ocorriam colisões no canal entre eles e o hospedeiro central. Essa observação levou Abramson a inventar, em 1970, o protocolo ALOHA puro, descrito neste capítulo. Em 1970, com o financiamento contínuo da ARPA, Abramson conectou sua ALOHAnet à ARPAnet. O trabalho de Abramson é importante não somente porque foi o primeiro exemplo de uma rede de pacotes por rádio, mas também porque inspirou Bob Metcalfe. Alguns anos depois, Metcalfe modificou o protocolo ALOHA e criou o protocolo CSMA/CD e a LAN Ethernet.

CSMA (Acesso múltiplo com detecção de portadora)

Tanto no slotted ALOHA quanto no ALOHA puro, a decisão de transmitir tomada por um nó independe da atividade dos outros nós ligados ao canal *broadcast*. Em particular, um nó não se preocupa se por acaso um outro nó está transmitindo quando ele começa a transmitir nem pára de transmitir se outro nó começar a interferir em sua transmissão. Em nossa analogia do coquetel, os protocolos ALOHA se parecem muito com um convidado mal-educado que continua a tagarelar mesmo quando outras pessoas estão falando. Como seres humanos, temos protocolos que nos levam não somente a nos comportar com mais civilidade, mas também a reduzir o tempo que gastamos ‘colidindo’ com outros durante a conversação e, consequentemente, a aumentar a quantidade de dados que trocamos durante nossas conversas. Especificamente, há duas regras importantes que regem a conversação educada entre seres humanos:

Ouça antes de falar. Se uma pessoa estiver falando, espere até que ela tenha terminado. No mundo das redes, isso é denominado **deteção de portadora** — um nó ouve o canal antes de transmitir. Se um quadro de outro nó estiver correntemente sendo transmitido para dentro do canal, o nó então esperará (se afastará — ‘back off’) por um período de tempo aleatório e, então, novamente sondará o canal. Se perceber que o canal está ocioso, o nó então começará a transmissão de quadros. Caso contrário, ele esperará por um outro período aleatório de tempo e repetirá esse processo.

Se alguém começar a falar ao mesmo tempo que você, pare de falar. No mundo das redes, isso é denominado **deteção de colisão** — um nó que está transmitindo ouve o canal enquanto transmite. Se esse nó detectar que outro nó está transmitindo um quadro interferente, ele para de transmitir e usa algum protocolo para determinar quando deve tentar transmitir novamente.

Essas duas regras estão incorporadas na família de protocolos CSMA (*carrier sense multiple access* — **acesso múltiplo com detecção de portadora**) e CSMA/CD (**CSMA com detecção de colisão**) [Kleinrock, 1975b; Metcalfe, 1976; Lam, 1980; Rom, 1990]. Foram propostas muitas variações do CSMA e do CSMA/CD. Para mais detalhes sobre esses protocolos, consulte as referências que acabamos de citar. Estudaremos detalhadamente o esquema CSMA/CD usado na Ethernet na Seção 5.5. Nesta seção, consideraremos algumas das características mais importantes e fundamentais do CSMA e do CSMA/CD.

A primeira pergunta que se poderia fazer sobre o CSMA é a seguinte: se todos os nós realizam detecção de portadora, por que ocorrem colisões? Afinal, um nó vai abster-se de transmitir sempre que perceber que um outro nó

está transmitindo. A resposta a essa pergunta pode ser ilustrada utilizando diagramas espaço/tempo [Molle, 1987]. A Figura 5.13 apresenta um diagrama espaço/tempo de quatro nós (A, B, C, D) ligados a um barramento linear de transmissão. O eixo horizontal mostra a posição de cada nó no espaço; o eixo vertical representa o tempo.

No tempo t_0 , o nó B percebe que o canal está ocioso, pois nenhum outro nó está transmitindo no momento. Assim, o nó B começa a transmitir e seus bits se propagam em ambas as direções ao longo do meio de transmissão. A propagação para baixo dos bits de B na Figura 5.13 com o aumento do tempo indica que é preciso uma quantidade de tempo de valor diferente de zero para que os bits de B realmente se propaguem (se bem que quase à velocidade da luz) ao longo do meio de transmissão. No tempo t_1 ($t_1 > t_0$), o nó D tem um quadro para enviar. Embora o nó B esteja transmitindo no tempo t_1 , os bits que estão sendo transmitidos por B ainda não alcançaram D. Assim, D percebe o canal como ocioso em t_1 . De acordo com o protocolo CSMA, D começa então a transmitir seu quadro. Pouco tempo depois, a transmissão de B passa a interferir na transmissão de D em D. Fica evidente na Figura 5.13 que o tempo de **atraso de propagação fim a fim de canal** para um canal broadcast — o tempo que leva para que um sinal se propague de um dos extremos do canal para outro — desempenhará um papel crucial na determinação de seu desempenho. Quanto mais longo for esse atraso de propagação, maior será a chance de um nó que detecta portadora ainda não poder perceber uma transmissão que já começou em outro nó da rede.

Na Figura 5.13, os nós não realizam detecção de colisão; ambos, B e D, continuam a transmitir seus quadros integralmente mesmo que ocorra uma colisão. Quando um nó realiza detecção de colisão, ele cessa a transmissão assim que detecta uma colisão. A Figura 5.14 mostra o mesmo cenário da Figura 5.13, exceto que cada um dos dois nós aborta sua transmissão pouco tempo após detectar uma colisão. Claramente, adicionar detecção de colisão a um protocolo de acesso múltiplo ajudará o desempenho do protocolo por não transmitir inteiramente um quadro inútil, cujo conteúdo está corrompido (pela interferência de um

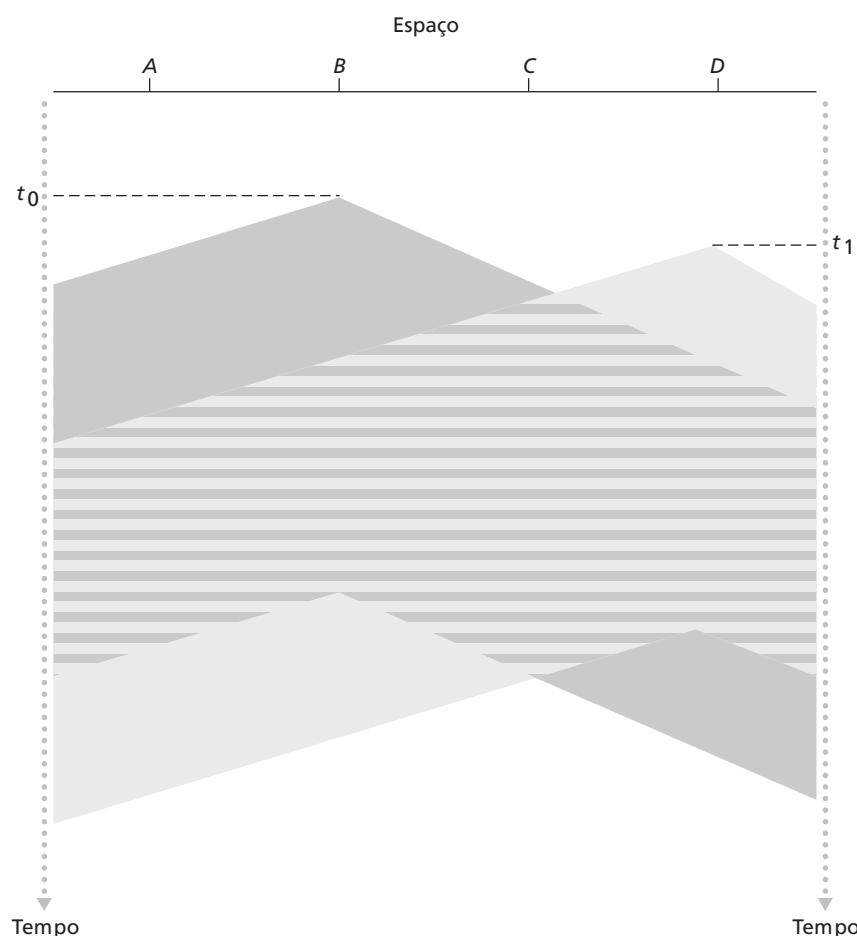


Figura 5.13 Diagrama espaço/tempo de dois nós CSMA com colisão de transmissões

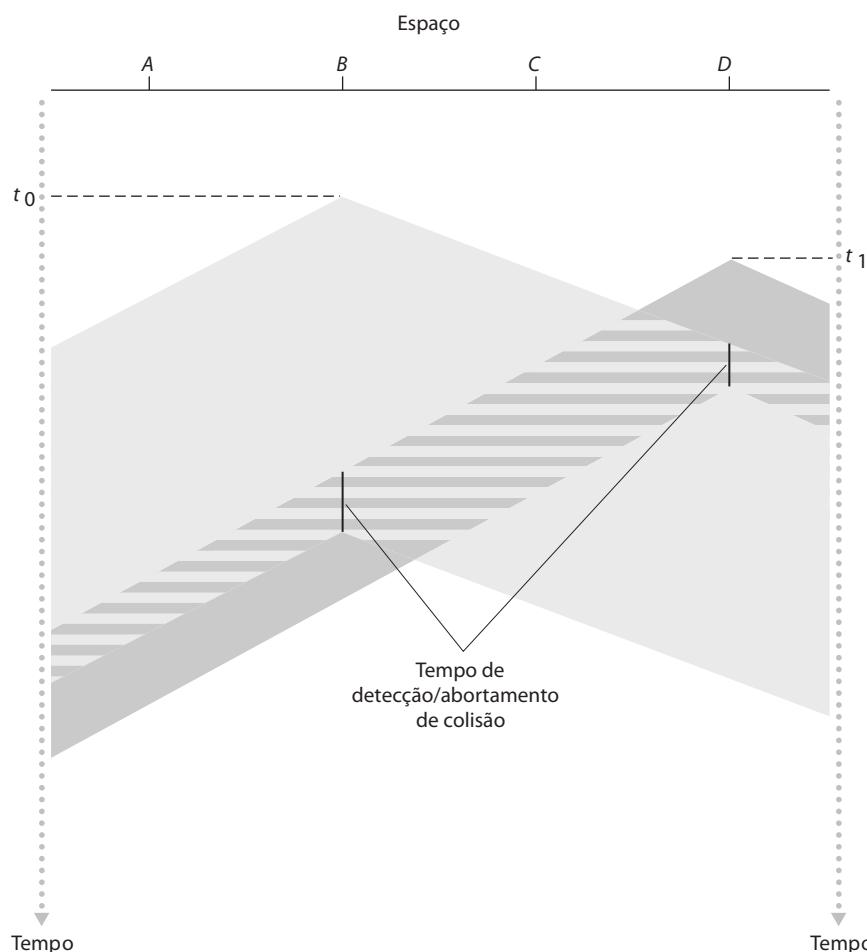


Figura 5.14 CSMA com detecção de colisão

quadro de um outro nó). O protocolo Ethernet, que estudaremos na Seção 5.5, é um protocolo CSMA que usa detecção de colisão.

5.3.3 Protocolos de revezamento

Lembre-se de que duas propriedades desejáveis de um protocolo de acesso múltiplo são: (1) quando apenas um nó está ativo, esse nó ativo tem uma vazão de R bps; (2) quando M nós estão ativos, então cada nó ativo tem uma vazão de aproximadamente R/M bps. Os protocolos ALOHA e CSMA têm a primeira propriedade, mas não a segunda. Isso motivou os pesquisadores a criarem uma outra classe de protocolos — os **protocolos de revezamento**. Como acontece com os protocolos de acesso aleatório, há dezenas de protocolos de revezamento, e cada um desses protocolos tem muitas variações. Discutiremos nesta seção dois dos mais importantes protocolos. O primeiro é o **protocolo de seleção** (polling). O protocolo de polling requer que um dos nós seja designado como nó mestre. O nó mestre **seleciona** cada um dos nós por alternância circular. Em particular, ele envia primeiramente uma mensagem ao nó 1 dizendo que ele (o nó 1) pode transmitir até um certo número máximo de quadros. Após o nó 1 transmitir alguns quadros, o nó mestre diz ao nó 2 que ele (o nó 2) pode transmitir até um certo número máximo de quadros. (O nó mestre pode determinar quando um nó terminou de enviar seus quadros observando a ausência de um sinal no canal.) O procedimento continua dessa maneira, com o nó mestre escolhendo cada um dos nós de maneira cíclica.

O protocolo de polling elimina as colisões e os intervalos vazios que atormentam os protocolos de acesso aleatório, o que permite que ele tenha uma eficiência muito maior. Mas esse protocolo também tem algumas

desvantagens. A primeira desvantagem é que o protocolo introduz um atraso de seleção — o período de tempo requerido para notificar um nó que ele pode transmitir. Se, por exemplo, apenas um nó estiver ativo, então esse nó transmitirá a uma velocidade menor do que R bps, pois o nó mestre tem de escolher cada um dos nós ociosos por vez, cada vez que um nó ativo tiver enviado seu número máximo de quadros. A segunda desvantagem é potencialmente mais séria: se o nó mestre falhar, o canal inteiro ficará inoperante. O Protocolo 802.15 e o protocolo de Bluetooth, que estudaremos na Seção 6.3, são exemplos de protocolos de polling.

O segundo protocolo de revezamento é o **protocolo de passagem de permissão**. Nesse protocolo, não há nó mestre. Um pequeno quadro de finalidade especial conhecido como uma **permissão** (token) é passado entre os nós obedecendo a uma determinada ordem fixa. Por exemplo, o nó 1 poderá sempre enviar a permissão ao nó 2, o nó 2 poderá sempre enviar a permissão ao nó 3, o nó N poderá sempre enviar a permissão ao nó 1. Quando um nó recebe uma permissão, ele a retém somente se tiver alguns quadros para transferir, caso contrário, imediatamente a repassa para o nó seguinte. Se um nó tiver quadros para transmitir quando recebe a permissão, ele enviará um número máximo de quadros e, em seguida, passará a permissão para o nó seguinte. A passagem de permissão é descentralizada e tem uma alta eficiência. Mas também tem seus problemas. Por exemplo, a falha de um nó pode derrubar o canal inteiro. Ou, se um nó acidentalmente se descuida e não libera a permissão, então é preciso chamar algum procedimento de recuperação para colocar a permissão novamente em circulação. Durante muitos anos, foram desenvolvidos muitos protocolos de passagem de permissão, e cada um deles teve de enfrentar esses e outros assuntos delicados; na próxima seção mencionaremos dois desses protocolos, o FDDI e o IEEE 802.5.

5.3.4 Redes locais (LANs)

Protocolos de acesso múltiplo são usados em conjunto com muitos tipos diferentes de canais *broadcast*. Eles têm sido utilizados por canais de satélite e sem fio, cujos nós transmitem sobre um espectro de frequência comum. Atualmente eles estão sendo usados no canal de acesso por cabo à Internet na direção usuário-provedor (veja a Seção 1.2) e são utilizados extensivamente em redes locais (LANs).

Lembre-se de que uma LAN é uma rede de computadores concentrada em uma área geográfica, tal como um prédio ou um campus universitário. Quando um usuário acessa a Internet a partir de um campus universitário ou de uma empresa, o acesso é quase sempre feito por meio de uma LAN. Especificamente, o acesso se dá do provedor para a LAN, para o roteador, para a Internet, como mostra a Figura 5.15. A velocidade de transmissão, R , da maioria das LANs é muito alta. Mesmo no início da década de 1980 já eram comuns LANs de 10 Mbps; hoje, LANs de 100 Mbps são comuns e há LANs disponíveis de 1 Gbps e 10 Gbps.

Na década de 1980 e no início da década de 1990, duas classes de tecnologias de LAN eram populares nos ambientes de trabalho. A primeira classe consistia nas LANs Ethernet (também conhecidas como LANs 802.3

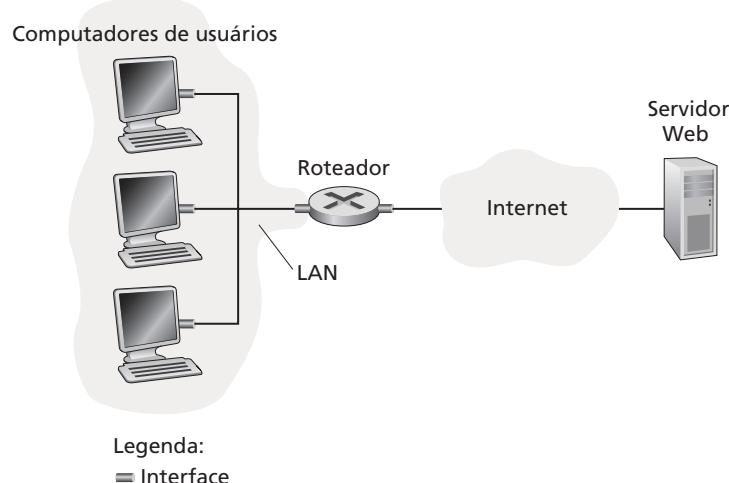


Figura 5.15 Computadores de usuários acessam um servidor Web da Internet por meio de uma LAN. O canal broadcast entre o provedor de um usuário e o roteador consiste em um enlace

[IEEE 802.3 2009]), que eram redes de acesso aleatório. A segunda classe de tecnologias de LAN compreendia as tecnologias de passagem de permissão, incluindo a **token ring** (também conhecida como IEEE 802.5 [IEEE 802.5 2009]) e a **FDDI — interface de dados distribuída de fibra** [Jain, 1994]. Como na Seção 5.5 estudaremos as tecnologias Ethernet com mais detalhes, nesta seção analisaremos apenas as LANs de passagem de permissão. Nossa discussão sobre as tecnologias de passagem de permissão é intencionalmente breve, já que a inexorável concorrência da Ethernet praticamente as extinguiu. Mesmo assim, para dar exemplos de tecnologia de passagem de permissão e apresentar uma pequena perspectiva histórica, será útil falar um pouco sobre anéis de passagem de permissão, ou token rings.

Em uma LAN token ring, os N nós da LAN (hospedeiros e roteadores) estão conectados em um anel por enlaces diretos. A topologia do anel define a ordem de passagem de permissão. Quando um nó obtém a permissão e envia um quadro, este se propaga ao redor do anel inteiro, criando, dessa maneira, um canal virtual de transmissão *broadcast*. À medida que o quadro se propaga, o nó de destino lê esse quadro no meio de transmissão da camada de enlace. O nó que envia o quadro tem a responsabilidade de remover o quadro do anel. A FDDI foi projetada para LANs de alcance geográfico maior, incluindo as denominadas **redes de área metropolitana** (*metropolitan area networks* — MANs). Para LANs de grande alcance geográfico (que se espalham por muitos quilômetros), é ineficiente permitir que um quadro se propague de volta ao nó remetente tão logo tenha passado do nó de destino. A FDDI faz com que o nó de destino remova o quadro do círculo. (Estritamente falando, a FDDI não é um canal *broadcast* puro, pois nem todos os nós recebem todos os quadros transmitidos.)

5.4 Endereçamento na camada de enlace

Nós — isto é, hospedeiros e roteadores — têm endereços de camada de enlace. Você talvez fique surpreso com isso ao lembrar que dissemos, no Capítulo 4, que nós também têm endereços de camada de rede e talvez se pergunte por que, afinal de contas, é preciso ter endereços na camada de rede e na camada de enlace. Nesta seção, além de descrevermos a sintaxe e a função dos endereços de camada de enlace, esperamos esclarecer por que as duas camadas de endereços são úteis e, na verdade, indispensáveis.

Estudaremos também o Protocolo de Resolução de Endereços (*Address Resolution Protocol* — ARP), que provê um mecanismo que habilita os nós a traduzirem endereços IP para endereços de camada de enlace.

5.4.1 Endereços MAC

Na verdade, não é o nó (isto é, o hospedeiro ou o roteador) que tem um endereço de camada de enlace, mas o adaptador do nó. Isso é ilustrado na Figura 5.16. Um endereço de camada de enlace é também denominado um **endereço de LAN**, um **endereço físico** ou um **endereço MAC** (*media access control* — controle de acesso ao

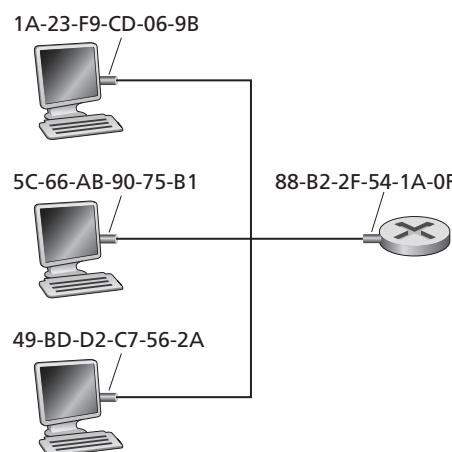


Figura 5.16 Cada adaptador conectado à LAN tem um endereço MAC exclusivo

meio). Como a expressão endereços MAC parece ser o mais popular, daqui em diante nos referiremos a endereços de camada de enlace como endereços MAC. Para a maior parte das LANs (incluindo a Ethernet e as LANs 802.11 sem fio), o endereço MAC tem 6 bytes de comprimento, o que dá 248 possíveis endereços MAC. Como ilustrado na Figura 5.16, esses endereços de 6 bytes são tipicamente expressos em notação hexadecimal, com cada byte do endereço expresso como um par de números hexadecimais. Apesar de os endereços MAC serem projetados como permanentes, é possível mudar o endereço de MAC de um adaptador via software. No entanto, pelo resto desta seção presumiremos que o endereço MAC de um adaptador é fixo.

Uma propriedade interessante dos endereços MAC é que não existem dois adaptadores com o mesmo endereço. Isso pode parecer surpreendente, dado que os adaptadores são fabricados em muitos países por inúmeras empresas diferentes. Como uma empresa fabricante de adaptadores em Taiwan se certifica de que está usando endereços diferentes dos usados por uma empresa fabricante de adaptadores na Bélgica? A resposta é que o IEEE gerencia o espaço físico de endereços MAC. Em particular, quando uma empresa quer fabricar adaptadores, compra, por uma taxa nominal, uma parcela do espaço de endereços que consiste em 224 endereços. O IEEE aloca a parcela de 224 endereços fixando os primeiros 24 bits de um endereço MAC e permitindo que a empresa crie combinações exclusivas com os últimos 24 bits para cada adaptador.

O endereço MAC de um adaptador tem uma estrutura linear (oposta à estrutura hierárquica) e nunca muda, não importando para onde vá o adaptador. Um computador portátil com um cartão Ethernet tem sempre o mesmo endereço MAC, não importando para aonde o computador vá. Um PDA com uma interface 802.11 tem sempre o mesmo endereço MAC aonde quer que vá. Lembre-se de que, ao contrário, um endereço IP tem uma estrutura hierárquica (isto é, uma parte que é da rede e uma parte que é do hospedeiro) e que o endereço IP de um nó precisa ser trocado quando o hospedeiro muda de lugar, por exemplo, muda a rede ao qual está conectado. O endereço MAC de um adaptador é análogo ao número do CPF de uma pessoa, que também tem uma estrutura linear e não muda, não importando para onde a pessoa vá. Um endereço IP é análogo ao endereço postal de uma pessoa, que é hierárquico e precisa ser trocado quando a pessoa muda de lugar. Exatamente como uma pessoa pode achar útil ter um endereço postal, bem como um número de CPF, também é útil para um nó ter um endereço de camada de rede, bem como um endereço MAC.

Como descrevemos no início desta seção, quando um adaptador quer enviar um quadro para algum adaptador de destino, o adaptador remetente insere no quadro o endereço MAC do destino e envia o quadro para dentro da LAN. Se a LAN utilizar transmissão broadcast (como a LAN 802.11 e muitas LANs Ethernet), o quadro será recebido e processado por todos os outros adaptadores na LAN. Em particular, cada adaptador que recebe o quadro verificará se o endereço MAC de destino que está no quadro combina com seu próprio endereço MAC. Se os endereços combinarem, o adaptador extrairá o datagrama encerrado no quadro e o passa para cima na pilha de protocolos até seu nó pai. Se os endereços não combinarem, o adaptador descartará o quadro sem passar o datagrama de camada de rede para cima na pilha de protocolos. Assim, somente o adaptador no nó de destino interromperá seu nó pai quando receber um quadro.

No entanto, às vezes um adaptador remetente quer que todos os outros adaptadores na LAN recebam e processem o quadro que ele está prestes a enviar. Nesse caso, o adaptador remetente insere **um endereço de broadcast** MAC especial no campo de endereço do destinatário do quadro. Para LANs que usam endereços de 6 bytes (como a Ethernet e as LANs de passagem de permissão), o endereço de broadcast é uma cadeia de 48 bits 1 consecutivos (isto é, FF-FF-FF-FF-FF-FF em notação hexadecimal).

5.4.2 ARP (protocolo de resolução de endereços)

Como existem endereços de camada de rede (por exemplo, endereços IP da Internet) e endereços de camada de enlace (isto é, endereços MAC), é preciso fazer a tradução de um para o outro. Para a Internet, esta é uma tarefa do **protocolo de resolução de endereços** (*address resolution protocol — ARP*) [RFC 826].

Para compreender a necessidade de um protocolo como o ARP, considere a rede mostrada na Figura 5.17. Nesse exemplo simples, cada nó tem um único endereço IP e o adaptador de cada nó tem um único endereço MAC. Como sempre, endereços IP são mostrados em notação decimal com pontos e endereços MAC, em notação

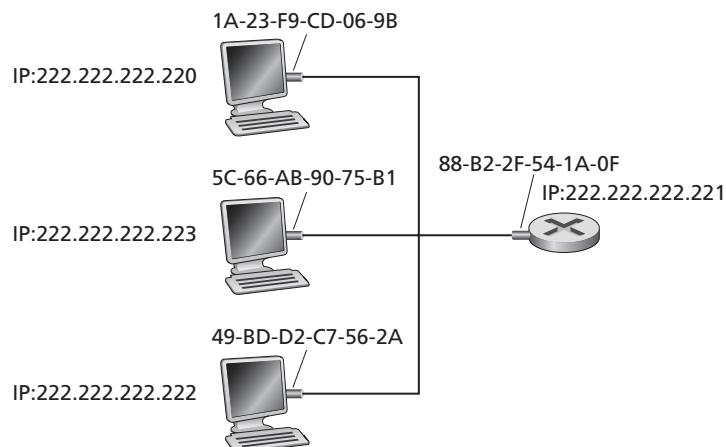


Figura 5.17 Cada nó em uma LAN tem um endereço IP e o adaptador de cada nó tem um endereço MAC

hexadecimal. Agora, suponha que o nó com endereço IP 222.222.222.220 queira mandar um datagrama IP para o nó 222.222.222.222. Nesse exemplo, os nós de fonte e de destino estão na mesma rede (LAN), no sentido do endereçamento estudado na Seção 4.4.2. Para enviar um datagrama, o nó da fonte deve dar a seu adaptador não somente o datagrama IP, mas também o endereço MAC para o nó de destino 222.222.222.222. O adaptador do nó remetente montará então um quadro de camada de enlace contendo o endereço MAC do nó receptor e enviará o quadro para dentro da LAN.

A pergunta importante considerada nesta seção é como o nó remetente determina o endereço MAC para o nó com endereço IP 222.222.222.222? Como você já deve ter adivinhado, ele usa o ARP. Um módulo ARP no nó remetente toma como entrada qualquer endereço IP na mesma LAN e retorna o endereço MAC correspondente. No exemplo em questão, o nó remetente 222.222.222.220 fornece a seu módulo ARP o endereço IP 222.222.222.222 e o módulo ARP retorna endereço MAC correspondente, 49-BD-D2-C7-56-2A.

Assim, vemos que o ARP converte um endereço IP para um endereço MAC. Sob muitos aspectos, o ARP é análogo ao DNS (estudado na Seção 2.5), que converte nomes de hospedeiros para endereços IP. Contudo,

Princípios na prática

Mantendo a independência das camadas

Há diversas razões por que os nós têm endereços MAC além de endereços de camada de rede. Em primeiro lugar, LANs são projetadas para protocolos de camada de rede arbitrários, e não apenas para IP e para a Internet. Se os adaptadores recebessem endereços IP, e não os ‘neutros’ endereços MAC, eles não poderiam suportar com facilidade outros protocolos de camada de rede (por exemplo, IPX ou DECNet). Em segundo lugar, se adaptadores usassem endereços de camada de rede, em vez de endereços MAC, o endereço de camada de rede teria de ser armazenado na RAM do adaptador e reconfigurado toda vez que o adaptador mudasse de local (ou fosse ligado). Outra opção é não usar nenhum endereço nos adaptadores e fazer com que cada um deles passe os dados (caracteristicamente, um datagrama IP) de cada quadro que recebe para cima na pilha de protocolos. A camada de rede poderia, então, verificar se o endereço combina com o endereço da camada de rede. Um problema com essa opção é que o nó pai seria interrompido por cada quadro enviado à LAN, bem como por quadros destinados a outros nós na mesma LAN broadcast. Em resumo, para que as camadas sejam blocos construtivos em grande parte independentes na arquitetura de uma rede, muitas delas precisam ter seu próprio esquema de endereços. Já vimos até agora três tipos diferentes de endereços: nomes de hospedeiros para a camada de aplicação, endereços IP para a camada de rede e endereços MAC para a camada de enlace.

uma importante diferença entre os dois conversores é que o DNS converte nomes de hospedeiros para máquinas em qualquer lugar da Internet, ao passo que o ARP converte endereços IP apenas para nós na mesma sub-rede. Se um nó na Califórnia tentasse usar o ARP para converter o endereço IP de um nó no Mississippi, o ARP devolveria um erro.

Agora que já explicamos o que o ARP faz, vamos ver como ele funciona. Cada nó (hospedeiro ou roteador) tem em sua RAM uma **tabela ARP** que contém mapeamentos de endereços IP para endereços MAC. A Figura 5.18 mostra como seria uma tabela ARP no nó 222.222.222.220. A tabela ARP também contém um valor de tempo de vida (TTL) que indica quando cada mapeamento será apagado da tabela. Note que a tabela não contém necessariamente um registro para cada nó da sub-rede; alguns nós podem ter tido registros que já expiraram, ao passo que outros nós podem jamais ter sido registrados na tabela. Um tempo de remoção típico para um registro é de 20 minutos a partir do momento em que foi colocado em uma tabela ARP.

Suponha que o nó 222.222.222.220 queira enviar um datagrama que tem endereço IP para outro nó daquela sub-rede. O nó remetente precisa obter o endereço MAC do nó de destino, dado o endereço IP daquele mesmo nó. Essa tarefa será fácil se a tabela ARP do nó remetente tiver um registro para esse nó de destino. Mas, e se, naquele momento, a tabela ARP não tiver um registro para o nó destinatário? Em particular, suponha que o nó 222.222.222.220 queira enviar um datagrama para o nó 222.222.222.222. Nesse caso, o nó remetente usa o protocolo ARP para converter o endereço. Primeiro, o nó remetente monta um pacote especial denominado **ARP Query**. Um pacote ARP tem diversos campos, incluindo os endereços IP e MAC de envio e de recepção.

Os pacotes ARP de consulta e de resposta têm o mesmo formato. A finalidade do pacote de consulta ARP é pesquisar todos os outros nós na sub-rede para determinar o endereço MAC correspondente ao endereço IP que está sendo convertido.

Voltando ao nosso exemplo, o nó 222.222.222.220 passa um pacote de consulta ARP ao adaptador juntamente com uma indicação de que o adaptador deve enviar o pacote ao endereço MAC de broadcast, a saber, FF-FF-FF-FF-FF-FF. O adaptador encapsula o pacote ARP em um quadro de camada de enlace, usa o endereço de broadcast como endereço de destino do quadro e transmite o quadro para dentro da sub-rede. Retomando nossa analogia de número do CPF/endereço postal, note que a consulta ARP equivale a um pessoa gritar em um escritório panorâmico cheio de divisórias de alguma empresa (digamos, a empresa AnyCorp): “Qual é o número do CPF da pessoa cujo endereço é divisória 13, Sala 112, AnyCorp, Palo Alto, Califórnia?” O quadro que contém a consulta ARP é recebido por todos os outros adaptadores na sub-rede (por causa do endereço de broadcast) e cada adaptador passa o pacote ARP que está dentro do módulo ARP daquele nó. Cada nó verifica se seu endereço IP combina com o endereço IP de destino no pacote ARP. O único nó (no máximo) que atende a essa condição devolve um pacote ARP de resposta ao nó que fez a consulta, com o mapeamento desejado. O nó que fez a consulta (222.222.222.220) pode, então, atualizar sua tabela ARP e enviar seu datagrama IP, revestido com um quadro de camada de enlace, sendo seu destino MAC o nó que corresponde à consulta de ARP anterior.

O protocolo ARP apresenta algumas características interessantes. Em primeiro lugar, a mensagem de consulta ARP é enviada dentro de um quadro broadcast, ao passo que a mensagem de resposta ARP é enviada dentro de um quadro padrão. Antes de continuar a leitura, é bom que você pense por que isso acontece. Em segundo lugar, o ARP é do tipo plug-and-play, isto é, a tabela de um nó ARP é construída automaticamente — ela não tem de ser configurada por um administrador de sistemas. E, se um nó for desligado da sub-rede, seu registro será finalmente apagado da tabela dos nós remanescentes na sub-rede.

Estudantes se perguntam se o ARP é como um protocolo de camada de enlace ou um protocolo de camada de rede. Como vimos, um pacote ARP é encapsulado dentro de um quadro de camada de enlace e assim encontrase arquitetamente acima da camada de enlace. No entanto, um pacote ARP tem campos que contêm endereços de

Endereço IP	Endereço MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figura 5.18 Uma possível tabela ARP no nó 222.222.222.220

camada de rede, dessa forma é também argumentavelmente um protocolo de camada de rede. Em suma, o ARP é provavelmente o melhor considerado como protocolo que acerta em cheio as fronteiras entre as camadas de enlace e as camadas de rede — não se adequando perfeitamente nas simples pilhas de protocolo que estudamos no Capítulo 1. Tais são as complexidades dos protocolos do mundo real!

Envio de um datagrama para um nó que está fora da sub-rede

Agora já deve estar claro como o ARP funciona quando um nó quer enviar um datagrama a um outro nó na mesma sub-rede. Mas vamos examinar uma situação mais complicada, em que um nó de uma sub-rede quer enviar um datagrama de camada de rede para um nó *que está fora da sub-rede* (isto é, passa por um roteador e entra em outra sub-rede). Vamos discutir essa questão no contexto da Figura 5.19, que mostra uma rede simples constituída de duas sub-redes interconectadas por um roteador.

Há diversos pontos interessantes a notar na Figura 5.19. Primeiramente, há dois tipos de nós: hospedeiros e roteadores. Cada hospedeiro tem exatamente um endereço IP e um adaptador. Mas, como discutimos no Capítulo 4, um roteador tem um endereço IP para *cada* uma de suas interfaces. Para cada interface de roteador também há um módulo ARP (dentro do roteador) e um adaptador. Como o roteador da Figura 5.19 tem duas interfaces, ele tem dois endereços IP, dois módulos ARP e dois adaptadores. É claro que cada adaptador na rede tem seu próprio endereço MAC.

Note também que a Sub-rede 1 tem endereços de rede 111.111.111/24 e que a Sub-rede 2 tem endereços de rede 222.222.222/24. Assim, todas as interfaces conectadas à Sub-rede 1 têm o formato 111.111.111.xxx e todas as interfaces conectadas à Sub-rede 2 têm o formato 222.222.222.xxx.

Agora, vamos examinar como um hospedeiro na Sub-rede 1 enviaria um datagrama a um hospedeiro na Sub-rede 2. Especificamente, suponha que o hospedeiro 111.111.111.111 queira enviar um datagrama IP ao hospedeiro 222.222.222.222. O hospedeiro remetente passa o datagrama a seu adaptador, como sempre. Mas ele deve também indicar a seu adaptador um endereço MAC de destino apropriado. E que endereço MAC o adaptador deveria usar? Poderíamos arriscar o palpite de que o endereço MAC apropriado é aquele do adaptador do hospedeiro 222.222.222.222, a saber, 49-BD-D2-C7-56-2A. Mas esse palpite estaria errado! Se o adaptador remetente usasse aquele endereço MAC, nenhum dos adaptadores da Sub-rede 1 se preocuparia em passar os datagramas IP para cima, para sua camada de rede, já que o endereço de destino do quadro não combinaria com o endereço MAC de nenhum adaptador na Sub-rede 1. O datagrama apenas morreria e iria para o céu dos datagramas.

Se examinarmos cuidadosamente a Figura 5.19, veremos que, para um datagrama ir de 111.111.111.111 até um nó da Sub-rede 2, ele teria de ser enviado primeiramente à interface de roteador 111.111.111.110. Assim, o endereço MAC apropriado para o quadro é o endereço do adaptador para a interface de roteador 111.111.111.110, a saber, E6-E9-00-17-BB-4B. Como o hospedeiro remetente consegue o endereço MAC para a interface 111.111.111.110? Usando o ARP, é claro! Tão logo tenha esse endereço MAC, o adaptador remetente cria um quadro (contendo o datagrama endereçado como 222.222.222.222) e o envia para a Sub-rede 1. O adaptador do roteador na Sub-rede 1 verifica que o quadro de camada de enlace está endereçado a ele e, por conseguinte, o passa para a camada de rede do roteador. Viva — O datagrama IP foi transportado com sucesso do hospedeiro de origem para o roteador! Mas não acabamos. Ainda temos de levar o datagrama do roteador até o destino! O roteador agora tem de determinar

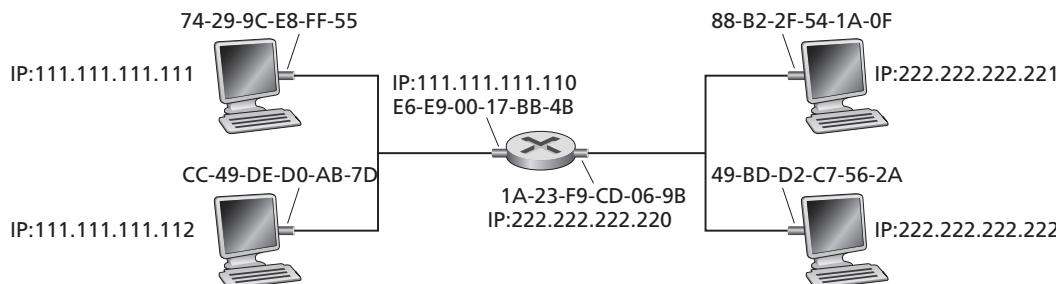


Figura 5.19 Duas sub-redes interconectadas por um roteador

a interface correta para a qual o datagrama deve ser repassado. Como discutimos no Capítulo 4, isso é feito pela consulta a uma tabela de repasse no roteador. A tabela de repasse indica o roteador para o qual o datagrama deve ser repassado via interface de roteador 222.222.222.220. Essa interface então passa o datagrama a seu adaptador, que o encapsula em um novo quadro e envia o quadro para a Sub-rede 2. Dessa vez, o endereço MAC de destino do quadro é, na verdade, o endereço MAC do destino final. E de onde o roteador obtém esse endereço MAC de destino? Do ARP, é claro!

O ARP para Ethernet está definido no RFC 826. Uma boa introdução ao ARP é dada no tutorial do TCP/IP, RFC 1180. Exploraremos o ARP mais detalhadamente nos exercícios ao final deste capítulo.

5.5 Ethernet

A Ethernet praticamente tomou conta do mercado de LANs com fio. Na década de 1980 e início da década de 1990, ela enfrentou muitos desafios de outras tecnologias LAN, incluindo token ring, FDDI e ATM. Algumas dessas outras tecnologias conseguiram conquistar uma parte do mercado de LANs durante alguns anos. Mas, desde sua invenção, em meados da década de 1970, a Ethernet continuou a se desenvolver e crescer e conservou sua posição dominante no mercado. Hoje, ela é de longe a tecnologia preponderante de LAN com fio e é provável que continue assim no futuro próximo. Podemos dizer que a Ethernet está sendo para a rede local o que a Internet tem sido para a rede global.

Há muitas razões para o sucesso da Ethernet. Em primeiro lugar, ela foi a primeira LAN de alta velocidade amplamente disseminada. Como foi disponibilizada cedo, os administradores de rede ficaram bastante familiarizados com a Ethernet — com suas maravilhas e sutilezas — e relutaram em mudar para outras tecnologias LAN quando estas apareceram em cena. Em segundo lugar, token ring, FDDI e ATM são tecnologias mais complexas e mais caras do que a Ethernet, o que desencorajou ainda mais os administradores na questão da mudança. Em terceiro lugar, a razão mais atraente para mudar para uma outra tecnologia LAN (como FDDI e ATM) era normalmente a velocidade mais alta da nova tecnologia; contudo, a Ethernet sempre se defendeu produzindo versões que funcionavam a velocidades iguais, ou mais altas. E, também, a Ethernet comutada foi introduzida no início da década de 1990, o que aumentou ainda mais sua velocidade efetiva de dados. Finalmente, como a Ethernet se tornou muito popular, o hardware para Ethernet (em particular, adaptadores, hubs e comutadores) se tornou mercadoria comum, de custo muito baixo.

A LAN Ethernet original foi inventada em meados da década de 1970 por Bob Metcalfe e David Boggs. A Figura 5.20 mostra o desenho esquemático de Metcalfe para a Ethernet. Na figura você notará que a LAN Ethernet

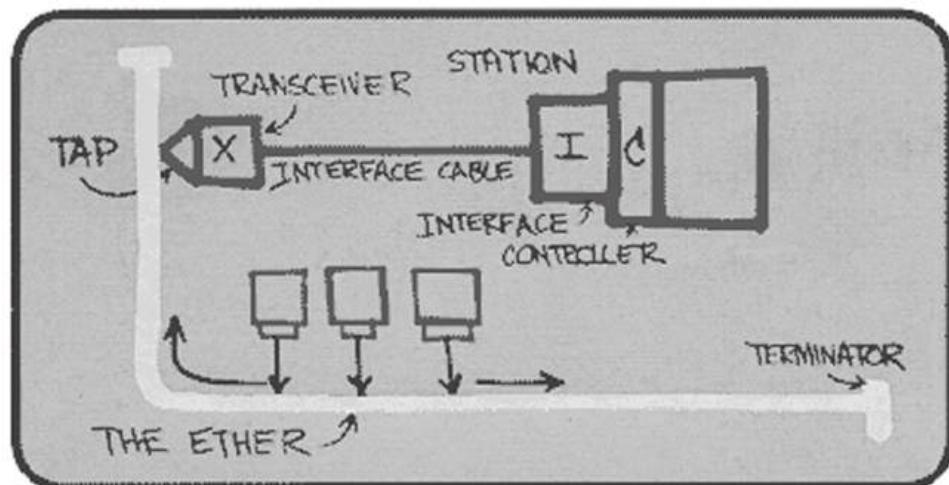


Figura 5.20 O projeto original de Metcalfe levou ao padrão Ethernet 10Base5. Esse padrão incluía um cabo de interface que conectava o adaptador Ethernet a um transceptor externo

original usava um barramento coaxial para interconectar os nós. As topologias de barramento da Ethernet persistiram durante toda a década de 1980 e até metade da década de 1990. Com uma topologia de barramento, a Ethernet é uma transmissão LAN de broadcast — todos os quadros transmitidos movem-se para, e são processados por, *todos* os adaptadores conectados ao barramento.

No fim da década de 90, a maioria das empresas e universidades já tinha substituído suas LANs com instalações Ethernet usando topologia de estrela baseada em um hub (repetidor). Como demonstrado na figura 5.21, nessas instalações os hospedeiros (e o roteador) estão diretamente conectados a um hub com um cabo de pares de cobre. Um hub é um dispositivo de camada física que age em bits individuais e não em quadros. Quando um bit, representando 0 ou 1, chega em uma interface, o hub simplesmente recria o bit, aumenta a energia e transmite o bit para todas as outras interfaces. Sendo assim, Ethernet com uma topologia de estrela baseada em um hub também é uma LAN da broadcast — sempre que um hub recebe um bit de uma de suas interfaces, ela envia uma cópia para todas as outras interfaces. Em particular, se um hub recebe quadros de duas diferentes interfaces ao mesmo tempo, ocorre uma colisão e os nós que criaram os quadros precisam retransmitir.

No começo dos anos 2000, Ethernet passou por outra grande mudança revolucionária. As instalações Ethernet continuaram a usar a topologia de estrela, mas o hub no núcleo foi substituído por um comutador. Examinaremos o comutador da Ethernet com mais atenção depois nesse capítulo. Por enquanto, só mencionaremos que um comutador é a prova de “colisões” e também um autêntico contador no modo armazenamento e encaminhamento; mas diferentemente dos roteadores, que operam até a camada 3, um comutador opera até a camada 2.

5.5.1 Estrutura do quadro Ethernet

Podemos aprender muito sobre a Ethernet examinando o quadro Ethernet que é mostrado na Figura 5.22. Para colocar nossa discussão de quadros Ethernet em um contexto tangível, vamos considerar o envio de um datagrama IP de um hospedeiro a outro, estando os dois na mesma LAN Ethernet (por exemplo, a LAN Ethernet da Figura 5.21). Embora a carga útil do nosso quadro Ethernet seja um diagrama IP, aproveitamos para comentar que um quadro Ethernet também pode carregar outros pacotes de camada de rede. Seja o adaptador do remetente, adaptador A, com o endereço MAC AA-AA-AA-AA-AA-AA; seja o adaptador receptor, adaptador B, com o endereço MAC BB-BB-BB-BB-BB-BB. O adaptador remetente encapsula o datagrama IP dentro de um quadro Ethernet e passa o quadro à camada física. O adaptador receptor recebe o quadro da camada física, extrai o datagrama IP e o passa para a camada de rede. Nesse contexto, vamos examinar agora os seis primeiros campos do quadro Ethernet, como ilustrados na Figura 5.22.

Campo de dados (46 a 1.500 bytes). Esse campo carrega o datagrama IP. A unidade máxima de transferência (MTU) da Ethernet é 1.500 bytes. Isso significa que, se o datagrama IP exceder 1.500 bytes, o hospedeiro terá de fragmentar o datagrama, como discutimos na Seção 4.4.1. O tamanho mínimo

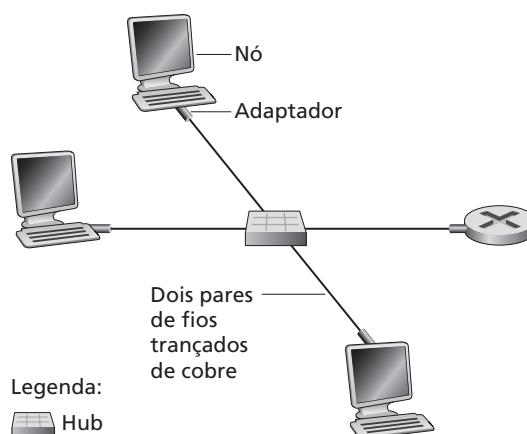


Figura 5.21 Topologia em estrela para Ethernet. Nós são interconectados com um hub



Figura 5.22 Estrutura do quadro Ethernet

do campo de dados é 46 bytes. Isso significa que, se um datagrama IP tiver menos do que 46 bytes, o campo de dados terá de ser ‘recheado’ de modo a completar os 46 bytes. Quando se usa recheio (stuffing), os dados passados à camada de rede contêm o recheio, bem como um datagrama IP. A camada de rede usa o campo de comprimento do cabeçalho do datagrama IP para remover o recheio.

Endereço de destino (6 bytes). Esse campo contém os endereços MAC do adaptador de destino, BB-BB-BB-BB-BB-BB. Quando o adaptador B recebe um quadro Ethernet cujo endereço de destino é ou BB-BB-BB-BB-BB-BB, ou o endereço de broadcast MAC, ele passa o conteúdo do campo de dados para a camada de rede. Se receber um quadro com qualquer outro endereço MAC, ele o descartará.

Endereço de fonte (6 bytes). Esse campo contém o endereço MAC do adaptador que transmite o quadro para a LAN, neste exemplo, AA-AA-AA-AA-AA-AA.

Campo de tipo (2 bytes). O campo de tipo permite que a Ethernet multiplexe protocolos de camada de rede. Para entender isso, é preciso ter em mente que hospedeiros podem usar outros protocolos de camada de rede além do IP. Na verdade, um dado hospedeiro pode suportar vários protocolos de camada de rede e usar protocolos diferentes para aplicações diferentes. Por essa razão, quando o quadro Ethernet chega ao adaptador B, o adaptador B precisa saber para qual protocolo de camada de rede ele deve passar (isto é, demultiplexar) o conteúdo do campo de dados. O IP e outros protocolos de camada de rede (por exemplo, Novell IPX ou AppleTalk) têm seu próprio número de tipo padronizado. Além disso, o protocolo ARP (discutido na seção anterior) tem seu próprio número de tipo, e se o quadro que chegar conter um pacote ARP (por exemplo, tem um campo de tipo 0806 hexadecimal), o pacote ARP será de multiplexado até o protocolo ARP. Note que o campo de tipo é análogo ao campo de protocolo no datagrama de camada de rede e aos campos de número de porta no segmento de camada de transporte; todos esses campos servem para ligar um protocolo de uma camada a um protocolo da camada acima.

Verificação de redundância cíclica (CRC) (4 bytes). Como discutido na Seção 5.2.3, a finalidade do campo de CRC é permitir que o adaptador receptor, o adaptador B, detecte se algum erro foi introduzido no quadro.

Preâmbulo (8 bytes). O quadro Ethernet começa com um campo de preâmbulo de 8 bytes. Cada um dos primeiros 7 bytes do preâmbulo tem um valor de 10101010; o último byte é 10101011. Os primeiros 7 bytes do preâmbulo servem para ‘despertar’ os adaptadores receptores e sincronizar seus relógios com o relógio do remetente. Por que os relógios poderiam estar fora de sincronia? Não esqueça que o adaptador A visa transmitir o quadro a 10 Mbps, 100 Mbps ou 1 Gbps, dependendo do tipo de LAN Ethernet. Contudo, como nada é absolutamente perfeito, o adaptador A não transmitirá o quadro exatamente à mesma velocidade-alvo; sempre haverá alguma variação em relação à velocidade-alvo, uma variação que não é conhecida *a priori* pelos outros adaptadores na LAN. Um adaptador receptor pode sincronizar com o relógio do adaptador A simplesmente sincronizando os bits dos primeiros 7 bytes do preâmbulo. Os dois últimos bits do oitavo byte do preâmbulo (os primeiros dois 1s consecutivos) alertam o adaptador B de que ‘algo importante’ está chegando.

A Ethernet usa transmissão em banda-base, isto é, o adaptador envia um sinal digital diretamente ao canal broadcast. A placa de interface não desloca o sinal para outra banda de frequência, como é feito nos sistemas ADSL e de modem a cabo. Muitas tecnologias Ethernet (por ex., 10BASE-T) também usam a codificação Manchester, como mostra a Figura 5.23. Com codificação Manchester cada bit contém uma transição; um 1 tem uma transição de cima para baixo, ao passo que um 0 tem uma transição de baixo para cima. A razão para o uso da codificação Manchester é que os relógios nos adaptadores remetentes e receptores não estão perfeitamente sincronizados. Ao incluir uma transição no meio de cada bit, o hospedeiro receptor pode sincronizar seu relógio com o relógio do hospedeiro remetente. Tão logo o relógio do adaptador receptor esteja sincronizado, o receptor pode delinear cada bit e deter-

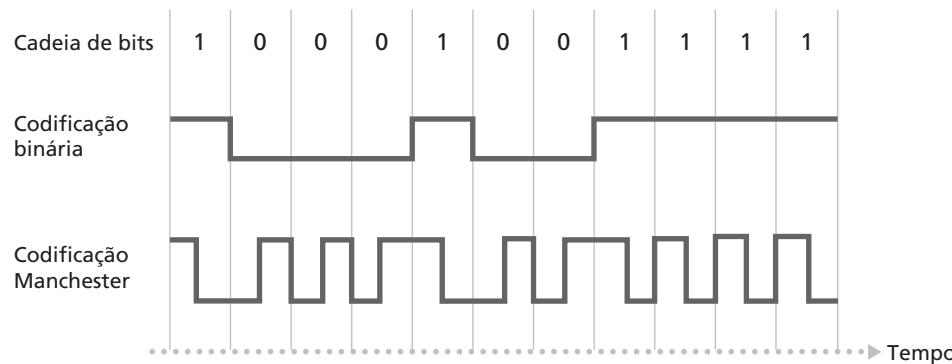


Figura 5.23 Codificação Manchester

minar se é um 1 ou um 0. A codificação Manchester é mais uma operação de camada física do que de camada de enlace; contudo, descrevemos essa operação brevemente aqui porque ela é muito usada na Ethernet.

Um serviço não confiável não orientado para conexão

Todas as tecnologias Ethernet fornecem **serviço não orientado para conexão** à camada de rede. Isto é, quando o adaptador A quer enviar um datagrama ao adaptador B, o adaptador A encapsula o datagrama em um quadro Ethernet e envia o quadro à LAN, sem se conectar previamente a B. Esse serviço de camada 2 não orientado para conexão é análogo ao serviço de datagrama de camada 3 do IP e ao serviço de camada 4 não orientado para conexão do UDP.

As tecnologias Ethernet fornecem um **serviço não confiável** à camada de rede. Especificamente, quando o adaptador B recebe um quadro do adaptador A, ele submete o quadro a uma verificação de CRC, mas não envia um reconhecimento quando um quadro passa na verificação CRC nem um reconhecimento negativo quando um quadro não passa na verificação de CRC. Quando um quadro não passa na verificação de CRC, o adaptador B simplesmente o descarta. Assim, o adaptador A não têm a mínima ideia se o quadro que transmitiu passou na verificação CRC. Essa falta de transporte confiável (na camada de enlace) ajuda a tornar a Ethernet simples e barata. Mas também significa que a sequência de datagramas passada à camada de rede pode ter lacunas.

Se houver lacunas devido a quadros Ethernet descartados, o protocolo de camada de aplicação do hospedeiro B também verá essas lacunas? Como aprendemos no Capítulo 3, isso depende exclusivamente de a aplicação estar usando UDP ou TCP. Se a aplicação estiver usando UDP, então a aplicação no hospedeiro B realmente sofrerá lacunas nos dados. Por outro lado, se a aplicação estiver usando TCP, então o TCP no hospedeiro B não reconhecerá os dados contidos em quadros descartados, fazendo com que o TCP no hospedeiro A retransmita. Note que, quando o TCP retransmite dados, esses dados eventualmente retornarão ao adaptador Ethernet no qual foram descartados. Assim, nesse sentido, a Ethernet realmente retransmite dados, embora não saiba se está transmitindo um datagrama novo com dados novos, ou um datagrama que contém dados que já foram transmitidos pelo menos uma vez.

5.5.2 CSMA/CD: o protocolo de acesso múltiplo da Ethernet

Quando os nós estão interconectados com um hub (e não a um comutador de camada de enlace), como demonstrado na Figura 5.21, a LAN Ethernet é uma verdadeira LAN de *broadcast* — isto é, quando um adaptador transmite um quadro, todos os adaptadores na LAN recebem o quadro. Como pode empregar *broadcast*, a Ethernet precisa de um protocolo de acesso múltiplo — ela usa o famoso protocolo de acesso múltiplo CSMA/CD. Lembre-se de que, na Seção 5.3 mostramos que o CSMA/CD faz o seguinte:

1. Um adaptador pode começar a transmitir a qualquer tempo, ou seja, não há noção de compartimentos de tempo.
2. Um adaptador nunca transmite um quadro quando percebe que algum outro adaptador está transmitindo, ou seja, ele usa detecção de portadora.



História

Bob Metcalfe e a Ethernet

Quando era estudante de doutorado na Universidade Harvard, no início da década de 1970, Bob Metcalfe trabalhava na ARPAnet no M.I.T. Durante seus estudos, ele tomou conhecimento do trabalho de Abramson com o ALOHA e os protocolos de acesso aleatório. Após ter concluído seu doutorado e pouco antes de começar um trabalho no PARC da Xerox (Palo Alto Research Center), fez uma visita de três meses a Abramson e seus colegas da Universidade do Havaí, quando pôde observar, em primeira mão, a ALOHAnet. No PARC da Xerox, Metcalfe conheceu os computadores Alto, que, em muitos aspectos, foram os predecessores dos computadores pessoais da década de 1980. Ele entendeu a necessidade de montar esses computadores em rede de um modo que não fosse dispendioso. Assim, munido com o que conhecia sobre ARPAnet, ALOHAnet e protocolos de acesso aleatório, Metcalfe, juntamente com seu colega David Boggs, inventou a Ethernet.

A Ethernet original de Metcalfe e Boggs executava a 2,94 Mbps e interligava até 256 hospedeiros a distâncias de até 1,5 km. Metcalfe e Boggs conseguiram que a maioria dos pesquisadores do PARC da Xerox se comunicassem por meio de seus computadores Alto. Então, Metcalfe forjou uma aliança entre a Xerox, a Digital e a Intel para estabelecer a Ethernet de 10 Mbps como padrão da Ethernet, ratificado pelo IEEE. A Xerox não demonstrou muito interesse em comercializar a Ethernet. Em 1979, Metcalfe abriu sua própria empresa, a 3Com, para desenvolver e comercializar tecnologia de rede, incluindo a tecnologia Ethernet. Em particular, a 3Com desenvolveu e comercializou cartões de Ethernet no início da década de 1980 para os então imensamente populares PCs da IBM. Metcalfe deixou a 3Com em 1990, quando a empresa tinha dois mil funcionários e 400 milhões de dólares de receita. Até novembro de 2003, a 3Com empregava mais de 2.900 pessoas no mundo inteiro.

3. Um adaptador que está transmitindo aborta sua transmissão quando percebe que algum outro adaptador está transmitindo, ou seja, usa detecção de colisão.
4. Antes de tentar uma retransmissão, um adaptador espera um período de tempo aleatório que é caracteristicamente pequeno em comparação com o tempo de transmissão de um quadro.

Esses mecanismos conferem ao CSMA/CD um desempenho muito melhor, em ambiente LAN, do que o do slotted ALOHA. De fato, se o atraso máximo de propagação entre estações for muito pequeno, a eficiência do CSMA/CD poderá ficar próxima a 100 por cento. Mas note que o segundo e o terceiro mecanismos que citamos requerem que cada adaptador Ethernet seja capaz de (1) perceber quando algum outro adaptador está transmitindo e (2) detectar uma colisão enquanto estiver transmitindo. Adaptadores Ethernet realizam essas duas tarefas medindo os níveis de tensão antes e durante a transmissão.

Cada adaptador executa o protocolo CSMA/CD sem coordenação explícita com os outros adaptadores na Ethernet. Dentro de um adaptador específico, o protocolo CSMA/CD funciona como segue:

1. O adaptador obtém um datagrama de camada de rede de seu nó pai, prepara um quadro Ethernet e o coloca no buffer do adaptador.
2. Se o adaptador percebe que o canal está ocioso (isto é, não há nenhuma energia de sinal entrando do canal para o adaptador durante 96 tempos de bits), ele começa a transmitir o quadro. Se o adaptador percebe que o canal está ocupado, espera até perceber que não há nenhuma energia de sinal (mais 96 tempos de bits) e então começa a transmitir o quadro.
3. Enquanto transmite, o adaptador monitora a presença de energia de sinal vinda de outros adaptadores. Se o adaptador transmitir o quadro inteiro sem detectar energia de sinal vinda de outros adaptadores, ele considerará concluída a operação com o quadro.
4. Se o adaptador detectar energia de sinal vinda de outros adaptadores enquanto está transmitindo, ele parará de transmitir seu quadro e, em vez dele, transmitirá um sinal de reforço de colisão de 48 bits.
5. Após abortar (isto é, transmitir o sinal de reforço), o adaptador entra em fase de **backoff exponencial**. Especificamente, quando está transmitindo um dado quadro, após sofrer a enésima colisão

em seguida para esse quadro, o adaptador escolhe um valor para K aleatoriamente de $\{0, 1, 2, \dots, 2^m - 1\}$, onde $m = \min(n, 10)$. O adaptador então espera $K \cdot 512$ tempos de bits e então retorna à etapa 2.

Alguns comentários adicionais sobre o protocolo CSMA/CD se fazem necessários. A finalidade do sinal de reforço de colisão é garantir que todos os outros adaptadores que estejam transmitindo fiquem cientes da colisão. Vamos ver um exemplo. Suponha que o adaptador A comece a transmitir um quadro e, exatamente antes de o sinal de A alcançar o adaptador B, o adaptador B comece a transmitir. Assim, B terá transmitido apenas alguns bits quando abortar sua transmissão. Esses poucos bits, na verdade, propagar-se-ão até A, mas podem não constituir energia suficiente para que A detecte a colisão. Para garantir que A detecte a colisão (de modo que ele também possa abortar), B transmite o sinal de reforço de colisão de 48 bits.

Em seguida, considere o algoritmo de backoff exponencial. A primeira coisa a notar aqui é que um tempo de bit (isto é, o tempo para transmitir um único bit) é muito curto; para uma Ethernet de 10 Mbps, um tempo de bit é 0,1 microsegundo. Agora, vamos considerar um exemplo. Suponha que um adaptador tente transmitir um quadro pela primeira vez e que, enquanto está transmitindo, detecte uma colisão. O adaptador então escolhe $K = 0$ com probabilidade de 0,5 ou escolhe $K = 1$ com probabilidade de 0,5. Se o adaptador escolher $K = 0$, imediatamente passará para a etapa 2 após transmitir o sinal de reforço de colisão. Se escolher $K = 1$, ele esperará 51,2 microsegundos antes de voltar à etapa 2. Após uma segunda colisão, K é escolhido com igual probabilidade no conjunto $\{0, 1, 2, 3\}$. Após três colisões, K é escolhido com igual probabilidade no conjunto $\{0, 1, 2, 3, 4, 5, 6, 7\}$. Após dez ou mais colisões, K é escolhido com igual probabilidade no conjunto $\{0, 1, 2, \dots, 1023\}$. Assim, o tamanho dos conjuntos dos quais K é escolhido cresce exponencialmente com o número de colisões (até $n = 10$); é por essa razão que o algoritmo de backoff da Ethernet é denominado ‘algoritmo de backoff exponencial’.

O padrão Ethernet impõe limites à distância entre dois nós quaisquer. Esses limites garantem que, se o adaptador A escolher um valor de K mais baixo do que escolheram todos os outros adaptadores envolvidos em uma colisão, ele poderá transmitir seu quadro sem sofrer uma nova colisão. Estudaremos essa propriedade mais detalhadamente nos exercícios ao final deste capítulo.

Por que usar backoff exponencial? Por que, por exemplo, não escolher um K entre $\{0, 1, 2, 3, 4, 5, 6, 7\}$ após cada colisão? A razão é que, quando um adaptador sofre sua primeira colisão, ele não tem ideia de quantos adaptadores estão envolvidos na colisão. Se houver somente um pequeno número de adaptadores colidindo, terá sentido escolher K de um pequeno conjunto de valores pequenos. Por outro lado, se muitos adaptadores estiverem envolvidos na colisão, terá sentido escolher K dentro de um conjunto de valores maiores e mais dispersos (por quê?). Aumentando o tamanho do conjunto após cada colisão, o adaptador se ajusta adequadamente a esses diferentes cenários.

Mencionamos também que cada vez que um adaptador prepara um novo quadro para transmissão, ele executa o algoritmo CSMA/CD que acabamos de apresentar, não levando em conta nenhuma colisão que possa ter ocorrido no passado recente. Assim, é possível que um adaptador com um novo quadro consiga uma transmissão imediatamente enquanto vários outros adaptadores estão no estado de backoff exponencial.

Eficiência da Ethernet

Quando apenas um nó tem um quadro para enviar, ele pode transmitir à velocidade total da tecnologia Ethernet (10 Mbps, 100 Mbps ou 1 Gbps). Contudo, se muitos nós tiverem quadros para transmitir, a velocidade efetiva de transmissão do canal poderá ser muito menor. Definimos a **eficiência da Ethernet** como a fração final de tempo durante a qual os quadros estão sendo transmitidos no canal, sem colisões, quando há um grande número de nós ativos, e cada nó tem um grande número de quadros a enviar. Para uma aproximação da eficiência da Ethernet, vamos designar como d_{prop} o tempo máximo para que a energia de sinal se propague entre dois adaptadores quaisquer. Seja d_{trans} o tempo que leva para transmitir um quadro Ethernet de tamanho máximo (aproximadamente 1,2 milissegundo para uma Ethernet de 10 Mbps). Uma derivação da eficiência da Ethernet ultrapassa o escopo deste livro — para mais detalhes, consulte [Lam, 1980] e [Bertsekas, 1991]. Nesta subseção, vamos apenas mencionar a seguinte aproximação:

$$\text{Eficiência} = \frac{1}{1 + 5d_{prop}/d_{trans}}$$

Vemos por essa fórmula que, quando d_{prop} se aproxima de 0, a eficiência se aproxima de 1. Isso está de acordo com a intuição que temos de que, se o atraso de propagação for 0, os nós em colisão vão abortar imediatamente sem desperdiçar o canal. Além disso, à medida que d_{trans} vai ficando muito grande, a eficiência se aproxima de 1. Isso também é intuitivo, porque, quando um quadro captura o canal, vai prendê-lo por muito tempo; assim, o canal estará realizando um trabalho produtivo durante a maior parte do tempo.

5.5.3 Tecnologias Ethernet

Em nossa discussão anterior, nos referimos à Ethernet como se fosse um protocolo único padrão. Mas na verdade, a Ethernet aparece em diferentes versões, com acrônimos um pouco confusos como 10BASE-T, 10BASE-2, 100BASE-T, 1000BASE-LX e 10GBASE-T. Estas e muitas outras tecnologias Ethernet foram padronizadas através dos anos pelos grupos de trabalho IEEE 802.3. Apesar de esses acrônimos parecerem confusos, existe uma ordem seguida. A primeira parte do acrônimo se refere à velocidade padrão: 10, 100, 1000 ou 10G, por 10 Megabits (por segundo), 100 Megabits e 10 Gigabits Ethernet, respectivamente. “BASE” se refere à banda base da Ethernet, significando que a mídia física só suporta o tráfego da Ethernet; quase todos os padrões 802.3 são para banda base. A parte final do acrônimo se refere à mídia física em si; a Ethernet é tanto uma camada de enlace quanto uma camada física que inclui um cabo coaxial, fio de cobre e fibra. Geralmente um “T” se refere a um cabo de par trançado de fios de cobre.

Historicamente, uma Ethernet era inicialmente concebida como um segmento de um cabo coaxial, como mostrado na Figura 5.20. Os primeiros padrões 10BASE-2 e 10BASE5 especificavam 10Mbps Ethernet sobre dois tipos de cabos coaxiais, cada um limitado a um comprimento de 200 metros. Extensões mais longas podiam ser obtidas usando um repetidor — um dispositivo de camada — que recebe um sinal no lado de entrada, e regenera o sinal no lado de saída. Um cabo coaxial, como na Figura 5.20, correspondia muito bem a nossa visão da Ethernet como uma transmissão broadcast — todos os quadros transmitidos por uma interface são recebidos em outras interfaces, e o protocolo CDMA/CD da Ethernet resolve satisfatoriamente os vários problemas de acesso. Os nós são simplesmente conectados ao cabo, e voilà, temos uma rede local!

A Ethernet passou por uma série de etapas de evolução ao longo dos anos, e a Ethernet atual é muito diferente do projeto original da topologia de barramento que usava cabos coaxiais. Na maioria das instalações de hoje, os nós são conectados a um comutador via segmentos ponto a ponto feitos de cabos de pares trançados de fios de cobre ou cabos de fibra ótica, como demonstrado na figura 5.24.

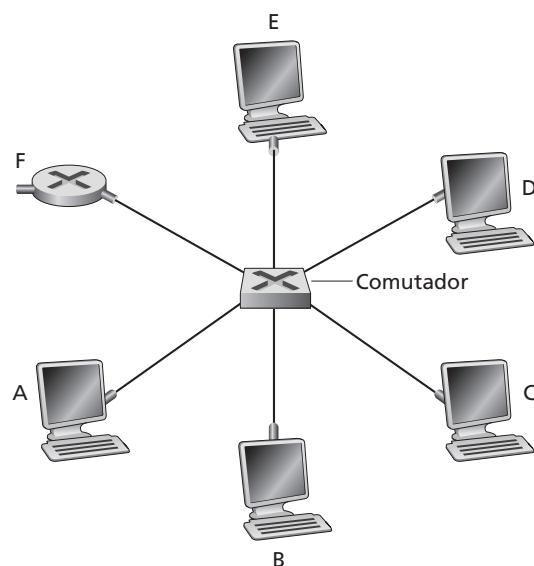


Figura 5.24 Uma camada de enlace interconectando seis nós.

No meio da década de 1990, a Ethernet foi padronizada em 100 Mbps, 10 vezes mais rápida do que a Ethernet 10 Mbps. O formato de quadro e o protocolo Ethernet MAC original foram preservados, mas camadas físicas de alta velocidade foram definidas para fios de cobre e fibra. A Figura 5.25 mostra padrões diferentes nos formato de quadro e protocolo Ethernet MAC comum. A Ethernet de 100 Mpbs é limitada a 100 metros de distância através de um cabo de par trançado, e vários quilômetros através de fibra, o que permite comutações Ethernet em diferentes prédios a serem conectados.

A rede Gigabit Ethernet é uma extensão dos muito bem-sucedidos padrões 10 Mbps e 100 Mbps. Oferecendo uma velocidade bruta de 1000 Mbps, a Gigabit Ethernet mantém total compatibilidade com a imensa base instalada de equipamentos Ethernet. A Gigabit Ethernet apresenta o seguinte padrão de funcionamento:

- Usa o formato-padrão do quadro Ethernet (Figura 5.22) e é compatível com as tecnologias 10BaseT e 100BaseT. Isso permite fácil integração da Gigabit Ethernet com a base instalada de equipamentos Ethernet.
- Permite enlaces ponto a ponto, bem como canais *broadcast* compartilhados. Enlaces ponto a ponto usam comutadores, ao passo que canais *broadcast* usam hubs, como descrito anteriormente para a 10BaseT e para a 100BaseT. No jargão da Gigabit Ethernet, os hubs são denominados *distribuidores com buffer*.
- Utiliza CSMA/CD para canais *broadcast* compartilhados. Para conseguir eficiência aceitável, a distância máxima entre os nós deve ser severamente limitada.
- Permite operação *full-duplex* a 1.000 Mbps em ambas as direções para canais ponto a ponto.

Inicialmente operando através de fibra ótica, a Gigabit Ethernet está disponível para ser instalada por meio de cabeamento UTP categoria 5. A 10 Gbps Ethernet (10GBASET-T) foi padronizada em 2007, sustentando uma capacidade de transmissão ainda mais alta.

Vamos concluir nossa discussão sobre a tecnologia Ethernet considerando uma dúvida que pode estar incomodando você. Na época em que a topologia de barramento e o hub baseado em topologia de estrela eram usados, a Ethernet era evidentemente um enlace broadcast (como definido na Seção 5.3), onde colisões de quadro ocorriam quando nós transmitiam ao mesmo tempo. Para lidar com essas colisões, o padrão Ethernet incluiu o protocolo CSMA/CD, que é especialmente efetivo para transmissões de LAN abrangendo pequenos raios geográficos. Mas se o uso atual prevalente da Ethernet é baseado em comutadores com a topologia de estrela, usando o modo de comutação de armazenamento e encaminhamento, existe a necessidade de se usar um protocolo Ethernet MAC? Como veremos na Seção 5.6, um comutador coordena sua transmissão e nunca encaminha mais de um quadro por vez na mesma interface. Além disto, comutadores modernos são full-duplex (ou seja, fazem comunicações simultâneas em ambas as direções), de modo que um comutador e um nó possam sem interferência enviar quadros um ao outro ao mesmo tempo. Ou seja, em uma Ethernet Lan com um comutador não ocorrem colisões, portanto não existe a necessidade de um protocolo MAC!

Como vimos, a Ethernet atual é muito diferente da Ethernet original concebida por Metcalfe e Boggs há mais de 30 anos atrás — as velocidades tiveram um aumento de três ordens de grandeza, os quadros da Ethernet são transportadas por uma variedade de mídias. Comutadores Ethernets se tornaram dominantes e até o protocolo MAC é muitas vezes desnecessário. Será que tudo isso ainda é Ethernet? A resposta é, obviamente, “sim, por definição.” No entanto, é interessante observar que por todas essas mudanças existiu uma constante que continuou inalterada por mais de 30 anos — o formato do quadro Ethernet. Talvez então isso seja a peça central do padrão Ethernet.

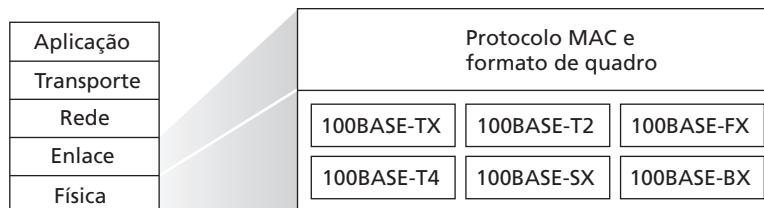


Figura 5.25 Padrões 100 Mpbs Ethernet: uma camada de enlace comum, diferentes camadas físicas.

5.6 Comutadores de camada de enlace

Como demonstrado na Figura 5.26 as LANs Ethernet modernas usam uma topologia em estrela, com cada nó conectado a um comutador central. Até aqui, temos sido vagos sobre como um comutador trabalha e o que ele faz. A função de um comutador é receber quadros de camada de enlace e encaminhá-los para enlaces de saída; estudaremos essa função de encaminhamento detalhadamente em breve. O comutador em si é transparente até aos nós; ou seja, um nó endereça um quadro a outro nó (em vez de endereçar o quadro ao comutador) que alegremente envia o quadro à LAN, inconsciente de que um comutador irá receber o quadro e encaminhá-lo a outros nós. A velocidade com que os quadros chegam a qualquer interface de saída do comutador pode temporariamente exceder a capacidade do enlace daquela interface. Para resolver esse problema, interfaces de saídas do comutador têm buffers, da mesma forma que uma interface de saída de um roteador tem buffers para datagramas. Vamos agora observar mais atentamente o funcionamento de um comutador.

5.6.1 Repasse e filtragem

Filtragem é a capacidade de um comutador que determina se um quadro deve ser repassado para alguma interface ou se deve apenas ser descartado. **Repasse** é a capacidade de um comutador que determina as interfaces para as quais um quadro deve ser dirigido e então dirigir o quadro a essas interfaces. Filtragem e repasse por comutadores são feitos com uma **tabela de comutação**. A tabela de comutação contém registros para alguns nós da LAN, mas não necessariamente para todos. Um registro de um nó na tabela de comutação contém (1) o endereço MAC do nó, (2) a interface do comutador que leva em direção ao nó e (3) o horário em que o registro para o nó foi colocado na tabela. Um exemplo de tabela de comutação para a LAN da Figura 5.26 é mostrado na Figura 5.27. Embora essa descrição de repasse de quadros possa parecer semelhante à nossa discussão de repasse de datagramas no Capítulo 4, veremos brevemente que há diferenças significativas. Uma diferença importante é que os comutadores encaminham pacotes baseados em endereços MAC em vez de endereços de IP. Também veremos que uma tabela de comutação é montada de maneira diferente da montagem de tabelas de roteamento.

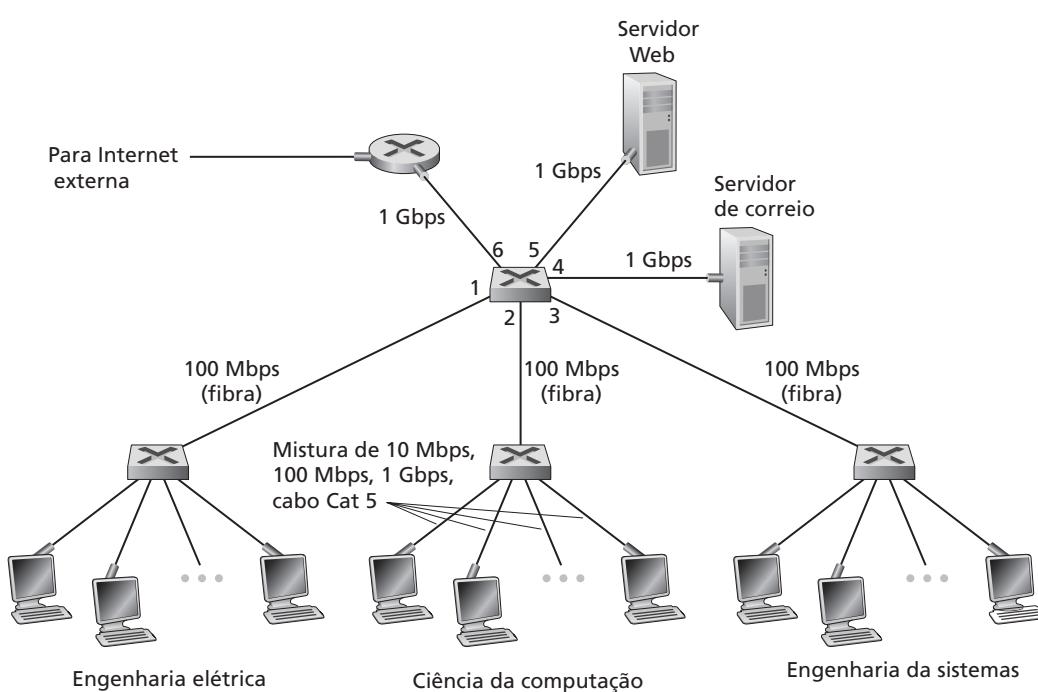


Figura 5.26 Três Ethernets departamentais interconectados com um hub

Endereço	Interface	Horário
62-FE-F7-11-89-A3	1	9h32
7C-BA-B2-B4-91-10	3	9h36
....

Figura 5.27 Parte de uma tabela de comutação para a LAN da Figura 5.26

Para entender como funciona a filtragem e o repasse por comutadores, suponha que um quadro com endereço de destino DD-DD-DD-DD-DD-DD chegue ao comutador na interface x. O comutador indexa sua tabela com o endereço MAC DD-DD-DD-DD-DD-DD. Existem três casos possíveis:

- Não existe entrada na tabela para DD-DD-DD-DD-DD-DD. Nesse caso, o comutador encaminha anteriormente cópias do quadro para os buffers de saída a todas as interfaces, exceto a interface x. Ou seja, se não existe entrada para o endereço de destino, o comutador transmite o quadro em broadcast.
- Existe uma entrada na tabela, associando DD-DD-DD-DD-DD-DD com a interface x. Não existe necessidade de encaminhar o quadro para qualquer outra interface, o comutador realiza a função de filtragem ao descartar o quadro.
- Existe uma entrada na tabela, associando DD-DD-DD-DD-DD-DD com a interface $x \neq y$. Nesse caso o quadro precisa ser encaminhado ao segmento da LAN anexado à interface y. O comutador realiza sua função de encaminhamento ao colocar o quadro em um buffer de saída que precede a interface y.

Vamos examinar essas regras para a rede da Figura 5.26 e sua tabela de comutação na Figura 5.27 Suponha que um quadro com endereço de destino 62-FE-F7-11-89-A3 chegue ao comutador vindo da interface 1. O comutador examina sua tabela e vê que o destino está no segmento de LAN conectado à interface 1 (isto é, do departamento de engenharia elétrica). Isso significa que o quadro já foi transmitido por broadcast no segmento de LAN que contém o destino. Por conseguinte, o comutador filtra (isto é, descarta) o quadro. Agora suponha que um quadro com o mesmo endereço de destino chegue da interface 2. O comutador novamente examina sua tabela e verifica que o destino está na direção da interface 1; por conseguinte, ele repassa o quadro para o buffer de saída que precede a interface 1. Deve ficar claro, por esse exemplo, que, enquanto a tabela de comutação permanecer completa e precisa, o comutador encaminha quadros até seus destinos sem nenhuma transmissão.

Assim, o comutador é “mais esperto” do que um hub. Mas como uma tabela de comutação é configurada afinal? Existem equivalentes de camadas de enlace a protocolos de roteamento de camada de rede? Ou um gerente cansado deve configurar manualmente a tabela de comutação?

Aprendizagem automática

Um comutador tem a maravilhosa propriedade (em especial, para o administrador de rede, que quase sempre está sobrecarregado) de montar sua tabela automática, dinâmica e autonomamente — sem nenhuma intervenção de um administrador de rede ou de um protocolo de configuração. Em outras palavras, comutadores são **autodidata**s. Essa capacidade é conseguida como segue:

1. A tabela de comutação inicialmente está vazia.
2. Para cada quadro recebido em uma interface, o comutador armazena em sua tabela (1) o endereço MAC que está no campo de endereço de fonte do quadro, (2) a interface da qual veio o quadro e (3) o horário corrente. Dessa maneira, o comutador registra em sua tabela o segmento MAC no qual reside o nó remetente. Se cada nó da LAN finalmente enviar um quadro, então cada nó será finalmente registrado na tabela.
3. O comutador apagará um endereço na tabela se nenhum quadro que tenha aquele endereço como endereço de fonte for recebido após um certo período de tempo (o **tempo de envelhecimento**). Desse modo, se um PC for substituído por outro PC (com um adaptador diferente), o endereço MAC do PC original acabará sendo expurgado da tabela de comutação.

Vamos examinar a propriedade de aprendizagem automática para a rede da Figura 5.28 e sua tabela de comutação correspondente apresentada na Figura 5.27. Suponha que no horário 9h39 um quadro com endereço

de fonte 01-12-23-34-45-56 venha da interface 2. Suponha também que esse endereço não esteja na tabela de comutação. Então, o comutador anexa um novo registro à tabela, como mostra a Figura 5.28.

Continuando com esse mesmo exemplo, suponha ainda que o tempo de envelhecimento para esse comutador seja 60 minutos e que nenhum quadro com endereço de fonte 62-FE-F7-11-89-A3 chegue ao comutador entre 9h32 e 10h32. Então, no horário 10h32, o comutador remove esse endereço de sua tabela.

Comutadores são **dispositivos do tipo plug-and-play** porque não requerem a intervenção de um administrador de rede ou de um usuário. Um administrador de rede que quiser instalar um comutador não precisa fazer nada mais do que conectar os segmentos de LAN às interfaces do comutador. O administrador não precisa configurar as tabelas de comutação na hora da instalação nem quando um hospedeiro é removido de um dos segmentos de LAN. Comutadores também são full-duplex, o que significa que para qualquer enlace conectando um nó a um comutador, o nó e o comutador podem transmitir ao mesmo tempo sem colisões.

5.6.3 Propriedades de comutação da camada de enlace

Tendo descrito as operações básicas da comutação da camada de enlace, vamos considerar suas propriedades e funcionalidades. Usando a ilustração de uma LAN na Figura 5.24, podemos identificar diversas vantagens em se usar comutadores, em vez de se usar enlaces broadcast como barramentos ou hub com topologias de estrela:

Eliminação de colisões: Em uma LAN montada com comutadores (e sem hubs), não existe desperdício de banda devido a colisões! Os comutadores armazenam os quadros e nunca transmitem mais de um quadro em um segmento ao mesmo tempo. Como em um roteador, a vazão máxima agregado de um comutador é a soma da velocidade de todas interfaces do comutador. Portanto, o switch obtém uma melhora de desempenho significativa em relação aos meios broadcasts.

Enlaces heterogêneos: Sendo que o comutador isola um enlace do outro, os diferentes enlaces na LAN podem operar em diferentes velocidades e podem ser executados através de diferentes mídias. Na Figura 5.24, por exemplo, A pode ser conectado a 10 Mbps sobre cobre 100BASE-T,

Management. In addition to providing enhanced security (see sidebar on Focus on Security), a switch also eases network management. For example, if an adapter malfunctions and continually sends Ethernet frames (called a jabbering adapter), a switch can detect the problem and internally disconnect the malfunctioning adapter. With this feature, the network administrator need not get out of bed and drive back to work in order to correct the problem. Similarly, a cable cut disconnects only that node that was using the cut cable to connect to the switch. In the days of coaxial cable, many a network manager spent hours “walking the line” (or more accurately, “crawling the floor”) to find the cable break that brought down the entire network. As discussed in Chapter 9 (Network Management), switches also gather statistics on bandwidth usage, collision rates, and traffic types, and make this information available to the network manager. This information can be used to debug and correct problems, and to plan how the LAN should evolve in the future. Researchers are exploring adding yet more management functionality into Ethernet LANs in prototype deployments [Casado 2007].

5.6.4 Comutadores versus roteadores

Como aprendemos no Capítulo 4, roteadores são comutadores de pacotes do tipo armazena-e-repassa que transmitem pacotes usando endereços de camada de rede. Embora um comutador também seja um comutador

Endereço	Interface	Horário
01-12-23-34-45-56	2	9h39
62-FE-F7-11-89-A3	1	9h32
7C-BA-B2-B4-91-10	3	9h36
....

Figura 5.28 O comutador aprende a localização do adaptador com endereço 01-12-23-34-45-56

de pacotes do tipo armazena-e-repassa, ele é fundamentalmente diferente de um roteador, pois repassa pacotes usando endereços MAC. Enquanto um roteador é um comutador de pacotes de camada 3, um comutador opera com protocolos de camada 2.

Mesmo sendo fundamentalmente diferentes, é comum que os administradores de rede tenham de optar entre um comutador e um roteador ao instalar um dispositivo de interconexão. Por exemplo, para a rede da Figura 5.26, o administrador de rede poderia facilmente ter optado por usar um roteador, em vez de um comutador, para conectar o departamento de LANs, servidores e roteador de internet. Na verdade, um roteador também teria mantido separados os três domínios de colisão, permitindo, ao mesmo tempo, a comunicação interdepartamental. Dado que ambos, comutadores e roteadores, são candidatos a dispositivos de interconexão, quais são os prós e os contras das duas abordagens?

Vamos considerar, inicialmente, os prós e os contras de comutadores. Como mencionamos antes, comutadores são do tipo plug-and-play, uma propriedade que é apreciada por todos os administradores de rede atarefados do mundo. Comutadores também podem ter velocidades relativamente altas de filtragem e repasse — como mostra a Figura 5.29, eles têm de processar quadros apenas até a camada 2, enquanto roteadores têm de processar pacotes até a camada 3. Por outro lado, para evitar a ciclagem da transmissão de quadros, a topologia de uma rede de comutação está restrita a uma *spanning tree*. E mais, uma rede de comutação de grande porte exigiria, nos nós, tabelas ARP também de grande porte e geraria tráfego e processamento ARP substanciais. Além do mais, comutadores não oferecem nenhuma proteção contra tempestades de *broadcast* — se um hospedeiro se desorganiza e transmite uma corrente sem fim de quadros Ethernet *broadcast*, os comutadores repassam todos esses quadros, causando o colapso da rede inteira.

Agora, vamos considerar os prós e os contras dos roteadores. Como na rede o endereçamento muitas vezes é hierárquico (e não linear, como o endereçamento MAC), os pacotes normalmente não ficam circulando nos roteadores, mesmo quando a rede tem trajetos redundantes. (Na verdade, eles podem circular quando as tabelas de roteadores estão mal configuradas; mas, como aprendemos no Capítulo 4, o IP usa um campo de cabeçalho de datagrama especial para limitar a circulação.) Assim, pacotes não ficam restritos a uma topologia de *spanning tree* e podem usar o melhor trajeto entre fonte e destino. Como roteadores não sofrem a limitação da topologia *spanning tree*, em árvore, eles permitiram que a Internet fosse montada com uma topologia rica que inclui, por exemplo, múltiplos enlaces ativos entre a Europa e a América do Norte. Outra característica dos roteadores é que eles fornecem proteção de firewall contra as tempestades de *broadcast* de camada 2. Talvez a desvantagem mais significativa dos roteadores seja o fato de que eles não são do tipo plug-and-play — eles e os hospedeiros que a eles se conectam precisam de seus endereços IP para ser configurados. Além disso, roteadores muitas vezes apresentam tempo de processamento por pacote maior do que comutadores porque têm de processar até os campos da camada 3. Finalmente, há dois modos diferentes de pronunciar a palavra ‘router’: ‘rúter’ ou ‘ráuter’, e as pessoas perdem muito tempo discutindo qual é a melhor pronúncia [Perlman, 1999].

Dado que comutadores e roteadores têm seus prós e contras (como foi resumido na Tabela 5.1), quando uma rede institucional (por exemplo, uma rede de um campus universitário ou uma rede empresarial) deveria usar comutadores e quando deveria usar roteadores? Em geral, redes pequenas, com algumas centenas de hospedeiros, têm uns poucos

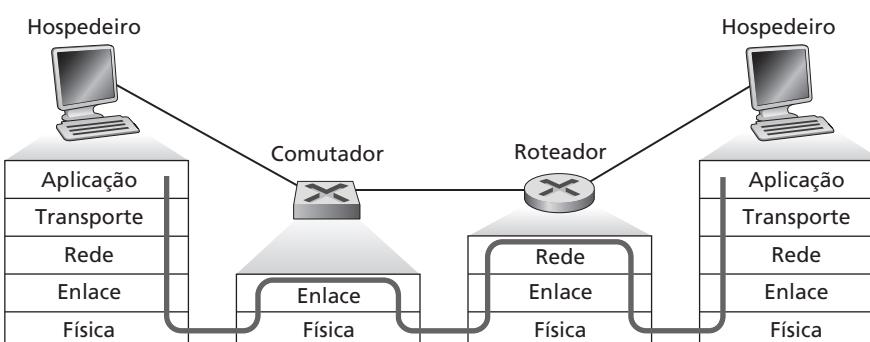


Figura 5.29 Processamento de pacotes em comutadores, roteadores e hospedeiros

	Hubs	Roteadores	Comutadores
Isolamento de tráfego	não	sim	sim
Plug-and-play	sim	não	sim
Roteamento ótimo	não	sim	não

Tabela 5.1 Comparação entre as características típicas de dispositivos de interconexão populares

segmentos de LAN. Para essas redes pequenas comutadores serão satisfatórios, pois localizam o tráfego e aumentam a vazão agregada sem exigir nenhuma configuração de endereços IP. Mas redes maiores, com milhares de hospedeiros, tipicamente incluem roteadores na rede (além de comutadores). Roteadores fornecem isolamento de tráfego mais robusto, controlam tempestades de *broadcast* e usam rotas ‘mais inteligentes’ entre os hospedeiros da rede.

Para maiores informações sobre os prós e os contras das redes de comutadores e das redes de roteadores, assim como informações sobre como a tecnologia de LAN comutada pode ser estendida para servir duas ordens de magnitude de hospedeiros a mais que a Ethernet atual, veja [Kim, 2008].

5.6.5 Rede Local Virtual (VLANs [Virtual Local Area Network])

Na nossa discussão anterior sobre a Figura 5.26, notamos que as LANs institucionais modernas são frequentemente configuradas hierarquicamente, com cada grupo de trabalho (departamento) com seu próprio comutador de LAN conectado ao comutador de LAN de outros grupos via uma hierarquia de comutadores. Enquanto tais configurações funcionam bem em um mundo ideal, o mundo real é bem diferente. Três desvantagens podem ser identificadas na configuração da Figura 5.26:

Falta de isolamento do tráfego: Apesar de a hierarquia localizar o tráfego de grupos dentro de um único comutador, o tráfego broadcast (por exemplo, quadros carregando mensagens ARP e DHCP ou quadros com endereços de destino que ainda não foram apreendidas por um comutador com aprendizagem automática) tem que ainda percorrer toda a rede institucional. Limitar o escopo desse tráfego de broadcast aprimoraria o desempenho da LAN. Talvez mais importante que isso, também seria desejável limitar o tráfego broadcast da LAN por razões de segurança e privacidade. Por exemplo, se um grupo contém o time de gerência executiva de uma empresa e outro grupo contém funcionários decepcionados executando o analisador de pacote Wireshark, o gerente de rede pode preferir que o tráfego da gerência executiva não alcance os computadores dos funcionários. Esse tipo de isolamento pode ser substituído trocando o comutador central da Figura 5.26 por um roteador. Em breve veremos que esse isolamento também pode ser realizado via uma solução de comutação (camada 2).

Uso ineficiente de comutadores. Se em vez de três grupos, a instituição tivesse 10 grupos, os 10 comutadores de primeiro nível seriam necessários. Se cada grupo fosse pequeno, com menos de 10 pessoas, um comutador de 96 pontos seria suficiente para atender a todos, mas esse único comutador não fornece isolamento de tráfego.

Gerenciamento de usuários. Se um funcionário se locomove entre os grupos o cabeamento físico deve ser mudado para conectar o funcionário a um comutador diferente na Figura 5.26. Funcionários pertencentes a dois grupos dificultam o problema.

Felizmente, cada uma dessas dificuldades pode ser resolvida com um comutador que suporte uma Rede Local Virtual (VLANs).

Como o nome já sugere, um comutador que suporta VLANs permite que diversas redes locais virtuais sejam implementadas através de uma única infraestrutura física de uma rede local virtual. Hospedeiros dentro de uma VLAN se comunicam como se eles (e não outros hospedeiros) estivessem conectados ao comutador. Em uma VLAN baseada em pontos, as interfaces do comutador são divididas em grupos pelo gerente da rede. Cada grupo constitui uma VLAN,

com as interfaces em cada VLAN formando um domínio de broadcast (por exemplo, o tráfego de broadcast de uma interface só pode alcançar outras interfaces no grupo). A Figura 5.30 mostra um único comutador com 16 interfaces. As interfaces de 2 a 8 pertencem a EE VLAN, enquanto as interfaces de 9 a 15 pertencem a CS VLAN (interfaces de 1 e 16 não são atribuídas). Essa VLAN soluciona todas as dificuldades citadas acima — quadros EE e CS VLAN são isolados uns dos outros, os dois comutadores na Figura 5.26 foram substituídos por um único comutador, e se o usuário da interface 8 se juntar ao Departamento CS, o operador da rede simplesmente reconfigura o software da VLAN para que a interface 8 seja associada com CS VLAN. Qualquer um poderia imaginar facilmente como o comutador VLAN opera e é configurado — o gerente de rede declara uma interface pertencente a uma dada VLAN (com interfaces não declaradas pertencentes à VLAN padrão) usando um software de gerenciamento de comutação, uma tabela de interface de mapeamento VLAN é mantida dentro do comutador; e um hardware de comutação somente entrega quadros entre as interfaces pertencentes à mesma VLAN.

Mas isolando as duas VLANs, criamos um novo problema! Como o tráfego do Departamento EE pode ser enviado para o Departamento CS? Uma maneira de se lidar com isso seria conectar uma interface de comutação VLAN (exemplo, interface 1 na Figura 5.30) a um roteador externo, configurando aquela interface para que pertença a ambos departamentos EE e CS VLANs. Nesse caso, apesar de os Departamentos EE e CS compartilharem um mesmo roteador físico, a configuração faria parecer que os Departamentos EE e CS têm comutadores diferentes conectados por um roteador. Um datagrama IP, indo do departamento EE para o CS, iria passar primeiramente pela VLAN EE para alcançar o roteador e depois ser encaminhado de volta pelo roteador através da VLAN CS até o hospedeiro CS. Felizmente, os fornecedores de comutadores fazem com que as configurações sejam fáceis para o gerente de rede. Eles montam um dispositivo único que contém um comutador VLAN e um roteador, para que um roteador externo não seja necessário. Uma lição de casa ao final do capítulo explora esse exemplo mais detalhadamente.

Voltando à Figura 5.26, vamos supor que em vez de termos um departamento separado de Engenharia da Computação, parte do corpo docente de EE e de CS estejam alojado em um prédio separado, aonde (é claro!) eles precisariam de acesso à rede, e (é claro!) gostariam de fazer parte do VLAN do seu desempenho. A Figura 5.31 mostra um segundo comutador com 8 entradas, onde as entradas do comutador foram definidas como pertencentes a VLAN EE ou CS, conforme necessário. Mas como esses dois comutadores seriam interconectados? Uma solução fácil seria definir uma entrada como pertencente à VLAN CS em cada comutador (semelhantemente à VLAN EE) e conectar essas entradas umas às outras, como demonstrado na Figura 5.31(a). No entanto, essa solução não permite crescimento, já que N VLANs exigiriam portas N em cada comutador para simplesmente interconectar os dois comutadores.

Uma abordagem mais escalável para interconectar os comutadores das VLANs é conhecido como Entroncamento de VLANs. Na abordagem do Entroncamento de VLANs, demonstrada na Figura 5.31 (b), uma porta especial em cada comutador (interface 16 no comutador esquerdo e interface 1 no comutador direito) é configurada como uma porta de tronco para interconectar os dois comutadores VLANs. A porta de tronco pertence a todas VLANs, e quadros enviados a qualquer VLAN são encaminhados através do enlace de tronco ao

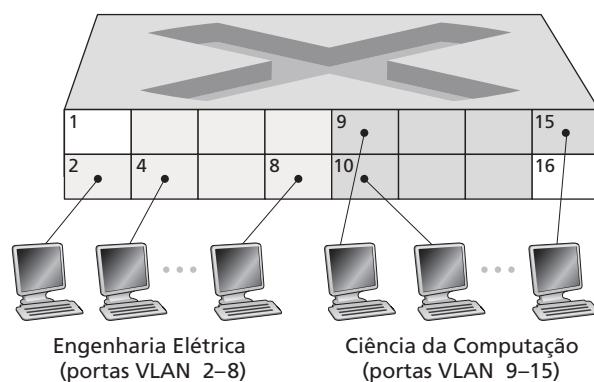


Figura 5.30 Um único comutador com duas VLANs configuradas

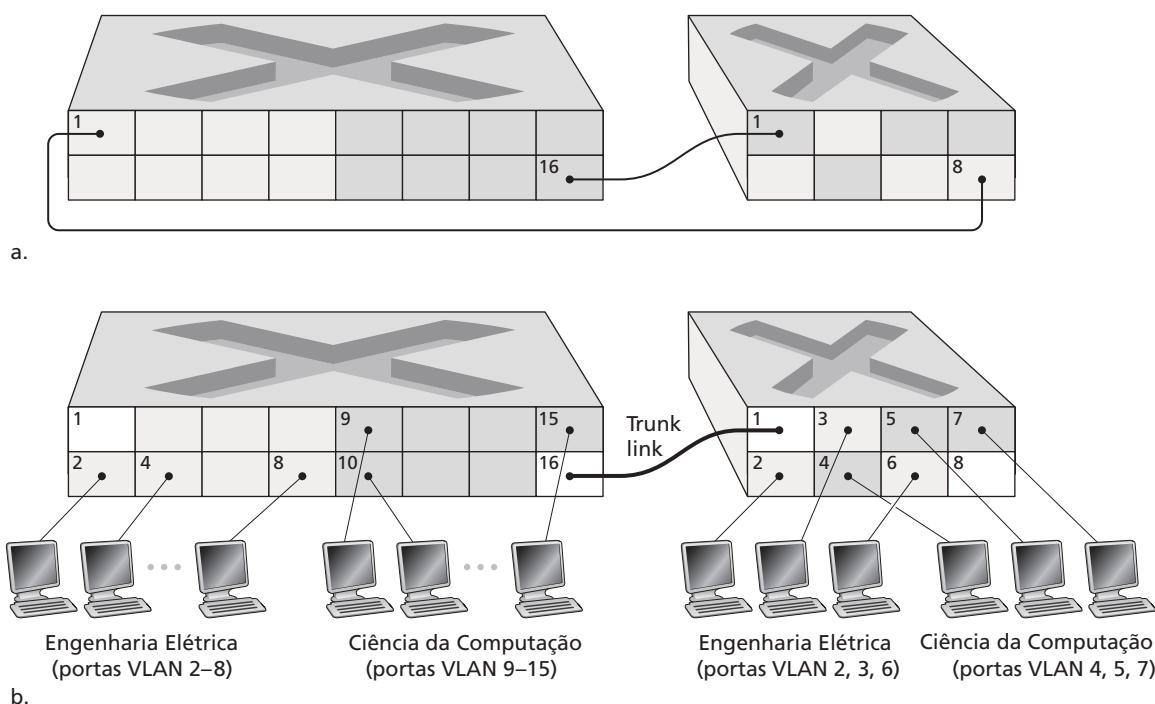


Figura 5.31 Conectando 2 comutadores VLANs com duas VLANs: (a) 2 cabos (b) entrонcados

outro comutador. Mas isso gera mais uma dúvida: como um comutador “sabe” que um quadro que está chegando a uma porta de tronco pertence a uma VLAN específica? O IEEE definiu um formato de quadro estendido, 802.1Q, para quadros atravessando o tronco VLAN. Como demonstrado na Figura 5.32, o quadro 802.1Q consiste no quadro padrão Ethernet com um rótulo de VLAN de quatro bytes adicionados no cabeçalho que carrega a identidade da VLAN à qual o quadro pertence. O rótulo da VLAN é adicionado ao quadro pelo comutador no lado de envio do tronco de VLAN, analisado, e removido pelo comutador no lado de recebimento do tronco. O rótulo da VLAN consiste propriamente em um campo de 2 bytes chamado Rótulo de Identificação de Protocolo [TPID] (com um valor hexadecimal de 81-00 fixo), um campo de 2 bytes de Controle de Informação de Rótulo contendo um campo de identificação de VLAN com 12 bits, e um campo de prioridade com 3 bits similar em propósito ao campo TOS do datagrama IP.

Nesta análise, só fizemos uma breve citação sobre VLANs e focamos em VLANs baseadas em portas. Deveríamos mencionar também que as VLANs podem ser definidas de diversas maneiras. Em uma VLAN baseada em MAC, o administrador de rede especifica o grupo de endereços MAC que pertence a cada VLAN; quando um dispositivo é conectado a uma porta, a porta é conectada na VLAN apropriada baseada no endereço MAC do

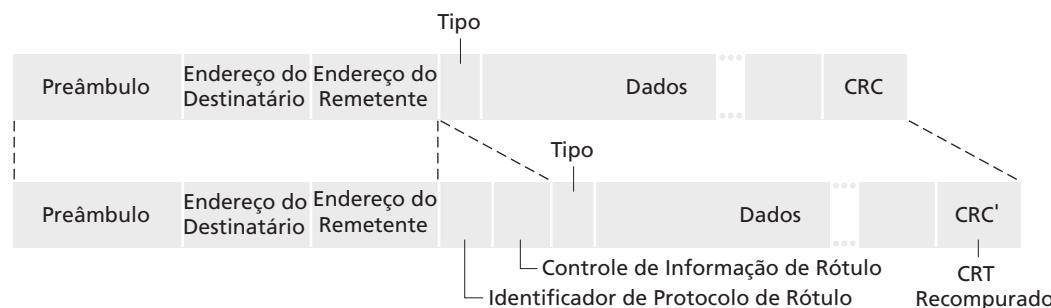


Figura 5.32 Quadro Ethernet (acima): quadro VLAN 802.1Q — tagged Ethernet

dispositivo. As VLANs também podem ser definidas por protocolos de camadas de rede (por exemplo, IPv4, IPv6 ou Appletalk) e outros critérios. Veja o padrão 802.1Q[IEEE802.1q2005] para maiores informações.

5.7 PPP: o protocolo ponto a ponto

Grande parte de nossa discussão sobre protocolos de camada de enlace esteve voltada, até aqui, aos protocolos para canais *broadcast*. Nesta seção, examinaremos um protocolo de camada de dados para enlaces ponto a ponto: o PPP. Como o PPP é o protocolo comumente escolhido para o enlace discado de hospedeiros residenciais, ele é, sem dúvida, um dos protocolos de camada de enlace mais utilizados hoje em dia. O outro importante protocolo de camada de dados em uso atualmente é o protocolo HDLC (*high-level data link control* — controle de enlace de dados de alto nível); consulte [Spragins, 1991] para detalhes sobre o HDLC. Nossa discussão do protocolo PPP, um protocolo mais simples, nos permitirá explorar muitas das mais importantes características de um protocolo ponto a ponto de camada de enlace.

Como o nome indica, o PPP [RFC 1661; RFC 2153] é um protocolo de camada de enlace que opera sobre um **enlace ponto a ponto** — um enlace que conecta diretamente dois nós, um em cada extremidade do enlace. O enlace ponto a ponto sobre o qual o PPP opera pode ser uma linha telefônica discada serial (por exemplo, uma conexão de modem de 56K), um enlace SONET/SDH, uma conexão X.25 ou um circuito ISDN (*integrated service digital network* — rede digital de serviço integrado). Como observado antes, o PPP se tornou o protocolo preferido para conectar usuários residenciais a seus ISPs por meio de uma conexão discada.

Antes de apresentarmos os detalhes do PPP, é instrutivo examinar as exigências originais da IETF para o projeto do PPP [RFC 1547]:

Enquadramento do pacote. No protocolo PPP de camada de enlace, o remetente deve ser capaz de pegar um pacote da camada de rede e encapsulá-lo dentro do quadro PPP da camada de enlace de tal modo que o receptor possa identificar o início e o fim do quadro de camada de enlace e também do pacote da camada de rede que está dentro do quadro.

Transparência. O protocolo PPP não deve impor nenhuma restrição sobre os dados que aparecem no pacote da camada de rede (cabeçalhos ou dados). Assim, por exemplo, ele não pode proibir a utilização de certos padrões de bits no pacote da camada de rede. Voltaremos a esse assunto em breve, quando discutirmos enchimento de bytes.

Múltiplos protocolos de camada de rede. O protocolo PPP deve estar habilitado a suportar múltiplos protocolos de camada de rede (por exemplo, IP e DECnet) que executam sobre o mesmo enlace físico e ao mesmo tempo. Exatamente como o protocolo IP tem de multiplexar diferentes protocolos de nível de transporte (por exemplo, TCP e UDP) sobre uma única conexão fim a fim, o PPP também deve estar habilitado a multiplexar diferentes protocolos de camada de rede sobre uma única conexão ponto a ponto. Essa exigência significa que, no mínimo, o PPP provavelmente necessitará de um campo de ‘tipo de protocolo’ ou de algum mecanismo similar para que seu lado receptor possa demultiplexar um quadro recebido até o protocolo de camada de rede apropriado.

Múltiplos tipos de enlaces. Além de poder carregar múltiplos protocolos de nível mais alto, o PPP deve também poder operar sobre uma grande variedade de tipos de enlaces, incluindo enlaces que são seriais (transmitem um bit por vez em uma dada direção), paralelos (transmitem bits em paralelo), síncronos (transmitem um sinal de relógio juntamente com os dados de bits) ou assíncronos, de baixa velocidade ou de alta velocidade, elétricos ou óticos.

Detecção de erros. Um receptor PPP deve estar habilitado a detectar erros de bits no quadro recebido.

Vida da conexão. O PPP deve estar habilitado a detectar uma falha no nível de enlace (por exemplo, a incapacidade de transferir dados do lado remetente do enlace para o lado receptor) e sinalizar essa condição de erro à camada de rede.

Negociação do endereço de camada de rede. O PPP deve fornecer um mecanismo para que as camadas de rede comunicantes (por exemplo, IP) aprendam ou configurem mutuamente seus endereços de camada de rede.

Simplicidade. Além de atender às características relacionadas, o PPP tem de atender a uma série de requisitos adicionais. E, como se tudo isso não bastasse, a exigência mais primordial e mais importante é a simplicidade. O RFC 1547 estabelece que “a marca de um protocolo ponto a ponto deve ser a simplicidade”. Uma ordem realmente difícil de ser cumprida, dadas todas as outras exigências feitas quanto ao projeto do PPP! Mais de 100 RFCs definem atualmente os vários aspectos desse protocolo ‘simples’.

Embora possa parecer que tenham sido muitas as exigências impostas ao projeto do PPP, na verdade a situação poderia ter sido muito pior! As especificações de projeto do PPP também estabeleceram explicitamente as funcionalidades de protocolo que o PPP *não* é obrigado a implementar.

Correção de erros. Exige-se que o PPP detecte erros de bits, mas não se exige que ele os corrija.

Controle de fluxo. Espera-se que um receptor PPP possa receber quadros à velocidade total da camada física subjacente. Se uma camada superior não puder receber pacotes a essa velocidade total, então será problema da camada superior descartar os pacotes ou regular o remetente na camada superior. Isto é, em vez de fazer com que o remetente PPP regule sua própria velocidade de transmissão, é responsabilidade de um protocolo de camada superior regular a velocidade na qual os pacotes são entregues ao PPP para envio.

Sequenciamento. Não se exige que o PPP entregue quadros ao enlace receptor na mesma ordem em que foram enviados pelo enlace remetente. É interessante notar que, embora essa flexibilidade seja compatível com o modelo de serviço IP (que permite que pacotes IP sejam entregues fim a fim em qualquer ordem), outros protocolos de camada de rede que operam sobre PPP exigem entrega de pacotes fim a fim sequenciada.

Enlaces multiponto. O PPP precisa operar apenas sobre enlaces que têm um único remetente e um único receptor. Outros protocolos de camada de enlace (por exemplo, o HDLC) podem atender a múltiplos receptores (por exemplo, em um cenário semelhante ao da Ethernet) em um enlace.

Agora que consideramos as metas de projeto (e as ‘não-metas’) para o PPP, vamos ver como o projeto do PPP atinge essas metas.

5.7.1 Enquadramento de dados PPP

A Figura 5.33 mostra um quadro de dados PPP que utiliza um enquadramento semelhante ao do HDLC [RFC 1662]. O quadro PPP contém os seguintes campos:

Campo de flag. Todo quadro PPP começa e termina com um campo de flag de um byte de valor 01111110.

Campo de endereço. O único valor possível para esse campo é 11111111.

Campo de controle. O único valor possível para esse campo é 00000011. Como os campos de endereço e de controle somente podem assumir um valor fixo, é de se perguntar, antes de mais nada, por que foram definidos. A especificação PPP [RFC 1662] estabelece que outros valores “poderão ser definidos mais tarde”, embora nenhum tenha sido definido até agora. Como os campos assumem valores fixos, o PPP permite que o remetente simplesmente não envie os bytes de endereço e de controle, economizando, dessa maneira, 2 bytes de sobrecarga no quadro PPP.

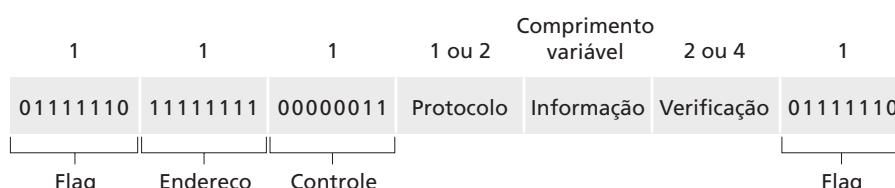


Figura 5.33 Formato do quadro de dados PPP

Protocolo. O campo de protocolo diz ao receptor PPP a qual protocolo de camada superior pertencem os dados encapsulados recebidos (isto é, o conteúdo do campo de informação do quadro PPP). Ao receber um quadro PPP, o receptor PPP verifica se o quadro está correto e, então, repassa os dados encapsulados ao protocolo apropriado. O RFC 1700 e o RFC 3232 definem os códigos de protocolo de 16 bits usados pelo PPP. São de nosso interesse o protocolo IP (isto é, os dados encapsulados no quadro PPP em um datagrama IP), que tem um valor hexadecimal de 21, outros protocolos de camada de rede como o AppleTalk (29) e o DECnet (27).

Informação. Esse campo contém o pacote encapsulado (dados) que está sendo enviado por um protocolo de camada superior (por exemplo, IP) pelo enlace PPP. O comprimento máximo padrão do campo de informação é 1.500 bytes, embora isso possa ser mudado quando o enlace é configurado pela primeira vez, como discutido a seguir.

Soma de verificação. O campo de soma de verificação é usado para detectar erros de bits em um pacote transmitido. Ele usa um código de redundância cíclica padrão HDLC de 2 ou 4 bytes.

Byte Stuffing

Antes de encerrarmos nossa discussão sobre o enquadramento PPP, vamos considerar um problema que surge quando qualquer protocolo usa um padrão específico de bits em um campo de flag para indicar o começo ou o fim do quadro. O que acontece se o padrão do flag ocorre em outro local do pacote? Por exemplo, o que acontece se o valor do campo de flag 01111110 aparece no campo de informação? O receptor vai detectar incorretamente o final do quadro PPP?

Uma das maneiras de resolver esse problema seria o PPP proibir que o protocolo de camada superior enviasse dados com o padrão de bits do campo de flag. A exigência de transparência do PPP, discutida anteriormente, impede essa possibilidade. Uma solução alternativa, que é usada pelo PPP e por muitos outros protocolos, é a técnica conhecida como **byte stuffing**.

O PPP define um byte de controle especial de escape, 01111101. Se a sequência do flag 01111110 aparecer em outro lugar do quadro, exceto no campo do flag, o PPP precederá aquele exemplar do padrão de flag com o byte de controle de escape. Em outras palavras, ele ‘recheia’ (adiciona) com um byte de controle de escape a sequência de dados transmitida, antes de 01111110, para mostrar que o 01111110 seguinte não é um valor de flag, mas sim dados. Um receptor que vir um 01111110 precedido de um 01111101 removerá, é claro, o byte de controle de escape introduzido para reconstruir os dados originais. De maneira semelhante, se o próprio padrão do byte de controle de escape aparecer como dados, ele também deverá ser precedido por um byte de controle de escape introduzido. Assim, quando o receptor vir um único byte de controle de escape isolado na corrente de dados, ele saberá que o byte foi introduzido na corrente de dados. Um par de bytes de controle de escape que ocorrem um atrás do outro significa que um exemplar do byte de controle de escape aparece nos dados originais que estão sendo enviados. A Figura 5.34 ilustra o byte stuffing PPP. (Na realidade, o PPP também realiza uma operação *ou exclusivo* entre o byte de dados afetado pelo caractere de escape e o valor 20 hexadecimal, um detalhe

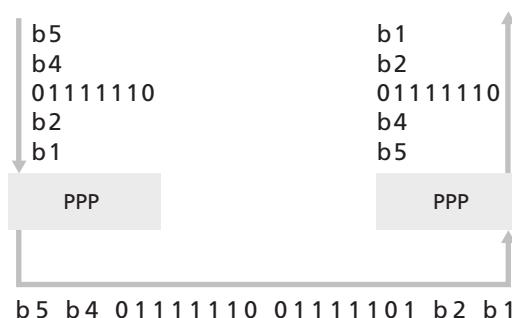


Figura 5.34 Byte stuffing



que omitimos aqui para simplificar). Observamos também que o PPP tem um protocolo de controle de enlace (LCP), sendo sua função a de fazer a inicialização, manutenção e o encerramento de um enlace PPP. O LCP é abordado com maiores detalhes no material online associado a este livro.

5.8 Virtualização de enlace: uma rede como camada de enlace

Como este capítulo trata de protocolos de camada de enlace, e dado que estamos chegando ao fim do capítulo, vamos refletir um pouco sobre como evoluiu o que entendemos como *enlace*. Começamos este capítulo considerando um enlace como um fio físico conectando dois hospedeiros comunicantes, como ilustrado na Figura 5.2. Quando estudamos protocolos de acesso múltiplo (Figura 5.9), vimos que vários hospedeiros podiam ser conectados por um fio compartilhado e que o ‘fio’ que conectava os hospedeiros podia ser o espectro de rádio ou qualquer outro meio. Isso nos levou a considerar o enlace, de um modo um pouco mais abstrato, como um canal, em vez de um fio. Quando estudamos LANs Ethernet (Figuras 5.26-5.31) vimos que, na verdade, os meios de interconexão poderiam ser uma infraestrutura de comutação bastante complexa. Durante toda essa evolução, entretanto, os hospedeiros sempre mantiveram a visão do meio de conexão simplesmente como um canal de camada de enlace conectando dois ou mais hospedeiros. Vimos, por exemplo, que um hospedeiro Ethernet pode ficar alegremente inconsciente do fato de estar ligado a outros hospedeiros de LAN por um único segmento curto de LAN (Figura 5.9) ou por uma LAN comutada geograficamente dispersa (Figura 5.26) ou pela VLAN (Figura 5.31).

Na Seção 5.7 vimos que o protocolo PPP é frequentemente usado sobre uma conexão de modem entre dois hospedeiros. Nesse caso, o enlace que conecta os dois hospedeiros é, na verdade, a rede de telefonia — uma rede global de telecomunicações logicamente separada, com seus próprios comutadores, enlaces e pilhas de protocolos para transferência e sinalização de dados. Entretanto, da perspectiva da camada de rede da Internet, a conexão discada por meio da rede de telefonia é vista como um simples ‘fio’. Nesse sentido, a Internet virtualiza a rede de telefonia, considerando-a como uma tecnologia de camada de enlace que provê conectividade de camada de enlace entre dois hospedeiros da Internet. Lembre-se de que, quando discutimos redes de sobreposição no Capítulo 2, dissemos que, de modo semelhante, essa rede vê a Internet como um meio de prover conectividade entre nós sobrepostos, procurando se sobrepor à Internet do mesmo modo que a Internet se sobrepõe à rede de telefonia.

Nesta seção consideraremos redes MPLS (*Multiprotocol Label Switching*). Diferentemente da rede de telefonia de comutação de circuitos, as redes MPLS, são, de direito, redes de comutação de pacotes por circuitos virtuais. Elas têm seus próprios formatos de pacotes e comportamentos de repasse. Assim, de um ponto de vista pedagógico, é bem coerente discutir MPLS quando estudamos camada de rede ou camada de enlace. Todavia, da perspectiva da Internet, podemos considerar a MPLS, assim como a rede de telefonia e a Ethernet comutada, como tecnologia de camada de enlace que serve para interconectar dispositivos IP. Assim, consideremos as redes MPLS ao discutirmos a camada de enlace. Redes frame-relay e ATM também podem ser usadas para interconectar dispositivos IP, embora representem uma tecnologia ligeiramente mais antiga (mas ainda disponibilizada), que não será discutida aqui; se quiser saber mais detalhes consulte o livro, de fácil leitura, de [Goralski, 1999]. Nossa estudo de MPLS será necessariamente breve, pois livros inteiros podem ser escritos (e foram) sobre essas redes. Recomendamos [Davie, 2000] para detalhes sobre MPLS. Aqui, focalizaremos primordialmente como essas redes servem para interconectar dispositivos IP, embora as tecnologias subjacentes também sejam examinadas com um pouco mais de profundidade.

Comutação de Rótulos Multiprotocolo (MPLS)

A Comutação de Rótulos Multiprotocolo (MPLS) evoluiu dos inúmeros esforços realizados pela indústria de meados ao final da década de 1990 para melhorar a velocidade de repasse de roteadores IP adotando um conceito fundamental do mundo das redes de circuitos virtuais: um rótulo de tamanho fixo. O objetivo não era abandonar a infraestrutura de repasse de datagramas IP com base no destino em favor de rótulos de tamanho fixo e circuitos virtuais, mas aumentá-la rotulando datagramas seletivamente e permitindo que roteadores repassassem datagramas com base em rótulos de tamanho fixo (em vez de endereços de destino IP), quando possível. O importante é que essas técnicas trabalhavam de mãos dadas com o IP, usando endereçamento e roteamento IP. A IETF reuniu

esses esforços no protocolo MPLS [RFC 3031, RFC 3032] misturando efetivamente técnicas de circuitos virtuais em uma rede de datagramas com roteadores.

Vamos começar nosso estudo do MPLS considerando o formato de um quadro de camada de enlace que é manipulado por um roteador habilitado para MPLS. A Figura 5.35 mostra que um quadro de camada de enlace transmitido por um enlace PPP ou LAN (tal como a Ethernet) tem um pequeno cabeçalho MPLS adicionado entre o cabeçalho de camada 2 (isto é, PPP ou Ethernet) e o cabeçalho de camada 3 (isto é, IP). O RFC 3032 define o formato do cabeçalho MPLS para esses enlaces; cabeçalhos de redes ATM e de frame relay também são definidos em outros RFCs. Entre os campos no cabeçalho MPLS estão o rótulo (que desempenha o papel de identificador de circuito virtual que estudamos na Seção 4.2.1), 3 bits reservados para uso experimental, um único bit, S, que é usado para indicar o final de uma série de rótulos MPLS ‘empilhados’ (um tópico avançado que não será abordado aqui), e um campo de tempo de vida.

A Figura 5.35 deixa imediatamente evidente que um quadro melhorado com MPLS somente pode ser enviado entre roteadores habilitados para MPLS (já que um roteador não habilitado para MPLS ficaria bastante confuso ao encontrar um cabeçalho MPLS onde esperava encontrar o cabeçalho IP!). Um roteador habilitado para MPLS é comumente denominado **roteador de comutação de rótulos**, pois repassa um quadro MPLS consultando o rótulo MPLS em sua tabela de repasse e, então, passando imediatamente o datagrama para a interface de saída apropriada. Assim, o roteador habilitado para MPLS não precisa extraír o endereço de destino e executar uma busca para fazer a compatibilização com o prefixo mais longo na tabela de repasse. Mas como um roteador sabe se seu vizinho é realmente habilitado para MPLS e como sabe qual rótulo associar com o dado destino IP? Para responder a essas perguntas, precisaremos estudar a interação entre um grupo de roteadores habilitados para MPLS.

No exemplo da Figura 5.36, os roteadores R1 a R4 são habilitados para MPLS. R5 e R6 são roteadores IP padrão. R1 anunciou a R2 e R3 que ele (R1) pode rotear para o destino A, e que um quadro recebido com rótulo MPLS 6 será repassado ao destino A. O roteador R3 anunciou ao roteador R4 que ele (R4) pode rotear para os destinos A e D e que os quadros que estão chegando e que portam os rótulos MPLS 10 e 12, respectivamente, serão comutados na direção desses destinos. O roteador R2 também anunciou ao roteador R4 que ele (R2) pode alcançar o destino A e que um quadro recebido portando o rótulo MPLS 8 será comutado na direção de A. Note que o roteador R4 agora está na interessante posição de ter *dois* caminhos MPLS para chegar até A: via a interface 0 com rótulo de saída MPLS 10 e via a interface 1 com um rótulo MPLS 8. O quadro geral apresentado na Figura 5.36 mostra que o modo como R4, R5, A e D estão conectados em conjunto via uma infraestrutura MPLS (roteadores habilitados a MPLS R1, R2, R3 e R4) é praticamente o mesmo modo pelo qual uma LAN comutada ou uma rede ATM podem conectar dispositivos IP entre si. E, do mesmo modo que uma LAN comutada ou uma rede ATM, os roteadores R1 a R4 habilitados para MPLS fazem isso sem jamais tocar o cabeçalho IP de um pacote.

Nessa discussão, não especificamos o protocolo específico utilizado para distribuir rótulos entre roteadores habilitados para MPLS, pois os detalhes dessa sinalização estariam muito além do escopo deste livro. Observamos, entretanto, que o grupo de trabalho da IETF para o MPLS especificou no RFC 3468 que uma extensão do protocolo RSVP (que estudaremos no Capítulo 7), conhecido como RSVP-TE [RFC 3209], será o foco de seus esforços para a sinalização MPLS. Assim, aconselhamos o leitor interessado a consultar o RFC 3209.

Até aqui, a ênfase de nossa discussão sobre o MPLS tem sido o fato de que esse protocolo executa comutação com base em rótulos, sem precisar considerar o endereço IP de um pacote. As verdadeiras vantagens do MPLS e a razão do atual interesse por ele, contudo, não estão nos aumentos substanciais nas velocidades de comutação, mas nas novas capacidades de gerenciamento de tráfego que o MPLS proporciona. Como observamos anteriormente,

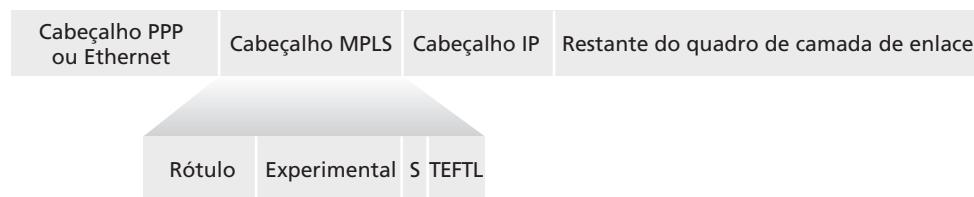


Figura 5.35 Cabeçalho MPLS: localizado entre os cabeçalhos de camada de enlace e camada de rede

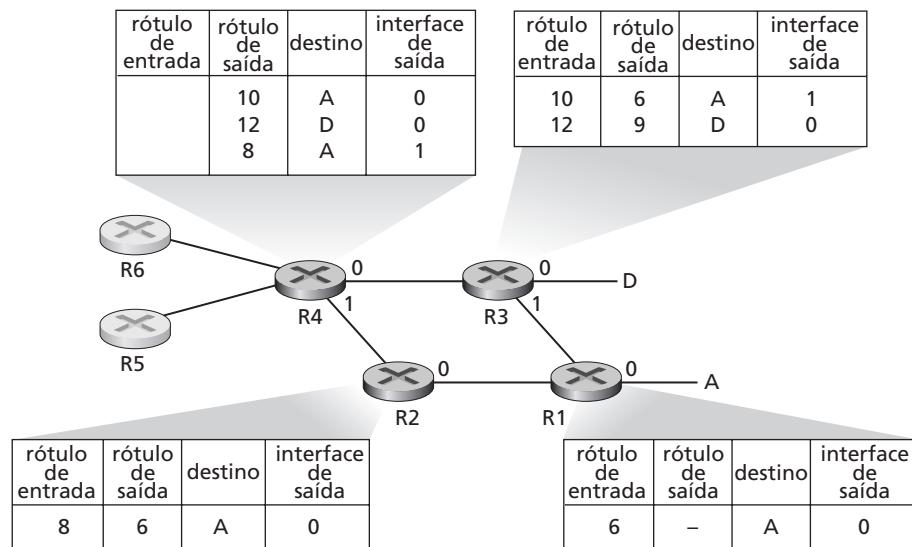


Figura 5.36 Repasse melhorado com MPLS

o R4 tem *dois* caminhos MPLS até A. Se o repasse fosse executado até a camada IP tendo como base o endereço IP, os protocolos de roteamento IP que estudamos no Capítulo 4 especificariam somente um único caminho de menor custo até A. Assim, o MPLS provê a capacidade de repassar pacotes por rotas que não seria possível utilizar com protocolos padronizados de roteamento IP. Essa é somente uma forma simples de **engenharia de tráfego** usando o MPLS [RFC 3346; RFC 3272; RFC 2702; Xiao, 2000], com a qual um operador de rede pode suplantar o roteamento IP normal e obrigar que uma parte do tráfego dirigido a um dado destino siga por um caminho, e que outra parcela do tráfego dirigido ao mesmo destino siga por um outro caminho (seja por política, por desempenho ou por alguma outra razão).

Também é possível utilizar MPLS para muitas outras finalidades. O protocolo pode ser usado para realizar restauração rápida de caminhos de repasse MPLS, por exemplo, mudar a rota do tráfego que passa sobre um caminho previamente calculado, em resposta à falha de enlace [Kar, 2000; Huang, 2002; RFC 3469]. O MPLS também pode ser utilizado para implementar a estrutura de serviço diferenciado ('diffserv'), que estudaremos no Capítulo 7. Finalmente, observamos que o MPLS pode ser, e tem sido, utilizado para implementar as denominadas **redes virtuais privadas** (*virtual private networks* — VPN). Ao implementar uma VPN para um cliente, um ISP utiliza uma rede habilitada para MPLS para conectar as várias redes do cliente. O MPLS também pode ser usado para isolar os recursos e o endereçamento utilizados pela VPN do cliente dos outros usuários que estão passando através da rede do ISP. Para mais detalhes, veja [DeClercq, 2002].

Nossa discussão sobre o MPLS foi necessariamente breve e aconselhamos o leitor a consultar as referências que mencionamos. Observamos que são tantas as utilizações possíveis do MPLS que ele parece estar se tornando rapidamente o canivetinho suíço da engenharia de tráfego da Internet!

5.9 Um dia na vida de uma solicitação de página Web

Agora que já cobrimos a camada de enlace neste capítulo, e da rede, de transporte e a camada de aplicação em capítulos anteriores, nossa jornada pela pilha de protocolos está completa! No começo do livro (Seção 1.1), nós escrevemos que “grande parte deste livro trata de protocolos de redes de computadores,” e nos primeiros cinco capítulos vimos que realmente isso é verdade. Antes de nos dirigirmos aos tópicos dos próximos capítulos da segunda parte deste livro, gostaríamos de finalizar nossa jornada pela pilha de protocolos considerando uma visão integrada e holística dos protocolos que vimos até agora. Uma forma de termos essa visão geral é identificarmos os

vários (vários!) protocolos envolvidos na satisfação de simples pedidos, como fazer o download de uma página web. A Figura 5.37 ilustra a imagem que queremos passar: um estudante, Bob, conecta seu laptop ao comutador Ethernet da sua escola e faz o download de uma página web (digamos que é a página principal de www.google.com). Como já sabemos, existe muito mais sob a superfície do que se imagina, para realizar esta solicitação aparentemente simples. O laboratório Wireshark examina cenários de comunicação mais detalhadamente no final deste capítulo, contendo vários pacotes envolvidos em situações parecidas.

Começando: DHCP, UDP, IP e Ethernet

Suponha que Bob liga seu laptop e o conecta através de um cabo Ethernet ao comutador Ethernet da escola, que por sua vez é conectado ao roteador da escola, como demonstrado na Figura 5.37. O roteador da escola é conectado a um ISP, que neste exemplo é comcast.net. Neste exemplo, a comcast.net fornece o serviço DNS para a escola; dessa forma, o servidor DNS se localiza em uma rede da comcast em vez de se localizar na rede da escola. Vamos supor que servidor DHCP está sendo executado dentro do roteador, como normalmente acontece.

Quando Bob conecta seu laptop à rede pela primeira vez, ele não consegue fazer nada (por exemplo, carregar uma página na Web) sem um endereço IP. Assim, a primeira ação relacionada à rede tomada pelo laptop é executar o protocolo DHCP para obter um endereço IP, bem como outras informações do servidor DHCP local:

1. O sistema operacional do laptop de Bob cria uma mensagem DHCP de solicitação (Seção 4.4.2) e coloca essa mensagem dentro do segmento UDP (Seção 3.3) com a porta de destino 67 (servidor DHCP) e porta de origem 68 (cliente DHCP). O segmento é então colocado dentro de um datagrama IP (Seção 4.4.1) com um endereço de destino IP transmitido (255.255.255.255) e um endereço de origem IP de 0.0.0.0, já que o laptop do Bob não tinha ainda um endereço IP.
2. O datagrama IP contendo uma mensagem de solicitação DHCP é colocado dentro de um quadro Ethernet (Seção 5.51). O quadro Ethernet tem endereços de destino MAC FF:FF:FF:FF:FF:FF para que o quadro seja transmitido a todos os dispositivos conectados ao comutador onde se espera que se inclua um servidor DHCP o quadro de origem MAC é do laptop do Bob, 00:16:D3:23:67:8A.
3. O quadro broadcast Ethernet contendo a solicitação DHCP é o primeiro quadro a ser enviado pelo laptop de Bob para o comutador Ethernet. O comutador transmite o quadro da entrada para todas as portas de saída, incluindo a porta conectada ao roteador.

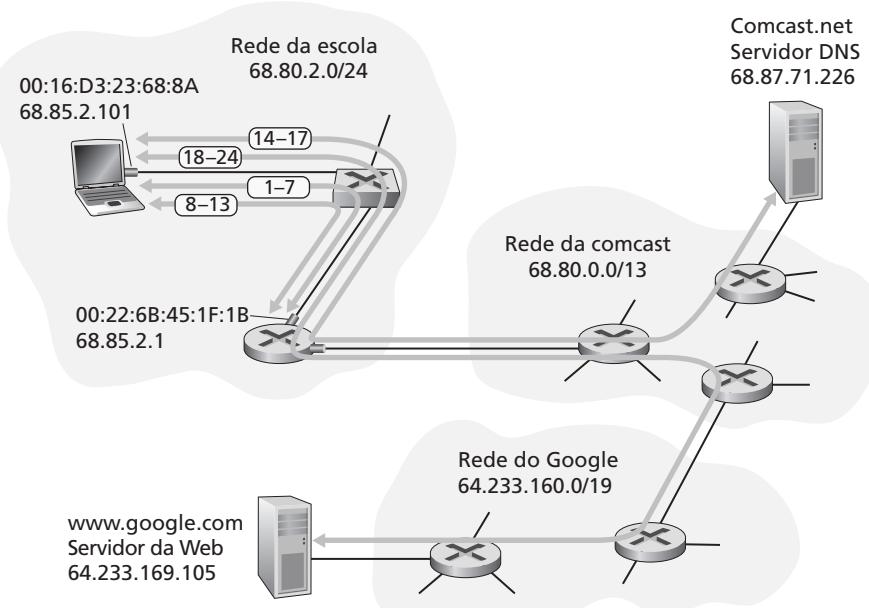


Figura 5.37

4. O roteador recebe o quadro Ethernet transmitido, que contém a solicitação DHCP na sua interface com endereço MAC 00:22:6B:45:1F:1B, e o datagrama IP é extraído do quadro Ethernet. O endereço de destino IP transmitido indica que este datagrama IP deveria ser processado por protocolos de camadas mais elevadas em seu nó, para que a carga útil (um segmento UDP) seja dessa forma demultiplexada (Seção 3.2) até o UDP, e a mensagem de solicitação extraída do segmento UDP. Agora o servidor DHCP tem a mensagem de solicitação DHCP.
5. Suponhamos que o servidor DHCP que esteja sendo executado dentro de um roteador possa alocar o endereço IP no bloco CIDR (Seção 4.4.2) 68.85.2.0/24. Neste exemplo, todos os endereços IP usados dentro da escola estão dentro do bloco de endereços da Comcast. Vamos supor que o servidor DHCP designe o endereço 68.85.2.101 ao laptop do Bob. O servidor DHCP cria uma mensagem DHCP ACK (Seção 4.4.2) contendo um endereço IP, assim como o endereço IP do servidor DNS (68.87.71.226), o endereço IP para o roteador de borda default (68.85.2.1), e o bloco de sub-rede (68.85.2.0/24) (equivalente à máscara de rede). A mensagem DHCP é colocada dentro de um segmento UDP, o qual é colocado dentro de um datagrama IP, o qual é colocado dentro de um quadro Ethernet. O quadro Ethernet tem o endereço MAC de origem da interface do roteador na rede doméstica (00.22.6B.45.1F.1B) e um endereço MAC de destino do laptop do Bob (00.16.D3.23.68.8A).
6. Uma vez que o comutador realiza a aprendizagem automática e que recebe previamente um quadro do laptop de Bob (que contém a solicitação DHCP), o comutador sabe como encaminhar um quadro endereçado a 00.16.D3.23.68.8A, somente para a porta de saída que leva ao laptop do Bob.
7. O laptop do Bob recebe o quadro Ethernet que contém o DHCP ACK, extrai o datagrama IP do quadro Ethernet, extrai o segmento UDP do datagrama IP, e extrai a mensagem DHCP ACK do segmento UDP. Então, o cliente DHCP do Bob grava seu endereço IP e o endereço IP do seu servidor DNS. Ele também instala o endereço da saída padrão (default gateway) em sua tabela IP de repasse (Seção 4.1). O laptop do Bob irá enviar todos os datagramas com endereços de destino fora de sua sub-rede 68.85.2.0/24 à saída padrão. Nesse momento, o laptop do Bob inicializou os seus componentes de rede e está pronto para começar a processar a busca da página web. (Observe que somente as duas últimas etapas DHCP das quatro apresentadas no Capítulo 4 são realmente necessárias).

Ainda começando: DNS, ARP

Quando Bob digita a URL para www.google.com em seu navegador web, ele inicia uma longa cadeia de eventos que finalmente resultarão na exibição da página principal do Google no seu navegador web. O navegador web do Bob inicia o processo ao criar um socket TCP (Seção 2.7) que será usado para enviar uma requisição HTTP (Seção 2.2) para www.google.com. Para criar o socket, o laptop do Bob precisará saber o endereço IP de www.google.com. Aprendemos na Seção 2.5, que o protocolo DNS é usado para fornecer serviços de traduções de nomes para endereço IP.

8. O sistema operacional do laptop do Bob cria então uma mensagem de consulta DNS (Seção 2.5.3), colocando a cadeia de caracteres www.google.com no campo de pergunta da mensagem DNS. Essa mensagem DNS é então colocada dentro de um segmento UDP, com a porta de destino 53 (servidor DNS). O segmento UDP é então colocado dentro de um datagrama IP com um endereço de destino IP 68.87.71.226 (o endereço do servidor DNS retomada pelo DHCP ACK na etapa 5) e um endereço de origem IP 68.85.2.101.
9. O laptop de Bob coloca então um datagrama contendo a mensagem de consulta DNS em um quadro Ethernet. Este quadro será enviado (endereçado, na camada de enlace) ao roteador de borda da rede da escola de Bob. No entanto, apesar de o laptop de Bob conhecer o endereço IP do roteador de borda da rede da escola (68.85.2.1), via a mensagem DHCP ACK na etapa 5 anterior, ele não sabe o endereço MAC do roteador de borda. Para que o laptop do Bob obtenha o endereço MAC do roteador de borda, ele precisará usar o protocolo ARP (Seção 5.4.2).
10. O laptop de Bob cria uma mensagem de consulta ARP direcionada ao endereço IP 68.85.2.1 (a porta padrão), coloca a mensagem ARP dentro do quadro Ethernet para ser transmitido em broadcast ao endereço de destino (FF:FF:FF:FF:FF:FF) e envia o quadro Ethernet ao comutador, que entrega o quadro a todos dispositivos, incluindo o roteador de borda.

11. O roteador de borda recebe um quadro contendo a mensagem de consulta ARP na interface da rede da escola, e encontra que o endereço alvo IP 68.85.2.1 na mensagem ARP é compatível com o endereço IP de sua interface. Então, o roteador de borda prepara uma resposta ARP, indicando que o seu endereço MAC 00:22:6B:45:1F:1B corresponde ao endereço IP 68.85.2.1. Ele coloca a mensagem de resposta ARP em um quadro Ethernet, com o endereço de destino 00:16:D3:23:68:8A (laptop do Bob) e envia o quadro ao comutador, que entrega o quadro ao laptop de Bob.
12. O laptop de Bob recebe o quadro que contém a mensagem de resposta ARP e extrai o endereço MAC do roteador de borda (00:22:6B:45:1F:1B) da mensagem de resposta ARP.
13. O laptop de Bob pode agora (finalmente!) endereçar o quadro Ethernet contendo a mensagem de consulta DNS ao endereço MAC do roteador de borda. Observe que o datagrama nesse quadro tem o endereço de destino IP 68.87.71.226 (servidor DNS), enquanto o quadro tem o endereço de destino 00:22:6B:45:1F:1B (roteador de borda). O laptop de Bob envia esse quadro ao comutador, que entrega o quadro ao roteador de borda.

Ainda começando: roteamento de intradomínio ao servidor DNS

14. O roteador de borda recebe o quadro e extrai o datagrama IP que contém a consulta DNS. O roteador procura o endereço de destino desse datagrama (68.87.71.226) e determina de sua tabela de encaminhamento que o datagrama deve ser enviado ao roteador da extremidade esquerda na rede Comcast, como na Figura 5.37. O datagrama IP é colocado dentro de um quadro de uma camada de enlace apropriado de acordo com o enlace conectando o roteador da escola ao roteador Comcast da extremidade esquerda e o quadro é enviado através desse enlace.
15. O roteador da extremidade esquerda na rede Comcast recebe o quadro, extrai o datagrama IP, examina o endereço de destino do datagrama (68.87.71.226) e determina a interface de saída pela qual irá enviar o datagrama ao servidor DNS de sua tabela de encaminhamento, que foi preenchida pelo protocolo Intradomínio da Comcast (como RIP, OSPF ou IS-IS, Seção 4.6), assim como o protocolo Intradomínio da Internet, BGP.
16. Finalmente, o datagrama IP contendo a consulta DNS chega ao servidor DNS. O servidor DNS extrai a mensagem de consulta DNS, procura o nome em www.google.com na sua base de dados DNS (Seção 2.5), e encontra o registro da fonte DNS que contém o endereço IP (64.233.169.105) para www.google.com (supondo-se que está em cache no servidor DNS). Lembre-se que este dado em cache foi originado no servidor DNS com autoridade (Seção 2.5.2) para googlecom. O servidor DNS forma uma mensagem DNS de resposta contendo o mapeamento do endereço IP para o nome fonte e o coloca a mensagem DNS de resposta em um segmento UDP, e o segmento dentro do datagrama IP endereçado ao laptop do Bob (68.85.2.101). Esse datagrama será encaminhado de volta à rede Comcast ao roteador da escola e de lá ao laptop do Bob, via o comutador Ethernet.
17. O laptop do Bob extrai o endereço IP do servidor www.google.com da mensagem DNS. Finalmente, depois de muito trabalho, o laptop do Bob está pronto para contatar o servidor www.google.com!

Interação cliente-servidor Web: TCP e HTTP

18. Agora que o laptop de Bob tem o endereço IP de www.google.com, ele está pronto para criar um socket TCP (Seção 2.7), que será usada para enviar uma mensagem HTTP GET (Seção 2.2.3) para www.google.com. Quando Bob cria um socket TCP, o TCP do laptop do Bob precisa primeiro executar uma apresentação de três vias (Seção 3.5.6) com o TCP em www.google.com. Então, o laptop de Bob primeiro cria um segmento TCP SYN com a porta de destino 80 (para HTTP), coloca o segmento TCP dentro de um datagrama IP, com o endereço de destino IP 64.233.169.105 (www.google.com), coloca o datagrama dentro de um quadro com o endereço de destino 00:22:6B:1F:1B (o roteador de borda) e envia o quadro ao comutador.
19. Os roteadores da rede da escola, da rede Comcast e da rede do Google encaminham o datagrama contendo o TCP SYN até www.google.com, usando a tabela de encaminhamento em cada roteador, como nas etapas 14-16 acima. Os itens da tabela de encaminhamento governando o envio de pacotes interdomínio entre as redes da Comcast e do Google, são determinadas pelo protocolo BGP (Seção 4.6.3).

20. Finalmente, o datagrama contendo o TCP SYN chega em www.google.com. A mensagem TCP SYN é extraída do datagrama e demultiplexada ao socket associado à porta 80. Um socket de conexão (Seção 2.7) é criado para a conexão TCP entre o servidor HTTP do Google e o laptop de Bob. Um segmento TCP SYNACK (Seção 3.5.6) é gerado, colocado dentro de uma datagrama endereçado ao laptop de Bob, e finalmente colocado dentro de um quadro de camada de enlace apropriado ao enlace conectando www.google.com ao seu roteador de primeiro salto.
21. O datagrama que contém o segmento TCP SYNACK é encaminhado através das redes do Google, Comcast e da escola, finalmente chegando no cartão Ethernet no computador de Bob. O datagrama é demultiplexado dentro do sistema operacional e entregue ao socket TCP criado na etapa 18, que entra em estado de conexão.
22. Agora, com o socket dentro do laptop de Bob (finalmente!), pronto pra enviar bytes a www.google.com, o navegador de Bob cria uma mensagem HTTP GET (Seção 2.2.3) contendo a URL a ser procurada. A mensagem HTTP GET é enviada ao socket, com a mensagem GET se tornando a carga útil do segmento TCP. O segmento TCP é colocado em um datagrama e enviado e entregue em www.google.com, como nas etapas 18-20 acima.
23. O servidor HTTP www.google.com lê a mensagem HTTP GET do socket TCP, cria uma resposta HTTP (Seção 2.2), coloca o conteúdo da página web requisitada no corpo da mensagem de resposta HTTP, e envia a mensagem pelo socket TCP.
24. O datagrama contendo a mensagem de resposta HTTP é encaminhada através das redes do Google, da Comcast e da escola e chega ao laptop de Bob. O programa do navegador de Bob lê a resposta HTTP do socket, extrai o html da página web do corpo da resposta HTTP, e finalmente (finalmente!) mostra a página web!

Nossa representação acima vimos muito do fundamento das redes de comunicação! Se você entendeu a maior parte da representação acima, então você também viu muito do fundamento desde que leu a Seção 1.1, onde escrevemos ““grande parte deste livro trata de protocolos de redes de computadores,” e você pode ter se perguntado o que na verdade era um protocolo! Por mais detalhado que o exemplo acima possa parecer, nós omitidos um número de protocolos possíveis (por exemplo, NAT executado no roteador de borda da escola, acesso sem fio à rede da escola, protocolos de segurança para acessar a rede da escola, ou segmentos ou datagramas codificados), e considerações (cache da web, hierarquia DNS) que poderíamos encontrar na internet pública. Estudaremos a maioria desses tópicos na segunda parte deste livro.

Finalmente, observamos que nosso exemplo acima era integrado e holístico, mas também muito resumido de muitos protocolos que estudamos na primeira parte deste livro. Este exemplo é mais focado nos aspectos de “como”, e não no “porquê”. Para uma visão mais ampla e reflectiva dos protocolos de rede em geral, veja [Clark, 1988, RFC 5218].

5.10 Resumo

Neste capítulo, examinamos a camada de enlace — seus serviços, os princípios subjacentes à sua operação e vários protocolos específicos importantes que usam esses princípios na implementação dos serviços da camada de enlace.

Vimos que o serviço básico da camada de enlace é mover um datagrama de camada de rede de um nó (roteador ou hospedeiro) até um nó adjacente. Vimos também que todos os protocolos de camada de enlace operam encapsulando um datagrama de camada de rede dentro de um quadro de camada de enlace antes de transmitir o quadro pelo enlace que vai até o nó adjacente. Além dessa função comum de enquadramento, contudo, aprendemos que diferentes protocolos de camada de enlace oferecem serviços muito diferentes de acesso ao enlace, de entrega (confiabilidade, detecção/correção de erros), de controle de fluxo e de transmissão (por exemplo, serviços *full-duplex* versus *half-duplex*). Essas diferenças são devidas, em parte, à vasta variedade de tipos de enlaces sobre os quais os protocolos de enlace devem operar. Um enlace ponto a ponto simples tem um único remetente e um único receptor comunicando-se por um único ‘fio’. Um enlace de acesso múltiplo é compartilhado por muitos remetentes e receptores; consequentemente, o protocolo de camada de enlace para um canal de acesso múltiplo

tem um protocolo (seu protocolo de acesso múltiplo) para coordenar o acesso ao enlace. Nos casos de ATM e MPLS, o ‘enlace’ que conecta dois nós adjacentes (por exemplo, dois roteadores IP que são adjacentes em sentido IP — ou seja, são roteadores IP do próximo salto na direção do destino) pode, na realidade, constituir uma rede em si e por si próprio. Em certo sentido, a ideia de uma rede ser considerada um ‘enlace’ não deveria parecer estranha. Um enlace telefônico que conecta um modem/computador residencial a um modem/roteador remoto, por exemplo, na verdade é um caminho que atravessa uma sofisticada e complexa rede de telefonia.

Dentre os princípios subjacentes à comunicação por camada de enlace, examinamos técnicas de detecção e correção de erros, protocolos de acesso múltiplo, endereçamento de camada de enlace e a construção de redes locais ampliadas por hubs e comutadores. No caso da detecção/correção de erros, examinamos como é possível adicionar bits ao cabeçalho de um quadro para detectar e algumas vezes corrigir erros de mudança de bits que podem ocorrer quando o quadro é transmitido pelo enlace. Analisamos esquemas simples de paridade e de soma de verificação, bem como o esquema mais robusto de verificação da redundância cílica. Passamos, então, para o tópico dos protocolos de acesso múltiplo. Identificamos e estudamos três abordagens amplas para coordenar o acesso a um canal *broadcast*: abordagens de divisão de canal (TDM, FDM, CDMA), abordagens de acesso aleatório (os protocolos ALOHA e os protocolos CSMA) e abordagens de revezamento (polling e passagem de permissão). Vimos que uma consequência do compartilhamento de um canal *broadcast* por múltiplos nós é a necessidade de prover endereços aos nós no nível da camada de enlace. Aprendemos que endereços físicos são bastante diferentes de endereços de camada de rede e que, no caso da Internet, um protocolo especial (o ARP — protocolo de resolução de endereço) é usado para fazer o mapeamento entre esses dois modos de endereçamento. Em seguida, examinamos como os nós que compartilham um canal *broadcast* formam uma LAN e como várias LANs podem ser conectadas para formar uma LAN de maior porte — tudo isso sem a intervenção de roteamento de camada de rede para a interconexão desses nós locais.

Também estudamos detalhadamente vários protocolos específicos de camada de enlace — Ethernet e PPP. Encerramos nosso estudo da camada de enlace focalizando como redes MPLS fornecem serviços de camada de enlace quando interconectadas com roteadores IP. Concluímos este capítulo (e, sem dúvida, os cinco primeiros capítulos) identificando os muitos protocolos que são necessários para buscar uma simples página web. Com isso, concluímos nossa jornada de cima para baixo da pilha de protocolos. É claro que a camada física fica abaixo da camada de enlace, mas provavelmente será melhor deixar os detalhes da camada física para outro curso. Contudo, discutimos brevemente vários aspectos da camada física — neste capítulo, por exemplo, discutimos a codificação Manchester na Seção 5.5, e no Capítulo 1, meios físicos na Seção 1.2. Consideraremos novamente a camada física quando estudarmos as características de enlaces sem fio no próximo capítulo.

Embora nossa jornada de cima para baixo da pilha de protocolos esteja encerrada, nosso estudo sobre rede de computadores ainda não chegou ao fim. Nos quatro capítulos seguintes, examinaremos rede sem fio, rede multimídia, segurança da rede e gerenciamento de rede. Esses quatro tópicos não se ajustam convenientemente a uma única camada; na verdade, cada um deles atravessa muitas camadas. Assim, entender esses tópicos (às vezes tachados de ‘tópicos avançados’ em alguns textos sobre redes) requer uma boa base sobre todas as camadas da pilha de protocolos — uma base que se completou com nosso estudo sobre a camada de enlace de dados!



Exercícios de fixação

Capítulo 5 Questões de revisão

Seções 5.1 a 5.2

1. Considere a analogia de transporte na Seção 5.1.1. Se o passageiro é análogo ao datagrama, o que é análogo ao quadro da camada de enlace?

2. Se todos os enlaces da Internet fornecessem serviço confiável de entrega, o serviço confiável de entrega TCP seria redundante? Justifique sua resposta.
3. Quais alguns possíveis serviços que um protocolo de camada de enlace pode oferecer à camada de rede.



Quais desses serviços de camada de enlace têm serviços correspondentes no IP? E no TCP?

Seção 5.3

4. Suponha que dois nós comecem a transmitir ao mesmo tempo um pacote de comprimento L por um canal broadcast de velocidade R . Denote o atraso de propagação entre os dois nós como t_{prop} . Haverá uma colisão se $t_{prop} < L/R$? Por quê?
5. Na Seção 5.3, relacionamos quatro características desejáveis de um canal broadcast. O slotted ALOHA tem quais dessas características? E o protocolo de passagem de permissão, tem quais dessas características?
6. Descreva os protocolos de polling e de passagem de permissão usando a analogia com as interações ocorridas em um coquetel.
7. Por que o protocolo de passagem de permissão seria ineficiente se uma LAN tivesse um perímetro muito grande?

Seção 5.4

8. Que tamanho tem o espaço de endereço MAC? E o espaço de endereço IPv4? E o espaço de endereço IPv6?
9. Suponha que cada um dos nós A, B e C esteja ligado à mesma LAN broadcast (por meio de seus adaptadores). Se A enviar milhares de datagramas IP a B com quadro de encapsulamento endereçado ao endereço MAC de B, o adaptador de C processará esses quadros? Se processar, ele passará os datagramas IP desses quadros para C (isto é, para o nó pai do adaptador)? O que mudaria em suas respostas se A enviasse quadros com o endereço MAC de broadcast?
10. Por que uma pesquisa ARP é enviada dentro de um quadro broadcast? Por que uma resposta ARP

é enviada dentro de um quadro com um endereço MAC de destino específico?

11. Na rede da Figura 5.19, o roteador tem dois módulos ARP, cada um com sua própria tabela ARP. É possível que o mesmo endereço MAC apareça em ambas as tabelas?

Seção 5.5

12. Compare as estruturas de quadro das redes Ethernet 10BaseT, 100BaseT e Gigabit Ethernet. Quais as diferenças entre elas?
13. Suponha que um adaptador de 10 Mbps envie para dentro de um canal uma cadeia infinita de 1s usando a codificação Manchester. Quantas transições por segundo terá o sinal que emerge do adaptador?
14. Em CSMA/CD, após a quinta colisão, qual é a probabilidade de que um nó escolha $K = 4$? O resultado $K = 4$ corresponde a um atraso de quantos segundos em uma rede Ethernet de 10 Mbps?

Seção 5.6

15. Considere a Figura 5.26. Quantas sub-redes existem, em relação ao acesso da Seção 4.4?

Seção 5.7

16. Qual o número máximo de VLANs que podem ser configuradas em um comutador que suporta o protocolo 802.Q1? Por quê?
17. Imagine que N comutadores que suportam K grupos de VLAN serão conectados via um protocolo de entroncamento. Quantas portas serão necessárias para conectar os comutadores? Justifique sua resposta.

Problemas

1. Suponha que o conteúdo de informação de um pacote seja o padrão de bits 1110 1011 1001 1101 e que um esquema de paridade par esteja sendo usado. Qual seria o valor do campo de soma de verificação para o caso de um esquema de paridade bidimensional? Sua resposta deve ser tal que seja usado um campo de soma de verificação de comprimento mínimo.
2. Dê um exemplo (que não seja o da Figura 5.6) mostrando que verificações de paridade bidimensional podem corrigir e detectar um erro de bit único. Dê um outro exemplo mostrando um erro de bit duplo que pode ser detectado, mas não corrigido.
3. Suponha que a porção de informação de um pacote (D da Figura 5.4) contenha 10 bytes consistindo na representação binária ASCII sem sinal de uma cadeia de caracteres da “Camada de Enlace”. Calcule a soma de verificação da Internet para esses dados.
4. Considere o problema anterior, mas em vez de conter os números binários do 0 ao 9, suponha que esses 10 bytes contenham:
 - a. A representação binária dos números de 1 a 10.
 - b. A representação ASCII das letras A a J (letras maiúsculas).

- c. A representação ASCII das letras a a j (letras minúsculas).
- Calcule a soma de verificação da Internet para esses dados.
5. Considere o gerador de 7 bits $G=10011$ e suponha que D tenha o valor de 1010101010. Qual é o valor de R ?
6. Considere o problema acima, mas suponha que D tenha o valor de:
- 1001000101
 - 1010001111
 - 0101010101
7. Neste problema, exploramos algumas propriedades de CRC. Para o gerador G (=1001) dado na Seção 5.2.3, responda às seguintes questões:
- Por que ele pode detectar qualquer erro de bit único no dado D ?
 - Pode o G acima detectar qualquer número ímpar de erros de bit? Por quê?
8. Na Seção 5.3, fornecemos um esboço da derivação da eficiência do slotted ALOHA. Neste problema, concluirímos a derivação.
- Lembre-se de que, quando há N nós ativos, a eficiência do slotted ALOHA é $Np(1 - p)^{N-1}$. Ache o valor de p que maximize essa expressão.
 - Usando o valor de p encontrado em (a), ache a eficiência do slotted Aloha fazendo com que N tenda ao infinito. Dica: $(1 - 1/N)^N$ tende a $1/e$ quando N tende ao infinito.
9. Mostre que a eficiência máxima do ALOHA puro é $1/(2e)$. Obs.: Este problema é fácil se você tiver concluído o anterior.
10. Considere dois nós, A e B, que usem um protocolo ALOHA slotted para competir pelo canal. Suponha que o nó A tenha mais dados para transmitir do que o nó B, e a probabilidade de retransmissão do nó A p_A é maior do que a probabilidade de retransmissão do nó B p_B .
- Determine a fórmula para a vazão média do nó A. Qual é a eficiência total do protocolo com esses dois nós.
 - Se $p_A = 2p_B$, a vazão média do nó A é duas vezes maior do que o do nó B? Por quê, ou por quê não? Se não, como escolher p_A e p_B para que isso aconteça?
 - No geral, suponha que existem N nós, e entre eles o nó A tem a probabilidade de retransmissão de $2p$ e todos os outros nós tem a probabilidade de retransmissão de p . Determine as expressões para computar a vazão média do nó A e de qualquer outro nó.
11. Suponha que quatro nós ativos — nós A, B, C e D — estão competindo pelo acesso a um canal usando o slotted ALOHA. Imagine que cada nó tem um número infinito de pacotes para enviar. Cada nó tenta transmitir em cada slot com probabilidade p . O primeiro slot é numerado como 1, o segundo slot como 2, e assim por diante.
- Qual a probabilidade de o nó A tenha sucesso pela primeira vez no slot 5?
 - Qual a probabilidade que algum nó (A, B, C ou D) tenha sucesso no slot 4?
 - Qual a probabilidade que o primeiro sucesso ocorra no slot 3?
 - Qual a eficiência nesse sistema de quatro nós?
12. Plote em gráfico a eficiência do slotted ALOHA e do ALOHA puro como uma função de p , para N :
- $N = 15$
 - $N = 20$
 - $N = 30$
13. Considere um canal *broadcast* com N nós e uma taxa de transmissão de R bps. Suponha que o canal *broadcast* use protocolo de polling (com um nó de polling adicional) para acesso múltiplo. Suponha que o intervalo de tempo entre o momento em que o nó conclui a transmissão e o momento em que o nó subsequente é autorizado a transmitir (isto é, o atraso de polling) seja d_{poll} . Suponha ainda que, dentro de uma rodada de polling, um dado nó seja autorizado a transmitir, no máximo, Q bits. Qual é a vazão máxima do canal *broadcast*?
14. Considere três LANs interconectadas por dois roteadores, como mostrado na Figura 5.38
- Atribua endereços a todas as interfaces. Para a Sub-rede 1, use endereços do tipo 192.168.1.xxx; para a Sub-rede 2, use endereços do tipo 192.168.2.xxx, e para a Sub-rede 3 use endereços do tipo 192.168.3.xxx.
 - Atribua endereços de MAC a todos os adaptadores.
 - Considere o envio de um datagrama IP do hospedeiro A ao hospedeiro F. Suponha que todas as tabelas ARP estejam atualizadas. Enumere todas as etapas como foi feito no exemplo de um único roteador na Seção 5.4.2.
 - Repita (d), admitindo agora que a tabela ARP do hospedeiro remetente esteja vazia (e que as outras tabelas estejam atualizadas).
15. Considere a Figura 5.38. Agora substituimos o roteador entre as sub-redes 1 e 2 pelo comutador S1, e etiquetamos o roteador entre as sub-redes 2 e 3 como R1.

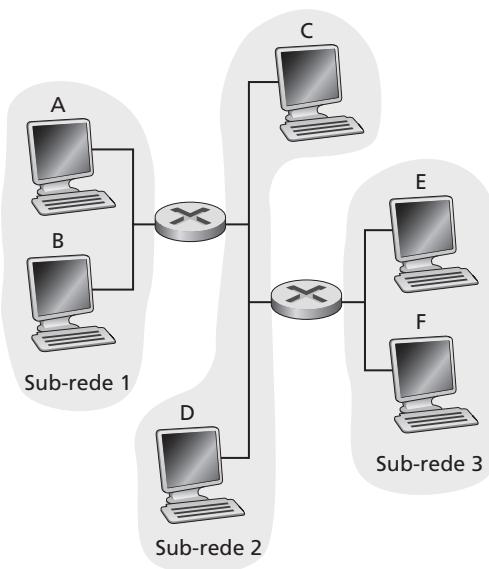


Figura 5.38 3 sub-redes, interconectadores por roteadores

- Considere o envio de um datagrama IP do Hospedeiro E ao Hospedeiro F. O Hospedeiro E pedirá ajuda ao roteador R1 para enviar o datagrama? Por quê? No quadro Ethernet que contém o datagrama IP, qual são os endereços de origem e destino IP e MAC?
- Suponha que E gostaria de enviar um datagrama IP a B, e que o cache ARP de E não tenha o endereço MAC de B. B preparará uma consulta ARP para descobrir o endereço MAC de B? Por quê? No quadro Ethernet (que contém o datagrama IP destinado a B) que é entregue ao roteador R1, quais são os endereços de origem e destino IP e MAC?
- Suponha que o Hospedeiro A gostaria de enviar um datagrama IP ao Hospedeiro B, e nem o cache ARP de A contém o endereço MAC de B, nem o ARP cache de B contém o endereço MAC de A. Suponha também que a tabela de encaminhamento do comutador S1 contenha entradas somente para o Hospedeiro B e para o roteador R1. Dessa forma, A transmitirá uma mensagem de requisição ARP. Que ações o comutador S1 tomará quando receber a mensagem de requisição ARP? O roteador R1 também receberá a mensagem de requisição ARP? Se sim, R1 encaminhará a mensagem para a Sub-rede 3? Assim que o Hospedeiro B receber esse mensagem de requisição ARP, ele enviará a mensagem de volta ao Hospedeiro A. Mas enviará uma mensagem ARP de consulta para o endereço MAC de A? Por quê? O que o comutador S1 fará quando receber mensagem de resposta ARP do Hospedeiro B?

- Considerando o problema anterior, mas suponha que o roteador entre as sub-redes 2 e 3 é substituído por um comutador. Responda às questões de (a) a (c) do exercício anterior nesse novo contexto.
 - Lembre-se de que, com o protocolo CSMA/CD, o adaptador espera $K \cdot 512$ tempos de bits após uma colisão, onde K é escolhido aleatoriamente. Para $K = 100$, quanto tempo o adaptador espera até voltar à etapa 2 para uma Ethernet de 10 Mbps? E para uma Ethernet de 100 Mbps?
 - Suponha que os nós A e B estejam no mesmo segmento Ethernet de 10 Mbps e que o atraso de propagação entre os dois nós seja de 225 tempos de bit. Suponha que o nó A comece a transmitir um quadro e que, antes de terminar, o nó B comece a transmitir um quadro. O nó A pode terminar de transmitir antes de detectar que B transmitiu? Por quê? Se a resposta for sim, então A acredita, incorretamente, que seu quadro foi transmitido com sucesso, sem nenhuma colisão.
- Dica: suponha que no tempo $t = 0$ tempo de bit, A comece a transmitir um quadro. No pior dos casos, A transmite um quadro de tamanho mínimo de $512 + 64$ tempos de bit. Portanto, A terminaria de transmitir o quadro em $t = 512 + 64$ tempos de bit. Então, a resposta será não, se o sinal de B chegar a A antes do tempo de bit $t = 512 + 64$ bits. No pior dos casos, quando o sinal de B chega a A?
- Explique por que um quadro de tamanho mínimo é exigido pela Ethernet. Por exemplo 10Base Ethernet impõe um quadro de tamanho mínimo 64 bytes. (Se você resolveu o problema anterior, você deve ter se dado conta da razão). Agora suponha que a distância entre as duas extremidades de uma LAN Ethernet seja d . Você consegue produzir a fórmula para achar o quadro de tamanho mínimo necessário ao pacote Ethernet? Baseado no seu resultado, qual é o tamanho de pacote mínimo necessário para que uma Ethernet tenha a amplitude de 2 quilômetros?
 - Suponha que você gostaria de aumentar a velocidade do enlace do seu cabo Ethernet. Como esse aprimoramento afetaria os tamanhos mínimos necessários ao pacote? Se ao aprimorar seu cabo para uma velocidade mais alta e perceber que não pode mudar o tamanho do pacote, o que mais você pode fazer para manter a operação correta?
 - Suponha que os nós A e B estejam no mesmo segmento de uma Ethernet de 10 Mbps e que o atraso de propagação entre os dois nós seja de 245 tempos de bit. Suponha também que A e B enviem quadros ao mesmo tempo, que os quadros colidam e que, então, A e B escolham valores diferentes de K no algoritmo CSMA/CD. Admitindo que nenhum outro nó esteja

ativo, as retransmissões de A e B podem colidir? Para nossa finalidade, é suficiente resolver o seguinte exemplo. Suponha que A e B comecem a transmitir em $t = 0$ tempo de bit. Ambos detectam colisões em $t = 245$ tempos de bit. Eles terminam de transmitir um sinal de reforço de colisão em $t = 245 + 48 = 293$ tempos de bit. Suponha que $K_A = 0$ e $K_B = 1$. Em que tempo B programa sua retransmissão? Em que tempo A começa a transmissão? (Nota: os nós devem esperar por um canal ocioso após retornar à etapa 2 — veja o protocolo.) Em que tempo o sinal de A chega a B? B se abstém de transmitir em seu tempo programado?

22. Considere uma Ethernet 100BASE-T de 100 Mbps. Para ter uma eficiência de 0,50, qual deve ser a distância máxima entre um nó e o hub? Admita um comprimento de quadro de 1.000 bytes e que não há repetidores. Essa distância máxima também garante que um nó transmissor A poderá detectar se outro nó transmitiu enquanto A estava transmitindo? Justifique sua resposta. Como se compara sua distância máxima com a estabelecida pelo próprio padrão de 100 Mbps? Suponha que a velocidade de propagação do sinal na 100BASE-T Ethernet é $1,8 \cdot 10^8$ m/seg.
23. Suponha que quatro nós, A, B, C e D, estão conectados a um hub via cabos Ethernet de 10 Mbps. As distâncias entre o hub e estes quatro nós são 300m, 400m, 500m e 700m, respectivamente. Lembre-se de que o protocolo CSMA/CD é usado para essa Ethernet. Suponha que a velocidade de propagação do sinal é 2×10^8 m/seg.
 - a. Qual a extensão de quadro mínima necessária? E a extensão máxima?
 - b. Se todos os quadros têm 1.500 bits de extensão, encontre a eficiência dessa Ethernet.
24. Neste problema, você derivará a eficiência de um protocolo de acesso múltiplo semelhante ao CSMA/CD. Nesse protocolo, o tempo é segmentado e todos os adaptadores estão sincronizados com os intervalos. Diferentemente do slotted ALOHA, contudo, o comprimento de um intervalo (em segundos) é muito menor do que um tempo de quadro (o tempo para transmitir um quadro). Seja S o comprimento de um intervalo. Suponha que todos os quadros tenham comprimento constante $L = kRS$, onde R é a taxa de transmissão do canal e k é um número inteiro grande. Suponha que haja N nós, cada um com um número infinito de quadros para enviar. Admitimos também que $d_{prop} < S$, de modo que todos os nós podem detectar uma colisão antes do final de um intervalo de tempo. O protocolo é como segue:

Se, para um dado intervalo, nenhum nó estiver de posse do canal, todos os nós disputam o canal; em particular, cada nó transmite no intervalo com probabilidade p . Se exatamente um nó transmitir no intervalo, esse nó tomará posse do canal para os $k - 1$ intervalos subsequentes e transmitirá seu quadro inteiro.

Se algum nó estiver de posse do canal, todos os outros nós evitarão transmitir até que o nó que está de posse do canal tenha terminado de transmitir seu quadro. Assim que esse nó tiver transmitido seu quadro, todos os nós disputarão o canal.

Note que o canal se alterna entre dois estados: o estado produtivo, que dura exatamente k intervalos, e o estado não produtivo, que dura um número aleatório de intervalos. A eficiência do canal é, claramente, a razão $k/(k + x)$, onde x é o número esperado de intervalos consecutivos não produtivos.

- a. Para N e p fixos, determine a eficiência desse protocolo.
 - b. Para N fixo, determine p que maximiza a eficiência.
 - c. Usando o p (que é uma função de N) encontrado em (b), determine a eficiência quando N tende ao infinito.
 - d. Mostre que essa eficiência se aproxima de 1 quando o comprimento do quadro é grande.
25. Suponha que dois nós, A e B, estejam ligados às extremidades opostas de um cabo de 900 metros e que cada um tenha um quadro de 1.500 bits (incluindo todos os cabeçalhos e preâmbulos) para transmitir um ao outro. Ambos os nós tentam transmitir no tempo $t = 0$. Suponha que haja quatro repetidores entre A e B, cada um inserindo um atraso de 20 bits. Admita que a taxa de transmissão é 100 Mbps e que é usado um CSMA/CD com intervalos de backoff de múltiplos de 512 bits. Após a primeira colisão, A sorteia $K = 0$ e B sorteia $K = 1$ no protocolo de backoff exponencial. Ignore o sinal de reforço e o tempo de atraso de 96 bits.
- a. Qual é o atraso de propagação em um sentido (incluindo os atrasos de repetidores) entre A e B em segundos? Suponha que a velocidade de propagação do sinal seja $2 \cdot 10^8$ m/s.
 - b. Em que tempo (em segundos) o pacote de A é completamente entregue em B?
 - c. Agora suponha que somente A tenha um pacote para enviar e que os repetidores sejam substituídos por comutadores. Suponha que cada comutador tenha um atraso de processamento de 20

- bits além do atraso de armazenagem e repasse. Em que tempo, em segundos, o pacote de A é entregue em B?
26. No padrão Ethernet, um remetente espera 96 tempos de bit entre enviar quadros consecutivos. Este período de pausa é referido como gap de interquadro, e é usado para permitir que o dispositivo de recebimento complete o processo de um quadro recebido e se prepare para o recebimento de um próximo quadro. Desde que o padrão Ethernet foi especificado, muitas melhorias tecnológicas aconteceram, incluindo a velocidade dos processadores, memória e as taxas da Ethernet. Se o padrão pudesse ser reescrito, como essas melhorias influenciariam o gap de interquadro?
27. Considere a Figura 5.38 no problema P14. Determine os endereços MAC e IP para as interfaces do Hospedeiro A, ambos os roteadores e do Hospedeiro F. Determine os endereços MAC de origem e destino no quadro que encapsula desse datagrama IP, já que o quadro é transmitido (i) de A ao roteador esquerdo, (ii) do roteador esquerdo ao roteador direito, (iii) do roteador direito a F. Determine também os endereços IP de origem e destino no datagrama IP encapsulado no quadro em cada um desses pontos no tempo.
28. Suponha que o roteador da extremidade esquerda da Figura 5.38 seja substituído por um comutador. Os Hospedeiros A, B, C e D e o roteador direito tem uma conexão estrela a esse comutador. Determine os endereços MAC de destino e origem no quadro que encapsula esse datagrama IP enquanto o quadro é transmitido (i) de A ao comutador, (ii) do comutador ao roteador direito, (iii) do roteador direito a F. Determine também os endereços IP de origem e destino no datagrama IP encapsulado pelo quadro em cada um desses pontos no tempo.
29. Considere a Figura 5.26. Suponha que todos os enlaces têm 100 Mbps. Qual é a vazão total máxima agregada que pode ser atingida entre os 9 hospedeiros e 2 servidores nessa rede? Suponha que qualquer hospedeiro ou servidor pode enviar a qualquer outro servidor ou hospedeiro. Por quê?
30. Suponha que três comutadores departamentais na Figura 5.26 são substituídos por hubs. Todos os enlaces têm 100 Mbps. Agora responda às questões postas no problema 29.
31. Suponha que todos os comutadores na Figura 5.26 são substituídos por hubs. Todos os enlaces têm 100 Mbps. Agora responda às questões postas no problema 29.
32. Vamos considerar a operação de aprendizagem do comutador no contexto da Figura 5.24. Suponha que (i) B envia um quadro a E, (ii) E responde com um quadro a B, (iii) A envia uma quadro a B, (iv) B responde com um quadro a A. A tabela do comutador está inicialmente vazia. Demonstre o estado da tabela do comutador antes e depois de cada um desses eventos. Para cada um dos eventos, identifique os enlaces em que o quadro transmitido será encaminhado, e brevemente justifique suas respostas.
33. Neste problema exploraremos o uso de pequenos pacotes de aplicações de Voz Sobre IP. Uma vantagem de um pacote pequeno é que uma grande parte da largura de banda do enlace é consumido por bytes de cabeçalho. Por isso, suponha que o pacote é formado por P bytes e 5 bytes de cabeçalho.
- Considere o envio de uma fonte de voz codificada digitalmente diretamente. Suponha que a fonte esteja codificada a uma taxa constante de 128 Kbps. Admita que cada pacote esteja integralmente cheio antes de a fonte enviá-lo para dentro da rede. O tempo exigido para encher um pacote é o **atraso de empacotamento**. Determine, em termos de P , o atraso de empacotamento em milissegundos.
 - Os atrasos de empacotamento maiores do que 20 milissegundos podem causar ecos perceptíveis e desagradáveis. Determine o atraso de empacotamento para $P = 1.500$ bytes (correspondente, aproximadamente, a um pacote Ethernet de tamanho máximo) e para $P = 50$ bytes (correspondente a um pacote ATM).
 - Calcule o atraso de armazenagem e repasse em um único comutador ATM para uma taxa de enlace $R = 622$ Mbps para $P = 1.500$ bytes e $P = 50$ bytes.
 - Comente as vantagens de usar um pacote de tamanho pequeno.
34. Considere o único comutador VLAN da Figura 5.30, e suponha que um roteador externo está conectado a porta 1 do comutador. Atribua endereços IP aos hospedeiros EE e CS e à interface do roteador. Relacione as etapas usadas em ambas camadas de rede e de enlace para transferir o datagrama IP ao hospedeiro EE e ao hospedeiro CS. (Dica: Leia novamente nosso estudo da Figura 5.19)
35. Considere a rede MPLS mostrada na Figura 5.36 e suponha que os roteadores R5 e R6 agora são habilitados para MPLS. Suponha que queremos executar engenharia de tráfego de modo que pacotes de R6 destinados a A sejam comutados para A via R6-R4-R2-R1. Mostre as tabelas MPLS em R5 e R6, bem como a tabela modificada em R4, que tornariam isso possível.
36. Considere a mesma situação do problema anterior, mas suponha que os pacotes de R6 destinados a D estão comutados via R6-R4-R3, enquanto os pacotes

- de R5 destinados a D sejam comutados via R4-R2-R1-R3. Apresente as tabelas MPLS em todos os roteadores que tornariam isso possível.
37. Neste problema, você juntará tudo que aprendeu sobre protocolos de Internet. Suponha que você entre em uma sala, conecte-se à Ethernet e quer fazer o download de uma página web. Quais são etapas de protocolo utilizadas, desde ligar o computador até receber a página web? Suponha que não tenha nada no seu DNS ou no seu navegador quando você ligar seu computador. (Dica: as etapas incluem o uso de protocolos da Ethernet, DHCP, ARP, DNS, TCP e HTTP). Indique explicitamente em suas etapas como obter os endereços MAC e IP de um roteador de borda.



Questões dissertativas

Você deve navegar na Web para responder às perguntas a seguir.

38. Qual é a faixa de preço aproximada atual para um adaptador Ethernet de 10/100 Mbps? E para um adaptador Gigabit Ethernet? Compare esses preços com o preço de um modem discado de 56 kbps ou de um modem ASDL.
39. O preço dos comutadores geralmente é determinado segundo o número de interfaces (também denominadas *portas* no jargão da LAN). Qual é a faixa de preço aproximada das interfaces para um comutador somente com interfaces de 100 Mbps?
40. Muitas das funções de um adaptador podem ser realizadas em software que executa no nó da CPU. Quais as vantagens e as desvantagens de deslocar essa funcionalidade do adaptador para o nó?
41. Utilize a Web para achar os números de protocolo usados em um quadro Ethernet para um datagrama IP e para um pacote ARP.
42. Leia as referências [Xiao, 2000; Huang 2002 e RFC 3346] sobre engenharia de tráfego utilizando MPLS. Relacione um conjunto de metas para a engenharia de tráfego. Quais dessas metas podem ser cumpridas somente com MPLS e quais delas são alcançadas utilizando protocolos existentes que não são MPLS? No último caso, quais são as vantagens que o MPLS oferece?



Wireshark Lab

No Companion Website, www.aw.com/kurose_br você encontrará duas tarefas de Wireshark que examinam a operação do protocolo IEEE 802.3 e o formato do quadro Wireshark.

A segunda Wireshark Lab examina a sequência dos pacotes usados numa situação de rede doméstica, parecida com a que estudamos na Seção 5.3.7

Entrevista

Simon S. Lam

Simon S. Lam é professor e regente da cadeira de ciência da computação na Universidade do Texas, em Austin. De 1971 a 1974 trabalhou no ARPA Network Measurement Center na UCLA, com comutação por satélite e por rádio. Liderou um grupo de pesquisa que inventou os sockets seguros e construiu o primeiro protótipo da camada de sockets seguros, em 1993, denominada Programação de Rede Segura. Os pontos de interesse de sua pesquisa são o projeto e a análise de protocolos de rede e serviços de segurança. Graduou-se em engenharia elétrica na Washington State University e é mestre e doutor pela UCLA. Foi eleito a Academia Nacional de Engenharia em 2007



O que o fez se decidir pela especialização em redes?

Quando cheguei à UCLA, recém-formado, no outono de 1969, minha intenção era estudar teoria de controle. Então assisti às aulas de Leonard Kleinrock sobre teoria das filas e ele me impressionou muito. Durante um certo tempo trabalhei em controle adaptativo de sistemas de fila como possível tópico de tese. No início de 1972, Larry Roberts iniciou o projeto ALOHA Satellite System (mais tarde denominado Packet Satellite). O professor Kleinrock perguntou se eu queria participar do projeto. A primeira coisa que fiz foi introduzir um algoritmo de backoff simples, mas realista, no protocolo da slotted ALOHA. Pouco tempo depois, encontrei muitos problemas interessantes para pesquisar, como o problema da instabilidade da ALOHA e a necessidade de backoff, que formariam o núcleo central da minha tese.

O senhor teve participação ativa nos primórdios da Internet na década de 1970, desde seus tempos de estudante na UCLA. Como era o panorama naquela época? As pessoas faziam alguma ideia do que a Internet se tornaria?

Na verdade, a atmosfera não era diferente da de outros projetos de construção de sistemas que já vi na indústria e no ambiente acadêmico. A meta inicial determinada para a ARPAnet era bastante modesta, isto é, prover acesso a computadores caros a partir de localizações remotas de modo que mais cientistas pudessem utilizá-los. Contudo, com o início do projeto Packet Satellite em 1972 e do projeto Packet Radio em 1973, a meta da ARPA foi ampliada substancialmente. Em 1973, a ARPA estava montando, ao mesmo tempo, três redes de pacotes diferentes e tornou-se necessário que Vint Cerf e Bob Kahn desenvolvessem uma estratégia de interconexão.

Naquela época, todos esses desenvolvimentos progressivos na área de redes eram vistos (acredito eu) mais como lógicos do que mágicos. Ninguém poderia ter previsto a escalada da Internet e o poder dos computadores de hoje. Transcorreu uma década antes do aparecimento dos primeiros PCs. Para colocar as coisas em perspectiva, a maioria dos estudantes apresentava seus programas de computador em cartões perfurados, para processamento em lote. Somente alguns deles tinham acesso direto a computadores, que normalmente eram abrigados em áreas restritas. Os modems eram lentos e ainda raros. No meu tempo de estudante, eu tinha apenas um telefone sobre minha mesa e usava papel e lápis para fazer a maior parte do meu trabalho.

Em sua opinião, qual é o futuro do campo das redes e da Internet?

No passado, a simplicidade do protocolo IP da Internet era sua maior força para vencer a concorrência e se tornar o padrão de facto do trabalho em redes. Diferentemente de seus concorrentes, como X-25 na década de 1980 e ATM na década de 1990, o IP pode rodar sobre qualquer tecnologia de camada de enlace porque oferece apenas um serviço datagrama de melhor esforço. Assim, qualquer rede de pacotes pode se conectar com a Internet.



Atualmente, a maior força do IP agora é uma deficiência. O IP é como uma camisa de força que confina o desenvolvimento da Internet a direções específicas. Nos últimos anos, muitos pesquisadores redirecionaram seus esforços somente para as camadas de aplicação e transporte, para suporte de *multicast* e QoS. Há também uma grande concentração de pesquisas em redes sem fio e redes ad hoc, redes de sensores e redes por satélite. Essas redes podem ser consideradas ou como sistemas autônomos ou como sistemas de camada de enlace, que podem se desenvolver porque estão fora da camisa de força do IP.

Há muita gente animada com a possibilidade da utilização de sistemas P2P como plataforma para novas aplicações da Internet. Todavia, a utilização de recursos da Internet por sistemas P2P é altamente ineficiente. Uma das minhas preocupações é se a capacidade de transmissão e comutação do núcleo da Internet continuará a crescer mais rapidamente do que a demanda de tráfego na Internet, enquanto ela cresce para interconectar todos os tipos de equipamentos e suportar futuras aplicações habilitadas para P2P. Sem um superprovisionamento substancial de capacidade, garantir a estabilidade da rede na presença de congestão e de ataques mal-intencionados seria uma tarefa muito importante.

O fenomenal crescimento da Internet também requer a alocação de novos endereços de IP em uma taxa rápida à operadores de rede e empreendimentos ao redor do mundo. Se continuarmos assim, o bloco de endereços IPv4 não alocados vão se esgotar em alguns anos. Quando isso acontecer, grandes blocos de espaços de endereços só poderão ser alocados somente de espaços de endereço IPv6. Já que a adoção do IPv6 não está fazendo muito sucesso, devido à falta de incentivo para os novos usuários, IPv4 e IPv6 vão coexistir por muitos anos. A migração bem-sucedida de uma Internet IPv4 para uma IPv6 precisará de um grande esforço global.

Qual parte de seu trabalho lhe apresenta mais desafios?

A parte do meu trabalho que me apresenta mais desafios é ensinar e motivar *todos* os estudantes que assistem às minhas aulas e *todos* os estudantes de doutorado que supervisiono, e não apenas os superbem-dotados. Os muito inteligentes e motivados podem exigir um pouco de supervisão, mas não muito mais do que isso. Muitas vezes aprendo mais com esses estudantes do que eles aprendem comigo. Ensinar e motivar os que não se destacam muito é um grande desafio.

Que impactos sobre o aprendizado o senhor acha que a tecnologia terá no futuro?

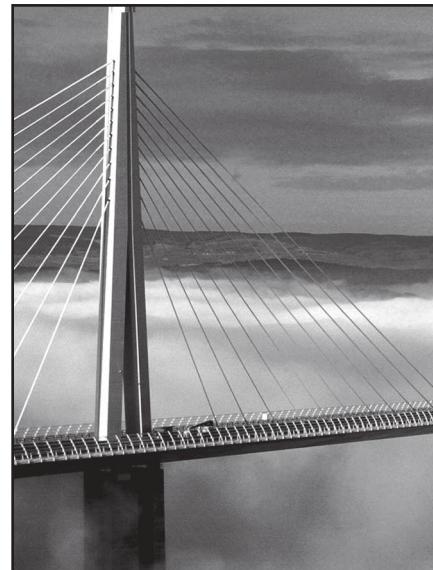
Finalmente, quase todo o conhecimento humano estará acessível pela Internet, algo que será a mais poderosa ferramenta de aprendizado. Essa vasta base de conhecimento terá o potencial de nivelar o terreno para estudantes em todo o mundo. Por exemplo, estudantes motivados de qualquer país poderão acessar os melhores sites Web de aulas, conferências multimídia e material de ensino. Já foi dito que as bibliotecas digitais do IEEE e da ACM aceleraram o desenvolvimento de pesquisadores da ciência da computação na China. Com o tempo, a Internet transcenderá todas as barreiras geográficas do aprendizado.





Capítulo 6

Redes sem fio e redes móveis



No mundo da telefonia pode-se dizer que os quinze últimos anos foram a década da telefonia celular. O número de assinantes de telefones móveis no mundo inteiro aumentou de 34 milhões em 1993 para quatro bilhões no final de 2008 e, agora, o número de assinantes da telefonia celular ultrapassa o número de linhas telefônicas convencionais [ITU Statistics, 2009]. As muitas vantagens dos telefones celulares são evidentes para todos — em qualquer lugar, a qualquer hora, acesso desimpedido à rede global de telefonia por meio de um equipamento leve e totalmente portátil. Com o advento de laptops, palmtops, PDAs e a promessa de acesso desimpedido à Internet global de qualquer lugar, a qualquer hora, será que estamos prestes a assistir a uma explosão semelhante da utilização de dispositivos sem fio para acesso à Internet?

Independentemente do crescimento futuro de equipamentos sem fio para Internet, já ficou claro que redes sem fio e os serviços móveis relacionados que elas possibilitam vieram para ficar. Do ponto de vista de rede, os desafios propostos por essas redes, particularmente nas camadas de enlace e de rede, são tão diferentes dos desafios das redes de computadores cabeadas que é necessário um capítulo inteiro (este capítulo) dedicado ao estudo de redes sem fio e redes móveis.

Iniciaremos este capítulo com uma discussão sobre usuários móveis, enlaces e redes sem fio e sua relação com as redes maiores (normalmente cabeadas) às quais se conectam. Traçaremos uma distinção entre os desafios propostos pela natureza *sem fio* dos enlaces de comunicação nessas redes e pela *mobilidade* que esses enlaces sem fio habilitam. Fazer essa importante distinção — entre sem fio e mobilidade — nos permitirá isolar, identificar e dominar melhor os conceitos fundamentais em cada área. Note que, na realidade, há muitos ambientes de rede nos quais os nós da rede são sem fio, mas não são móveis (por exemplo, redes residenciais sem fio ou redes de escritórios compostas por estações de trabalho estacionárias e monitores de grandes dimensões), e que existem formas limitadas de mobilidade que não requerem enlaces sem fio (por exemplo, um profissional que utiliza um laptop em casa, desliga o equipamento e o leva para seu escritório, onde o liga à rede cabeada da empresa em que trabalha). É claro que muitos dos ambientes sem fio mais interessantes são aqueles em que os usuários são sem fio e também móveis — por exemplo, um cenário no qual um usuário móvel (digamos, no banco traseiro de um carro) mantém uma chamada de voz sobre IP e várias conexões TCP ativas enquanto corre pela rodovia a 160 quilômetros por hora. É nesse ponto, em que o sem fio se cruza com a mobilidade, que encontraremos os desafios técnicos mais interessantes!

Começaremos por ilustrar, em primeiro lugar, o cenário no qual consideraremos comunicação e mobilidade sem fio — uma rede na qual usuários sem fio (e possivelmente móveis) estão conectados à infraestrutura da rede maior por um enlace sem fio na borda de rede. Então, consideraremos as características desse enlace sem fio na

Seção 6.2. Incluímos uma breve introdução ao Acesso Múltiplo por Divisão de Código (*Code Division Multiple Access* — CDMA), um protocolo de acesso ao meio compartilhado que é utilizado frequentemente em redes sem fio. Na Seção 6.3 estudaremos com certa profundidade os aspectos da camada de enlace do padrão da LAN sem fio IEEE 802.11 (Wi-Fi); também falaremos um pouco sobre Bluetooth e WiMAX. Na Seção 6.4 daremos uma visão geral do acesso celular à Internet, incluindo as tecnologias celulares emergentes 3G, que fornecem acesso à Internet por voz e a alta velocidade. Na Seção 6.5, voltaremos nossa atenção à mobilidade, focalizando os problemas da localização de um usuário móvel, do roteamento para o usuário móvel e da transferência (*handing off*) do usuário móvel que passa dinamicamente de um ponto de conexão com a rede para outro. Estudaremos como esses serviços de mobilidade são implementados no padrão IP móvel e em GSM nas Seções 6.6 e 6.7, respectivamente. Finalmente, na Seção 6.8 consideraremos o impacto dos enlaces e da mobilidade sem fio sobre protocolos de camada de transporte e aplicações em rede.

6.1 Introdução

A Figura 6.1 mostra o cenário no qual consideraremos os tópicos de comunicação de dados e mobilidade sem fio. Começaremos mantendo nossa discussão dentro de um contexto geral o suficiente para abranger uma ampla faixa de redes, entre elas LANs sem fio como a IEEE 802.11 e redes celulares como uma rede 3G; passaremos para uma discussão mais detalhada de arquiteturas sem fio específicas em seções posteriores. Podemos identificar os seguintes elementos em uma rede sem fio:

Hospedeiros sem fio. Como no caso de redes cabeadas (ou com fio), hospedeiros são os equipamentos de sistemas finais que executam aplicações. Um **hospedeiro sem fio** pode ser um laptop, um palmtop, um PDA, um telefone ou um computador de mesa. Os próprios hospedeiros podem ser ou não ser móveis.

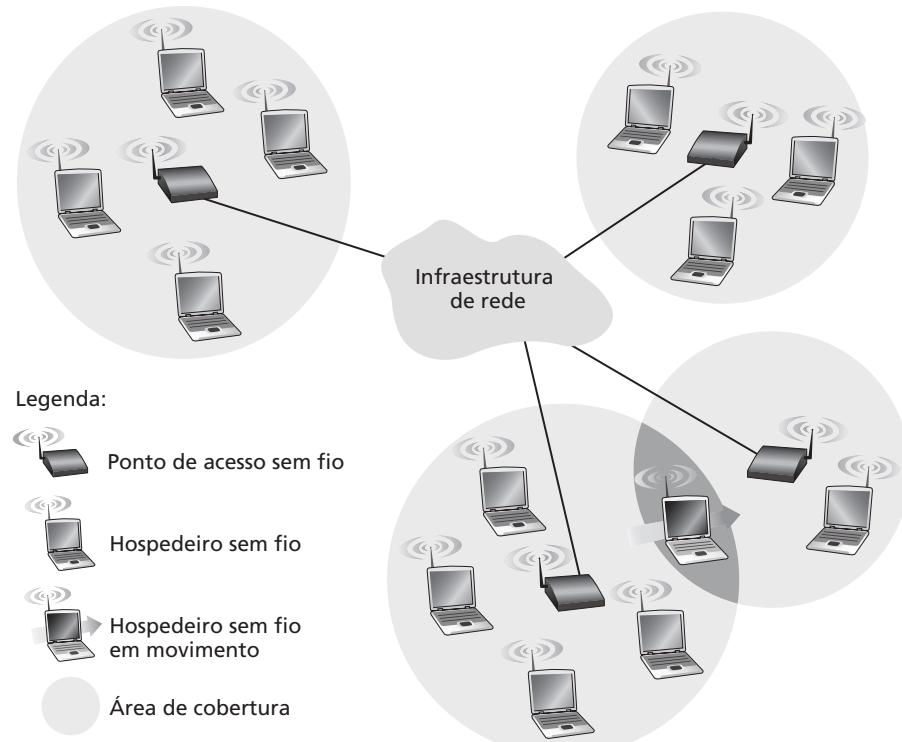


Figura 6.1 Elementos de uma rede sem fio

Enlaces sem fio. Um hospedeiro se conecta a uma estação-base (definida mais adiante) ou a um outro hospedeiro sem fio por meio de um **enlace de comunicação sem fio**. Tecnologias de enlace sem fio diferentes têm taxas de transmissão diferentes e podem transmitir a distâncias diferentes. A Figura 6.2 mostra duas características fundamentais (área de cobertura e taxa de enlace) dos padrões de enlace sem fio mais populares. (A figura serve somente para mostrar uma ideia aproximada dessas características. Por exemplo, alguns desses tipos de redes somente estão sendo empregados agora, e algumas taxas de enlace podem aumentar ou diminuir além dos valores mostrados dependendo da distância, condições do canal e do número de usuários na rede sem fio.) Abordaremos esses padrões mais adiante na primeira metade deste capítulo; consideraremos também outras características de enlaces sem fio (como suas taxas de erros de bits e as causas desses erros) na Seção 6.2.

Na Figura 6.1 enlaces sem fio conectam hospedeiros localizados na borda da rede com a infraestrutura da rede de maior porte. Não podemos nos esquecer de acrescentar que enlaces sem fio às vezes também são utilizados *dentro* de uma rede e para conectar roteadores, comutadores e outros equipamentos de rede. Contudo, neste capítulo, focalizaremos a utilização de comunicação sem fio nas bordas da rede, pois é aqui que estão ocorrendo muitos dos desafios técnicos mais interessantes e a maior parte do crescimento.

Estação-base. A **estação-base** é uma parte fundamental da infraestrutura de rede sem fio. Diferentemente dos hospedeiros e enlaces sem fio, uma estação-base não tem nenhuma contraparte óbvia em uma rede cabeada. Uma estação-base é responsável pelo envio e recebimento de dados (por exemplo, pacotes) de e para um hospedeiro sem fio que está associado com ela. Uma estação-base frequentemente será responsável pela coordenação da transmissão de vários hospedeiros sem fio com os quais está associada. Quando dizemos que um hospedeiro sem fio está ‘associado’ com uma estação-base, isso quer dizer que (1) o hospedeiro está dentro do alcance de comunicação sem fio da estação-base e (2) o hospedeiro usa a estação-base para retransmitir dados entre ele (o hospedeiro) e a rede maior. **Torres celulares** em redes celulares e **pontos de acesso** em uma LAN sem fio 802.11 são exemplos de estações-base.

Na Figura 6.1, a estação-base está conectada à rede maior (isto é, à Internet, à rede empresarial ou residencial ou à rede telefônica) e, portanto, funciona como uma retransmissora de camada de enlace entre o hospedeiro sem fio e o resto do mundo com o qual o hospedeiro se comunica.

Quando hospedeiros estão associados com uma estação-base, em geral diz-se que estão operando em **modo de infraestrutura**, já que todos os serviços tradicionais de rede (por exemplo, atribuição de endereço e roteamento) são fornecidos pela rede com a qual estiverem conectados por meio da estação-base. Em **redes ad hoc**, hospedeiros sem fio não dispõem de nenhuma infraestrutura desse tipo com

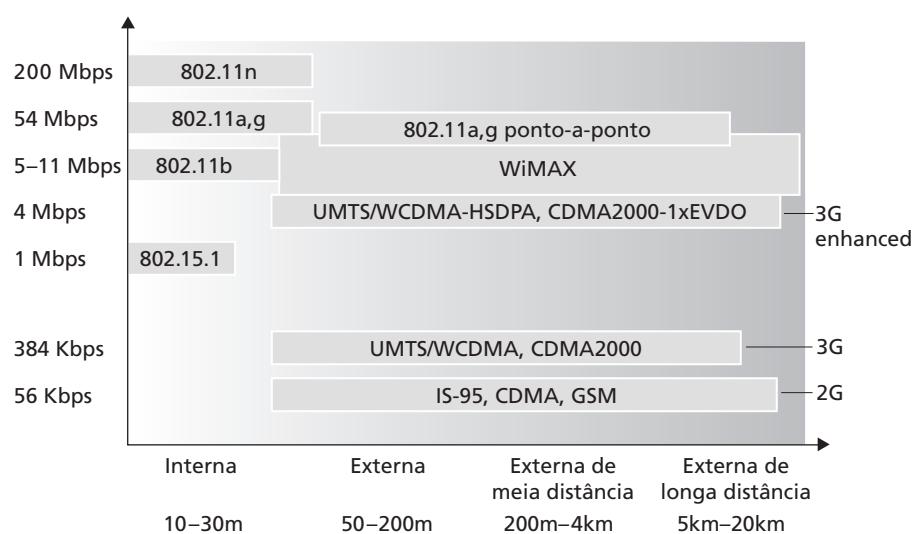


Figura 6.2 Características de enlaces de padrões selecionados de rede sem fio

a qual se conectar. Na ausência de tal infraestrutura, os próprios hospedeiros devem prover serviços como roteamento, atribuição de endereço, tradução de endereços semelhante ao DNS e outros.

Quando um hospedeiro móvel se desloca para fora da faixa de alcance de uma estação-base e entra na faixa de uma outra, ele muda seu ponto de conexão com a rede maior (isto é, muda a estação-base com a qual está associado) — um processo denominado **transferência** (handoff). Essa mobilidade dá origem a muitas questões desafiadoras. Se um hospedeiro pode se mover, como descobrir sua localização corrente na rede de modo que seja possível lhe transmitir dados? Como é realizado o endereçamento, dado que um hospedeiro pode estar em uma entre muitas localizações possíveis? Se o hospedeiro se movimentar *durante* uma conexão TCP ou ligação telefônica, como os dados serão roteados de modo que a conexão continue sem interrupção? Essas e muitas (mas muitas!) outras questões fazem das redes sem fio e móveis uma área muito interessante da pesquisa de redes.

Infraestrutura de rede. É a rede maior com a qual um hospedeiro sem fio pode querer se comunicar.

Após discutir sobre as “partes” da rede sem fio, observamos que essas partes podem ser combinadas de diversas maneiras diferentes para formar diferentes tipos de redes sem fio. Você pode achar uma taxonomia desses tipos de redes sem fio útil ao ler este capítulo, ou ler/aprender mais sobre redes sem fio além deste livro. No nível mais alto, podemos classificar as redes sem fio de acordo com dois critérios: (i) ou um pacote na rede sem fio atravessa exatamente *um salto sem fio* ou *diversos saltos sem fio*, e (ii) ou há *infraestrutura*, como uma estação-base na rede:

Salto único, com infraestrutura. Essas redes têm uma estação-base conectada a um rede cabeada maior (por exemplo, a Internet). Além disso, toda a comunicação é feita entre a estação-base e um hospedeiro sem fio através de um salto sem fio único. As redes 802.11 que você utiliza na sala de aula, no café ou na biblioteca; redes de telefonia celular; e as redes WiMAX 802.16 que aprenderemos de forma breve, encaixando-se nesta categoria.

Salto único, sem infraestrutura. Nessas redes, não existe estação-base conectada à rede sem fio. Entretanto, como veremos, um dos nós nessa rede de salto único pode coordenar as transmissões dos outros nós. As redes Bluetooth (que serão estudadas na Seção 6.3.6) e as redes 802.11 no modo ad hoc são redes de salto único, sem infraestrutura.

Múltiplos saltos, com infraestrutura. Nessas redes, está presente uma estação-base cabeada para as redes maiores. Entretanto, alguns nós sem fio podem ter que reestabelecer sua comunicação através de outros nós sem fio para se comunicarem por meio de uma estação-base. Algumas redes de sensores sem fio e as redes em malha sem fio se encaixam nesta categoria.

Múltiplos saltos, sem infraestrutura. Não existem estações de base nessas redes, e os nós podem ter de reestabelecer mensagens entre diversos outros nós para chegar a um destino. Os nós podem ser móveis, ocorrendo mudança de conectividade entre eles — uma categoria de redes conhecida como as redes móveis ad hoc (MANETs). Se os nós móveis forem veículos, essa rede é denominada rede veicular ad hoc (VANET). Como você pode imaginar, o desenvolvimento de protocolos para essas redes é desafiante e constitui o assunto de muita pesquisa em andamento.

Neste capítulo, iremos nos delimitar às redes de salto único e, depois, às redes com infraestrutura.

Agora vamos nos aprofundar um pouco mais nos desafios técnicos que surgem em redes sem fio e móveis. Começaremos considerando, em primeiro lugar, o enlace sem fio individual, adiando nossa discussão da mobilidade para outra parte deste capítulo.

6.2 Características de enlaces e redes sem fio

Vamos começar considerando uma rede simples cabeada, por exemplo, uma rede residencial com hospedeiros interconectados por um comutador Ethernet cabeado (ver Seção 5.6). Se substituíssemos a Ethernet cabeada por uma rede 802.11 sem fio, uma placa NIC sem fio substituiria as placas da Ethernet cabeada nos hospedeiros e um ponto de acesso substituiria o comutador Ethernet, mas, na camada de rede ou acima dela, praticamente nenhuma mudança seria necessária. Isso sugere que concentremos nossa atenção na camada de enlace ao pro-

História

Acesso público Wi-Fi: em breve facilmente a seu dispor?

Pontos de acesso Wi-Fi — locais públicos onde os usuários podem encontrar acesso sem fio 802.11 — estão se tornando cada vez mais comuns em hotéis, aeroportos e cafés ao redor do mundo. A partir do final de 2008, a T-Mobile oferece pontos de acesso em mais de 10 mil locais nos Estados Unidos (e mais de 45 mil no mundo), incluindo as cafeteria Starbucks e as lojas Borders Books & Music. A maioria dos campi universitários oferecem acesso sem fio espalhado por toda a parte, e é difícil encontrar um hotel que não oferece acesso à Internet sem fio. Diversas cidades, incluindo Filadélfia, San Francisco, Toronto e Hong Kong, anunciaram planos de prover esse acesso sem fio dentro de toda a cidade. O objetivo na Filadélfia foi “transformar a Filadélfia no maior ponto de acesso Wi-Fi do país e ajudar a melhorar a educação, eliminar a exclusão digital, aprimorar o desenvolvimento da região e reduzir os custos do governo”. O plano inicialmente exigiu a instalação de pontos de acesso sem fio 802.11 em aproximadamente 4 mil postes de iluminação e mecanismos de controle de tráfego. O ambicioso programa — um acordo entre a cidade, a Weireless Philadelphia (entidade sem fins lucrativos) e o ISP Earthlink — construiu uma rede operacional. O projeto GovWiFi de Hong Kong “proverá serviço Wi-Fi grátis em 350 dependências do governo em fases. Estabelecerá cerca de 2 mil pontos de acesso Wi-Fi públicos no território e tornará Hong Kong uma cidade sem fio com aproximadamente dez mil pontos de acesso Wi-Fi públicos em 2009.”

Criar planos para o Wi-Fi municipal onipresente, entretanto, provou ser uma tarefa difícil. A rede Wi-Fi municipal de San Francisco nunca passou da proposta. Em 2008, o Earthlink cancelou seu serviço Wi-Fi na Filadélfia. Mas a rede municipal paga de Toronto continua operando, assim como as redes Wi-Fi municipais em diversas cidades. A busca por rede sem fio municipal continua. O <http://www.muniwireless.com> é um site que acompanha o cenário das redes sem fio municipais.

curarmos diferenças importantes entre redes com fio e sem fio. Realmente, podemos encontrar várias diferenças importantes entre um enlace com fio e um enlace sem fio:

Redução da força do sinal. Radiações eletromagnéticas são atenuadas quando atravessam algum tipo de matéria (por exemplo, um sinal de rádio ao atravessar uma parede). O sinal se dispersará mesmo ao ar livre, resultando na redução de sua força (às vezes denominada **atenuação de percurso**) à medida que aumenta a distância entre emissor e receptor.

Interferência de outras fontes. Várias fontes de rádio transmitindo na mesma banda de frequência sofrem interferênciaumas das outras. Por exemplo, telefones sem fio de 2,4 GHz e LANs sem fio 802.11b transmitem na mesma banda de frequência. Assim, o usuário de uma LAN sem fio 802.11b que estiver se comunicando por um telefone sem fio de 2,4 GHz pode esperar que nem a rede nem o telefone funcionem particularmente bem. Além da interferência de fontes transmissoras, o ruído eletromagnético presente no ambiente (por exemplo, um motor ou um equipamento de microondas próximo) pode resultar em interferência.

Propagação multivias. **Propagação multivias** (ou multicaminhos) ocorre quando porções da onda eletromagnética se refletem em objetos e no solo e tomam caminhos de comprimentos diferentes entre um emissor e um receptor. Isso resulta no embaralhamento do sinal recebido no destinatário. Objetos que se movem entre o emissor e o receptor podem fazer com que a propagação multivias mude ao longo do tempo.

Para uma discussão detalhada sobre as características, canais, modelos e medidas do canal sem fio, consulte [Anderson, 1995].

A discussão anterior sugere que erros de bits serão mais comuns em enlaces sem fio do que em enlaces com fio. Por essa razão, talvez não seja nenhuma surpresa que protocolos de enlace sem fio (como o protocolo 802.11 que examinaremos na seção seguinte) empreguem não somente poderosos códigos de detecção de erros por CRC, mas também protocolos ARQ de nível de enlace que retransmitem quadros corrompidos.

Tendo considerado as falhas que podem ocorrer em um canal sem fio, vamos voltar nossa atenção para o hospedeiro que recebe o sinal sem fio. Esse hospedeiro recebe um sinal eletromagnético que é uma combinação de uma forma degradada do sinal original transmitido pelo remetente (degradada devido aos efeitos da atenuação e da propagação multivias discutidas acima, entre outros) e um ruído de fundo no ambiente. A razão sinal-ruído (SNR) é uma medida relativa da potência do sinal recebido (ou seja, a informação sendo transmitida) e o ruído. O SNR é, normalmente, medido em unidades de decibéis (dB), uma unidade de medida que algumas pessoas acham que é utilizada por engenheiros elétricos para confundir cientistas da computação. O SNR, medido em dB, é vinte vezes a razão do logaritmo de base 10 da amplitude do sinal recebido à amplitude do ruído. Para nossos fins, precisamos saber somente que um SNR maior facilita ainda mais para o destinatário extrair o sinal transmitido de um ruído de fundo.

A Figura 6.3 (adaptada de [Holland, 2001]) mostra a taxa de erro de bits (BER) — grosso modo, a probabilidade de um bit transmitido ser recebido com erro no destinatário — versus o SNR para três técnicas de modulação diferentes para codificar informações para a transmissão em um canal sem fio idealizado. A teoria da modulação e da codificação, bem como a extração do sinal e a BER, vai além do objetivo deste livro (consulte [Schwartz, 1980] para uma discussão sobre esses assuntos). Não obstante, o Figura 6.3 ilustra diversas características da camada física que são importantes na compreensão dos protocolos de comunicação sem fio da camada mais superior:

Para um determinado esboço da modulação, quanto mais alto for o SNR, mais baixo será a BER. Visto que um remetente pode aumentar o SNR elevando sua potência de transmissão, ele pode reduzir a probabilidade de um quadro ser recebido com erro diminuindo sua potência de transmissão. Observe, entretanto, que há um pequeno ganho prático no aumento da potência além de um certo patamar, digamos que para aumentar a BER de 10^{-12} para 10^{-13} . Existem também desvantagens associadas com o aumento da potência de transmissão: mais energia deve ser gasta pelo remetente (uma consideração importante para usuários móveis que utilizam bateria), e as transmissões do remetente têm mais probabilidade de interferir nas transmissões de outro remetente (consulte Figura 6.4 (b)).

Para um determinado SNR, uma técnica de modulação com um taxa de transmissão de bit maior (com erro ou não) terá uma BER maior. Por exemplo, na Figura 6.3, com um SNR de 10 dB, a modulação BPSK com uma taxa de transmissão de 1 Mbps possui uma BER menor do que 10^{-7} , enquanto a modulação QAM16 com uma taxa de transmissão de 4 Mbps, a BER é 10^{-1} , longe de ser praticamente útil. Entretanto, com um SNR de 20 dB, a modulação QAM16 possui uma taxa de transmissão de 4 Mbps e uma BER de 10^{-7} , enquanto a modulação BPSK possui uma taxa de transmissão de apenas 1 Mbps e uma

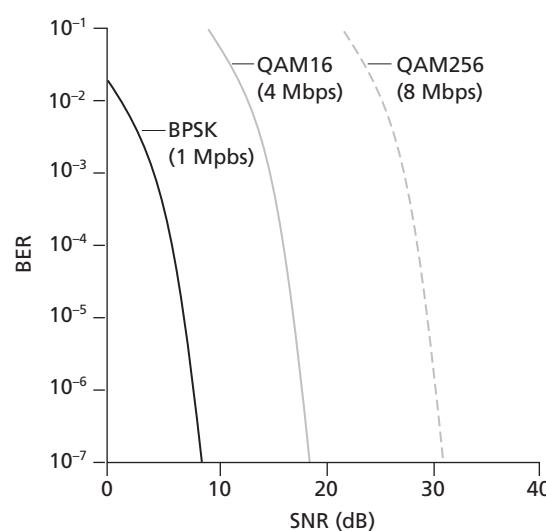


Figura 6.3 Taxa de erro de bits, taxa de transmissão e SNR

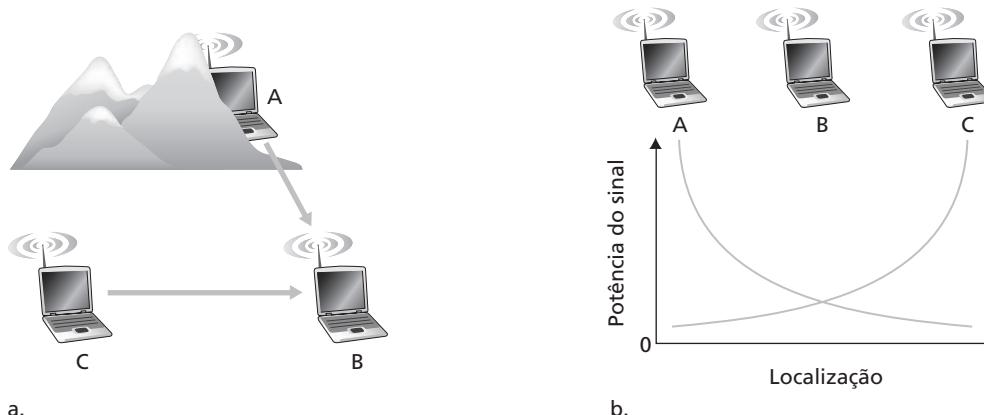


Figura 6.4 Problema do terminal oculto (a) e do desvanecimento (b)

BER tão baixa como estar (literalmente) “fora das paradas”. Se é possível suportar uma BER de 10^{-7} , a taxa de transmissão mais alta apresentada pela modulação QAM 16 a tornaria a técnica de modulação preferida nesta situação. Essas considerações dão origem à característica final, descrita abaixo.

Seleção dinâmica da técnica de modulação da camada física pode ser usada para adaptar a técnica de modulação para condições de canal. O SNR (e, portanto, a BER) pode alterar como resultado da mobilidade ou em razão das mudanças no ambiente. A modulação adaptiva e a codificação são usadas em sistemas de dados celulares, no WiMAX 802.16 e nas redes Wi-Fi 802.11 que estudaremos na Seção 6.3. Isso permite, por exemplo, a seleção de uma técnica de modulação que oferece a mais alta taxa de transmissão possível sujeita a uma limitação na BER, para determinadas características do canal.

Taxas de erros de bits mais altas e que variam ao longo do tempo não são as únicas diferenças entre um enlace com fio e um enlace sem fio. Lembre-se de que, no caso de enlaces broadcast cabeados, todos os nós recebem as transmissões de todos os outros nós. No caso de enlaces sem fio, a situação não é tão simples, como mostra a Figura 6.4.

Suponha que a estação A esteja transmitindo para a estação B. Suponha também que a estação C esteja transmitindo para a estação B. O denominado **problema do terminal oculto**, obstruções físicas presentes no ambiente (por exemplo, uma montanha ou um edifício), pode impedir que A e C escutem as transmissões de um e de outro, mesmo que as transmissões de A e C estejam interferindo no destino, B. Isso é mostrado na Figura 6.4(a). Um segundo cenário que resulta em colisões que não são detectadas no receptor é causado pelo **desvanecimento** da força de um sinal à medida que ele se propaga pelo meio sem fio. A Figura 6.4(b) ilustra o caso em que a localização de A e C é tal que as potências de seus sinais não são suficientes para que eles detectem as transmissões de um e de outro, mas, mesmo assim, são suficientemente fortes para interferir uma com a outra na estação B. Como veremos na Seção 6.3, o problema do terminal oculto e o desvanecimento tornam o acesso múltiplo em uma rede sem fio consideravelmente mais complexo do que em uma rede cabeadas.

6.2.1 CDMA

Lembre-se de que dissemos, no Capítulo 5, que, quando hospedeiros se comunicam por um meio compartilhado, é preciso um protocolo para que os sinais enviados por vários emissores não interfiram nos receptores. No mesmo capítulo descrevemos três classes de protocolos de acesso ao meio: de partição de canal, de acesso aleatório e de revezamento. O acesso múltiplo por divisão de código (Code Division Multiple Access — CDMA) é ainda um quarto tipo de protocolo de acesso a um meio compartilhado, predominante em tecnologias celulares e de LAN sem fio. Por ser tão importante no mundo sem fio, examinaremos o CDMA rapidamente agora, antes de passar para tecnologias específicas de acesso sem fio nas seções subsequentes.

Com um protocolo CDMA, cada bit que está sendo enviado é codificado pela multiplicação do bit por um sinal (o código) que muda a uma velocidade muito maior (conhecida como **velocidade de chipping**) do que a

sequência original de bits de dados. A Figura 6.5 mostra um cenário simples e idealizado de codificação/decodificação CDMA. Suponha que a velocidade com que bits de dados originais cheguem ao codificador CDMA defina a unidade de tempo, isto é, que cada bit original de dados a ser transmitido requeira um intervalo de tempo de um bit. Seja d_i o valor do bit de dados para o i -ésimo intervalo de bit. Por conveniência do cálculo matemático, representamos o bit de dados com valor 0 por -1. Cada intervalo de bit é ainda subdividido em M mini-intervalos. Na Figura 6.5, $M = 8$, embora, na prática, M seja muito maior. O código CDMA usado pelo remetente consiste em uma sequência de M valores, c_m , $m = 1, \dots, M$, cada um assumindo um valor de +1 ou -1. No exemplo da Figura 6.4, o código CDMA de M bits que está sendo usado pelo remetente é (1, 1, 1, -1, 1, -1, -1, -1).

Para ilustrar como o CDMA funciona, vamos focalizar o i -ésimo bit de dados, d_i . Para o m -ésimo mini-intervalo de tempo de transmissão de bits d_i , a saída do codificador CDMA, $Z_{i,m}$, é o valor de d_i multiplicado pelo m -ésimo bit do código CDMA escolhido, c_m :

$$Z_{i,m} = d_i \cdot c_m \quad (6.1)$$

Se o mundo fosse simples e não houvesse remetentes interferindo, o receptor receberia os bits codificados, $Z_{i,m}$, e recuperaria os bits de dados originais, d_i , calculando:

$$d_i = \frac{1}{M} \sum_{m=1}^M Z_{i,m} \cdot c_m \quad (6.2)$$

Talvez o leitor queira repassar os detalhes do exemplo da Figura 6.4 para verificar se os bits originais de dados são, de fato, corretamente recuperados no receptor usando a Equação 6.2.

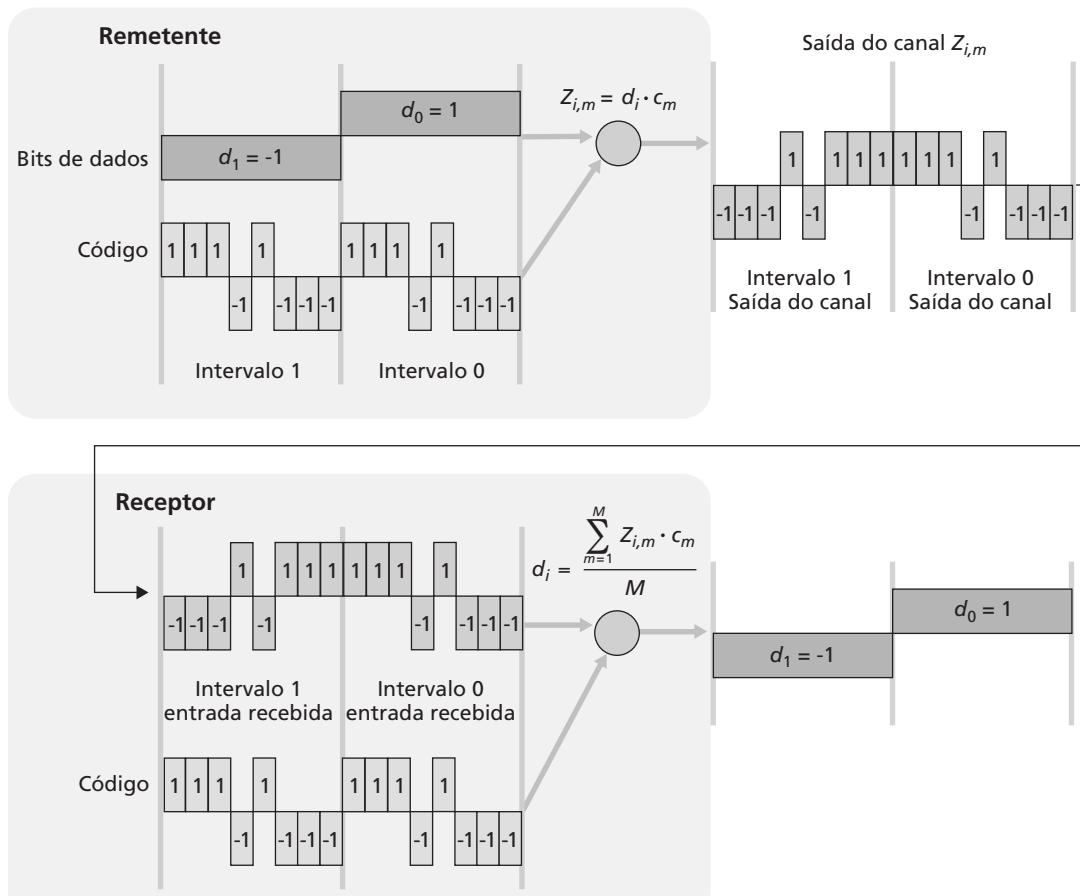


Figura 6.5 Um exemplo simples de CDMA: codificação no remetente, decodificação no receptor

No entanto, o mundo está longe de ser ideal e, como mencionamos antes, o CDMA deve funcionar na presença de remetentes que interferem e que estão codificando e transmitindo seus dados usando um código designado diferente. Mas, como um receptor CDMA pode recuperar bits de dados originais de um remetente quando esses bits de dados estão sendo embaralhados com bits que estão sendo transmitidos por outros remetentes? O CDMA funciona na presunção de que os sinais de bits interferentes que estão sendo transmitidos são aditivos. Isso significa, por exemplo, que, se três remetentes enviam um valor 1 e um quarto remetente envia um valor -1 durante o mesmo mini-intervalo, então o sinal recebido em todos os receptores durante o mini-intervalo é 2 (já que $1 + 1 + 1 - 1 = 2$). Na presença de vários remetentes, o remetente s calcula suas transmissões codificadas, $Z_{i,m}^s$, exatamente como na Equação 6.1. O valor recebido no receptor durante o m -ésimo mini-intervalo do i -ésimo intervalo de bit, contudo, é agora a soma dos bits transmitidos de todos os N remetentes durante o mini-intervalo:

$$Z_{i,m}^* = \sum_{s=1}^N Z_{i,m}^s$$

Surpreendentemente, se os códigos dos remetentes forem escolhidos com cuidado, cada receptor pode recuperar os dados enviados por um dado remetente a partir do sinal agregado apenas usando o código do remetente, como na Equação 6.2:

$$d_i = \frac{1}{M} \sum_{m=1}^M Z_{i,m}^* \cdot c_m \quad (6.3)$$

A Figura 6.6 ilustra um exemplo de CDMA com dois remetentes. O código CDMA de M bits que está sendo usado pelo remetente que está acima é $(1, 1, 1, -1, 1, -1, -1, -1)$, ao passo que o código CDMA usado pelo remetente que está embaixo é $(1, -1, 1, 1, 1, -1, 1, 1)$. A Figura 6.6 ilustra um receptor recuperando os bits de dados originais do remetente que está acima. Note que o receptor pode extrair os dados do remetente 1, a despeito da transmissão interferente do remetente 2.

Voltando à analogia do coquetel apresentada no Capítulo 5, um protocolo CDMA é semelhante à situação em que os convidados falam vários idiomas; nessa circunstância, os seres humanos até que são bons para manter conversações no idioma que entendem e, ao mesmo tempo, continuar filtrando outras conversações. Vemos aqui que o CDMA é um protocolo de partição, pois reparte o espaço de código (e não o tempo ou a frequência) e atribui a cada nó uma parcela dedicada do espaço de código.

Nossa discussão do código CDMA aqui é necessariamente breve; na prática devem ser abordadas inúmeras questões diferentes. Em primeiro lugar, para que receptores CDMA possam extrair o sinal de um emissor particular, os códigos CDMA devem ser escolhidos cuidadosamente. Em segundo lugar, em nossa discussão admitimos que as forças dos sinais recebidos de vários emissores são as mesmas; na realidade, isso pode ser difícil de conseguir. Existe uma quantidade considerável de literatura que aborda essas e outras questões relativas ao CDMA; veja [Pickholtz, 1982; Viterbi, 1995], se quiser mais detalhes.

6.3 Wi-Fi: LANs sem fio 802.11

Presentes no local de trabalho, em casa, em instituições educacionais, em cafés, aeroportos e esquinas, as LANs sem fio agora são uma das mais importantes tecnologias de rede de acesso na Internet de hoje. Embora muitas tecnologias e padrões para LANs sem fio tenham sido desenvolvidos na década de 1990, uma classe particular de padrões surgiu claramente como a vencedora: a **LAN sem fio IEEE 802.11**, também conhecida como **Wi-Fi**. Nesta seção estudaremos mais detalhadamente as LANs sem fio 802.11, examinando a estrutura do quadro 802.11, o protocolo 802.11 de acesso ao meio e a interconexão de LANs 802.11 com LANs Ethernet cabeadas.

Há diversos padrões 802.11 para tecnologia de LAN sem fio, entre eles 802.11b, 802.11a e 802.11g. A Tabela 6.1 apresenta um resumo das principais características desses padrões. Na época em que escrevímos este livro (primeiro semestre de 2009), mais mecanismos 802.11g eram oferecidos por pontos de acesso e por fornecedores de placas LAN. Estão também disponíveis um número de mecanismos de modos duplo (802.11a/g) e triplo (802.11a/b/g).

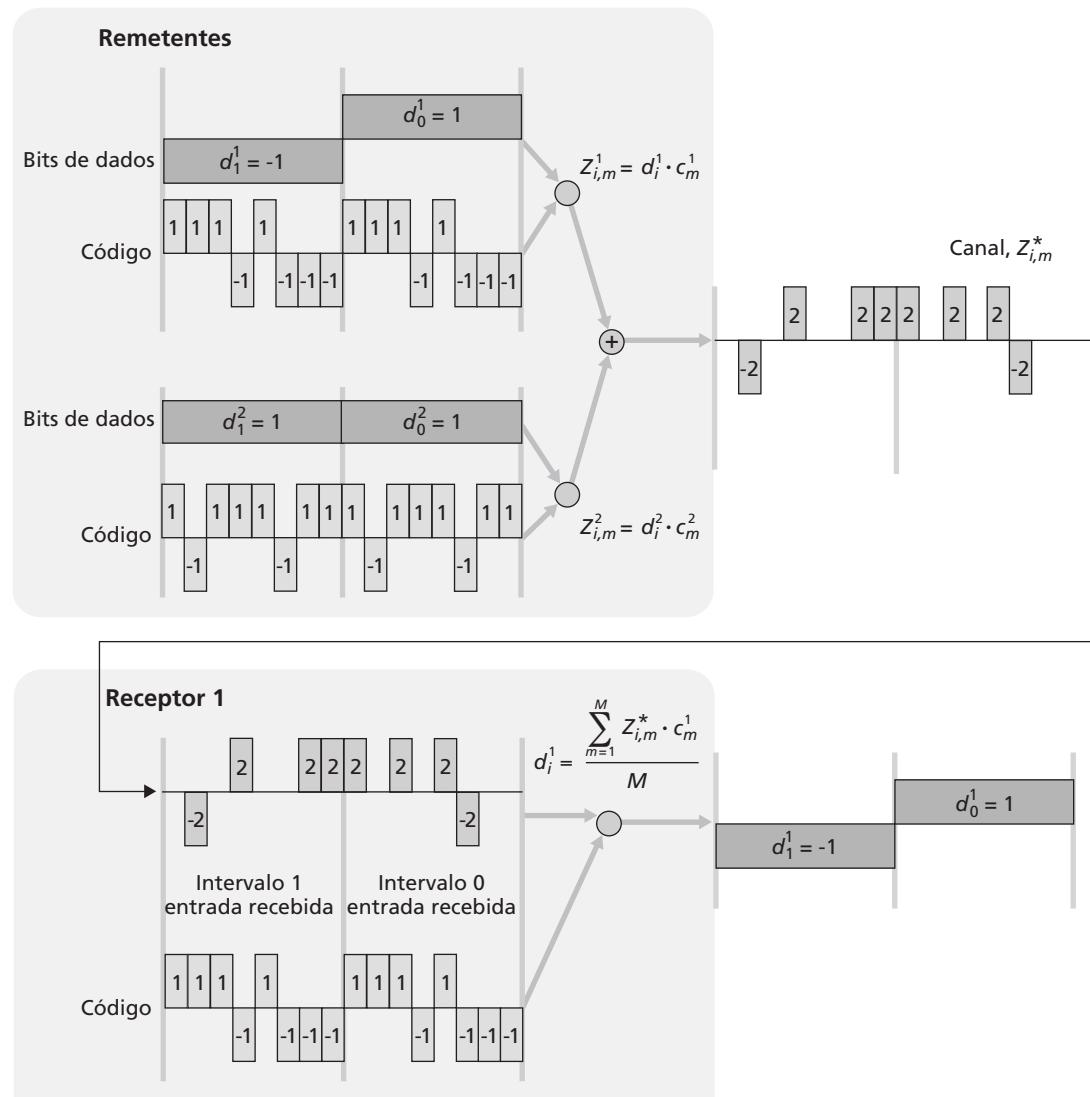


Figura 6.6 Um exemplo de CDMA com dois remetentes

Os três padrões 802.11 compartilham muitas características. Todos usam o mesmo protocolo de acesso ao meio, CSMA/CA, que discutiremos em breve. Todos os três também usam a mesma estrutura de quadro para seus quadros de camada de enlace. Todos os três padrões têm a capacidade de reduzir sua taxa de transmissão para alcançar distâncias maiores. E todos os três padrões permitem ‘modo de infraestrutura’ e ‘modo ad hoc’, como discutiremos em breve. Contudo, como mostra a Tabela 6.1, os três padrões apresentam algumas diferenças importantes na camada física.

A LAN sem fio 802.11b tem uma taxa de dados de 11 Mbps e opera na faixa de frequência não licenciada de 2,4 a 2,485 GHz, competindo por espectro de frequência com telefones e fornos de micro-ondas de 2,4 GHz. LANs sem fio 802.11a podem funcionar a taxas de bits significativamente mais altas, porém em frequências mais altas. Como operam a uma frequência mais alta, a distância de transmissão dessas LANs é mais curta para um dado nível de potência e elas sofrem mais com a propagação multivias. LANs sem fio 802.11g, que operam na mesma faixa de frequência mais baixa das LANs 802.11b e que são compatíveis com a 802.11b (para que seja possível atualizar clientes 802.11b de forma crescente), porém com as taxas de transmissão mais altas da 802.11a, devem permitir que os usuários tenham o melhor dos dois mundos.

Padrão	Faixa de frequência	Taxa de dados
802.11b	2,4–2,485 GHz	até 11 Mbps
802.11a	5,1–5,8 GHz	até 54 Mbps
802.11g	2,4–2,485 GHz	até 54 Mbps

Tabela 6.1 Resumo de padrões IEEE 802.11

Um novo padrão Wi-Fi, 802.11n [IEEE 802.11n 2009], está em processo de padronização. 802.11n utiliza antenas de saída múltipla e entrada múltipla (MIMO); ou seja, duas ou mais antenas no lado remetente e duas ou mais antenas no lado destinatário que estão transmitindo/recebendo sinais diferentes [Diggavi, 2004]. Embora o padrão ainda deva ser finalizado, estão disponíveis os produtos pré-padrão, sendo que os testes iniciais mostram a vazão de mais de 200 Mbps na prática [Newman, 2008]. Uma consideração importante em relação ao padrão atual é a maneira na qual os mecanismos 802.11 interagirão com os mecanismos 802.11a/b/g existentes.

6.3.1 A arquitetura 802.11

A Figura 6.7 ilustra os principais componentes da arquitetura de LAN sem fio 802.11. O bloco construtivo fundamental da arquitetura 802.11 é o **conjunto básico de serviço** (*basic service set* — BSS). Um BSS contém uma ou mais estações sem fio e uma **estaçao-base central**, conhecida como um **ponto de acesso** (*access point* — AP) na terminologia da 802.11. A Figura 6.7 mostra o AP em cada um dos dois BSSs conectando-se a um dispositivo de interconexão (tal como um hub, um comutador ou um roteador), que, por sua vez, leva à Internet. Em uma rede residencial típica, há apenas um AP e um roteador (normalmente integrados como uma unidade) que conecta o BSS à Internet.

Como acontece com dispositivos Ethernet, cada estação sem fio 802.11 tem um endereço MAC de 6 bytes que é armazenado no suporte lógico inalterável (firmware) do adaptador da estação (isto é, na placa de interface de rede 802.11). Cada AP também tem um endereço MAC para sua interface sem fio. Como na Ethernet, esses endereços MAC são administrados pelo IEEE e são (em teoria) globalmente exclusivos.

Como observamos na Seção 6.1, LANs sem fio que disponibilizam APs normalmente são denominadas **LANS sem fio de infraestrutura** e, nesse contexto, ‘infraestrutura’ quer dizer os APs juntamente com a infraestrutura de

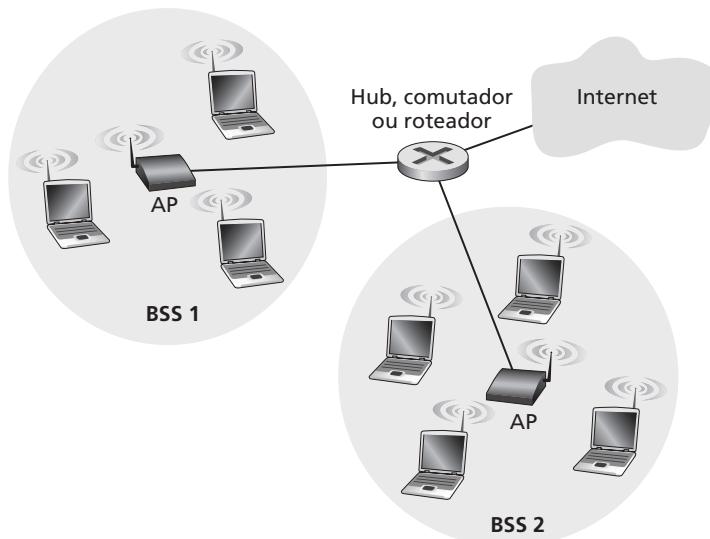


Figura 6.7 Arquitetura da LAN IEEE 802.11

Ethernet cabeada que interconecta os APs e um roteador. A Figura 6.8 mostra que estações IEEE 802.11 também podem se agrupar e formar uma rede ad hoc — rede sem nenhum controle central e sem nenhuma conexão com o ‘mundo externo’. Nesse caso, a rede é formada conforme a necessidade, por equipamentos móveis que, por acaso, estão próximos uns dos outros, têm necessidade de se comunicar e não dispõem de infraestrutura de rede no lugar em que se encontram. Uma rede ad hoc pode ser formada quando pessoas que portam laptops se reúnem (por exemplo, em uma sala de conferências, um trem ou um carro) e querem trocar dados na ausência de um AP centralizado. As redes ad hoc estão despertando um interesse extraordinário com a contínua proliferação de equipamentos portáteis que podem se comunicar. Porém, nesta seção, concentraremos nossa atenção em LANS sem fio de infraestrutura.

Canais e associação

Em 802.11, cada estação sem fio precisa se associar com um AP antes de poder enviar ou receber quadros 802.11 contendo dados de camada de rede. Embora todos os padrões 802.11 usem associação, discutiremos esse tópico especificamente no contexto da IEEE 802.11b.

Ao instalar um AP, um administrador de rede designa ao ponto de acesso um **Identificador de Conjunto de Serviços** (*Service Set Identifier* — SSID) composto de uma ou duas palavras. (O comando ‘veja redes disponíveis’ no Microsoft Windows XP, por exemplo, apresenta uma lista que mostra o SSID de cada AP ordenado por faixa.) O administrador também deve designar um número de canal ao AP. Para entender números de canal, lembre-se de que as redes 802.11b operam na faixa de frequência de 2,4 GHz a 2,485 GHz. Dentro dessa faixa de 85 MHz, o padrão 802.11b define 11 canais que se sobrepõem parcialmente. Não há sobreposição entre quaisquer dois canais se, e somente se, eles estiverem separados por quatro ou mais canais. Em particular, o conjunto dos canais 1, 6 e 11 é o único conjunto de três canais não sobrepostos. Isso significa que um administrador poderia criar uma LAN sem fio com uma taxa máxima de transmissão agregada de 33 Mbps instalando três APs 802.11b na mesma localização física, designando os canais 1, 6 e 11 aos APs e interconectando cada um desses APs com um comutador.

Agora que já entendemos o básico sobre canais 802.11, vamos descrever uma situação interessante (e que não é completamente fora do comum) — uma **selva de Wi-Fis** (Wi-Fi jungle). Uma **selva de Wi-Fis** é qualquer localização física na qual uma estação sem fio recebe um sinal suficientemente forte de dois ou mais APs. Por exemplo, em muitos cafés da cidade de Nova York, uma estação sem fio pode captar um sinal de numerosos APs próximos. Um desses APs pode ser o AP gerenciado pelo café, enquanto os outros podem estar localizados em apartamentos vizinhos. Cada um desses pontos de acesso provavelmente estaria localizado em uma sub-rede diferente e o canal que utilizam teria sido designado independentemente.

Agora suponha que você entra nessa selva de Wi-Fis com seu computador portátil, em busca de acesso à Internet e de um cafezinho com biscoitos. Suponha que há cinco APs na selva. Para conseguir acesso à Internet, sua estação sem fio terá de se juntar a exatamente uma das sub-redes e, portanto, precisará se **associar** com exatamente um dos APs. Associar significa que a estação sem fio cria um fio virtual entre ela mesma e o AP.

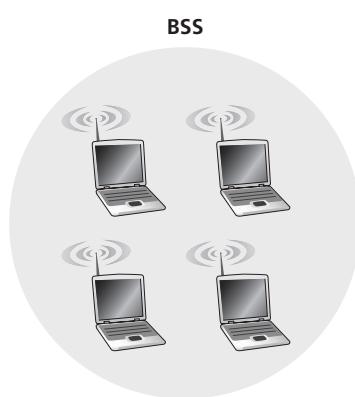


Figura 6.8 Uma rede ad hoc IEEE 802.11



Especificamente, somente o AP associado enviará quadros de dados (isto é, quadros contendo dados, tal como um datagrama) à sua estação sem fio e sua estação sem fio enviará quadros de dados à Internet somente por meio do AP associado. Mas como sua estação sem fio se associa com um determinado AP? E, o que é mais fundamental, como sua estação sem fio sabe quais APS estão dentro da selva, se é que há algum?

O padrão 802.11b requer que um AP envie periodicamente **quadros de sinalização**, cada qual incluindo o SSID e o endereço MAC do AP. Sua estação sem fio, sabendo que os APs estão enviando quadros de sinalização, faz uma varredura dos 11 canais em busca de quadros de sinalização de quaisquer APS que possam estar por lá (alguns dos quais podem estar transmitindo no mesmo canal — afinal, estamos na selva!). Ao tomar conhecimento dos APs disponíveis por meio dos quadros de sinalização, você (ou seu hospedeiro sem fio) seleciona um desses pontos de acesso para se associar.

O padrão 802.11 não especifica um algoritmo para selecionar com quais dos APs disponíveis se associar; esse algoritmo é de responsabilidade dos projetistas do firmware e do software 802.11 em seu hospedeiro sem fio. Normalmente, o hospedeiro escolhe o AP cujo quadro de sinalização é recebido com a força do sinal mais alta. Enquanto uma força alta do sinal for boa (veja, por exemplo, a Figura 6.3), a força do sinal não é a única característica do AP que determinará o desempenho que um hospedeiro recebe. Em particular, é possível que o AP selecionado possa ter um sinal forte, mas pode ser sobrecarregado com outros hospedeiros associados (que precisarão compartilhar a largura de banda sem fio naquele AP), enquanto um AP não carregado não é selecionado em razão a um sinal levemente mais fraco. Um número de maneiras alternativas para escolher os APs foram propostas recentemente [Vasudevan, 2005; Nicholson, 2006; Sudaresan, 2006]. Para uma discussão interessante e prática de como a força do sinal é medida, consulte [Bardwell, 2004].

O processo de varrer canais e ouvir quadros de sinalização é conhecido como varredura passiva (veja Figura 6.9a). Um hospedeiro sem fio pode também realizar uma varredura ativa, transmitindo um quadro de investigação que será recebido por todos os APs dentro de uma faixa do hospedeiro sem fio, como mostrado na Figura 6.9b. Os APs respondem ao quadro de requisição de investigação com um quadro de resposta de investigação. O hospedeiro sem fio pode, então, escolher o AP com o qual se associar dentre os APs de resposta.

Após selecionar o AP, o hospedeiro sem fio envia um quadro de solicitação de associação ao AP, e este responde com um quadro de resposta de associação. Observe que essa segunda apresentação de solicitação/resposta é necessária com a varredura ativa, visto que um AP de resposta ao quadro de solicitação de investigação inicial não sabe quais dos (possivelmente muitos) APs de resposta o hospedeiro escolherá para se associar, do mesmo modo que um cliente CHCP pode escolher entre servidores múltiplos DHCP (veja Figura 4.21). Uma vez associado ao AP, o hospedeiro desejará entrar na sub-rede (no sentido do endereçamento IP da Seção 4.4.2) à qual pertence o AP. Assim, o hospedeiro normalmente enviará uma mensagem de descoberta DHCP (veja Figura 4.21) à sub-rede através de um AP a fim de obter um endereço IP na sub-rede. Logo que o endereço é obtido, o restante da Internet, então, vê esse hospedeiro simplesmente como outro hospedeiro com um endereço IP naquela sub-rede.

Para criar uma associação com um determinado AP, a estação sem fio possivelmente terá de se autenticar perante o AP. LANs sem fio 802.11 dispõem de várias alternativas para autenticação e acesso. Uma abordagem, usada por muitas empresas, é permitir acesso a uma rede sem fio com base no endereço MAC de uma estação. Uma segunda abordagem, usada por muitos cafés Internet, emprega nomes de usuários e senhas. Em ambos os casos, o AP normalmente se comunica com um servidor de autenticação usando um protocolo como o RADIUS [RFC 2865] ou o DIAMETER [RFC 3588]. Separar o servidor de autenticação do AP permite que um servidor de autenticação atenda a muitos APs, centralizando as decisões de autenticação e acesso (quase sempre delicadas) em um único servidor e mantendo baixos os custos e a complexidade do AP. Veremos, na Seção 8.8.4, que o novo protocolo IEEE 802.11i, que define aspectos de segurança da família de protocolos 802.11, adota exatamente essa abordagem.

6.3.2 O protocolo MAC 802.11

Uma vez associada com um AP, uma estação sem fio pode começar a enviar e receber quadros de dados de e para o ponto de acesso. Porém, como várias estações podem querer transmitir quadros de dados ao mesmo

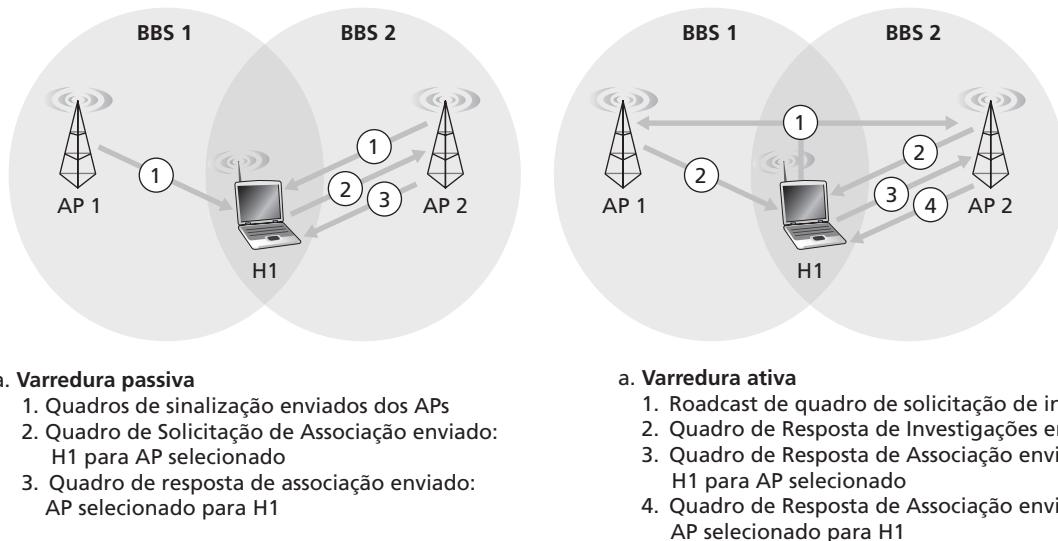


Figura 6.9 Varredura passiva e ativa para pontos de acesso

tempo sobre o mesmo canal, é preciso um protocolo de acesso múltiplo para coordenar as transmissões. Aqui, **estação** significa uma estação sem fio ou um AP. Como discutimos no Capítulo 5 e na Seção 6.2.1, em termos gerais há três classes de protocolos de acesso múltiplo: partição de canal (incluindo CDMA), acesso aleatório e revezamento. Inspirados pelo enorme sucesso da Ethernet e seu protocolo de acesso aleatório, os projetistas do 802.11 escolheram um protocolo de acesso aleatório para as LANs sem fio 802.11. Esse protocolo de acesso aleatório é denominado **CSMA com prevenção de colisão** ou, mais sucintamente, **CSMA/CA**. Do mesmo modo que o CSMA/CD da Ethernet, o ‘CSMA’ de CSMA/CA quer dizer ‘acesso múltiplo por detecção de portadora’, o que significa que cada estação sonda o canal antes de transmitir e abstém-se de transmitir quando percebe que o canal está ocupado. Embora tanto a Ethernet quanto o 802.11 usem acesso aleatório por detecção de portadora, os dois protocolos MAC apresentam diferenças importantes. Em primeiro lugar, em vez de usar detecção de colisão, o 802.11 usa técnicas de prevenção de colisão. Em segundo lugar, devido às taxas relativamente altas de erros de bits em canais sem fio, o 802.11 (diferentemente da Ethernet) usa um esquema de reconhecimento/retransmissão (ARQ) de camada de enlace. Mais adiante descreveremos os esquemas usados pelo 802.11 para prevenção de colisão e reconhecimento na camada de enlace.

Lembre-se de que nas seções 5.3 e 5.5 dissemos que, com o algoritmo de detecção de colisão, uma estação Ethernet ouve o canal à medida que transmite. Se, enquanto estiver transmitindo, a estação detectar que uma outra estação também está transmitindo, ela abortará sua transmissão e tentará transmitir novamente após uma pequena unidade de tempo aleatória. Diferentemente do protocolo Ethernet 802.3, o protocolo MAC 802.11 *não* implementa detecção de colisão. Isso se deve a duas razões importantes:

A capacidade de detectar colisões exige as capacidades de enviar (o próprio sinal da estação) e de receber (para determinar se uma outra estação está transmitindo) ao mesmo tempo. Como a potência do sinal recebido normalmente é muito pequena em comparação com a potência do sinal transmitido no adaptador 802.11, é caro construir um hardware que possa detectar colisões.

Mais importante, mesmo que o adaptador pudesse transmitir e ouvir ao mesmo tempo (e, presumivelmente, abortar transmissões quando percebesse um canal ocupado), ainda assim ele não seria capaz de detectar todas as colisões devido ao problema do terminal escondido e do desvanecimento, como discutimos na Seção 6.2.

Como LANs 802.11 sem fio não usam detecção de colisão, uma vez que uma estação comece a transmitir um quadro, *ela o transmite integralmente*; isto é, tão logo uma estação inicie, não há volta. Como é de esperar, transmitir quadros inteiros (particularmente os longos) quando existe grande possibilidade de colisão pode degradar

significativamente o desempenho de um protocolo de acesso múltiplo. Para reduzir a probabilidade de colisões, o 802.11 emprega diversas técnicas de prevenção de colisão, que discutiremos em breve.

Antes de considerar prevenção de colisão, contudo, em primeiro lugar devemos examinar o esquema de **reconhecimento na camada de enlace** do 802.11. Lembre-se de que dissemos, na Seção 6.2, que, quando uma estação em uma LAN sem fio envia um quadro, o quadro pode não chegar intacto à estação de destino por uma variedade de razões. Para lidar com essa probabilidade não desprezível de falha, o MAC 802.11 usa reconhecimentos de camada de enlace. Como ilustrado na Figura 6.8, quando a estação de destino recebe um quadro que passou na verificação de CRC, ela espera um curto período de tempo, conhecido como **Espaçamento Curto Interquadros** (*Short Inter-Frame Spacing — SIFS*), e então devolve um quadro de reconhecimento. Se a estação transmissora não receber um reconhecimento dentro de um dado período de tempo, ela admitirá que ocorreu um erro e retransmitirá o quadro usando novamente o protocolo CSMA/CA para acessar o canal. Se a estação transmissora não receber um reconhecimento após um certo número fixo de retransmissões, desistirá e descartará o quadro.

Agora que já discutimos como o 802.11 usa reconhecimentos de camada de enlace, estamos prontos para descrever o protocolo CSMA/CA 802.11. Suponha que uma estação qualquer (pode ser uma estação sem fio ou um AP) tenha um quadro para transmitir.

1. Se inicialmente a estação perceber que o canal está ocioso, ela transmitirá seu quadro após um curto período de tempo conhecido como **Espaçamento Interquadros Distribuído** (*Distributed Inter-Frame Space — DIFS*); ver Figura 6.10.
2. Caso contrário, a estação escolherá um valor aleatório de backoff e fará a contagem regressiva a partir desse valor quando perceber que o canal estiver ocioso. Se a estação perceber que o canal está ocupado, o valor do contador permanecerá estacionário.
3. Quando o contador chegar ao zero (note que isso pode ocorrer somente quando a estação percebe que o canal está ocioso), a estação transmitirá o quadro inteiro e então ficará esperando um reconhecimento.

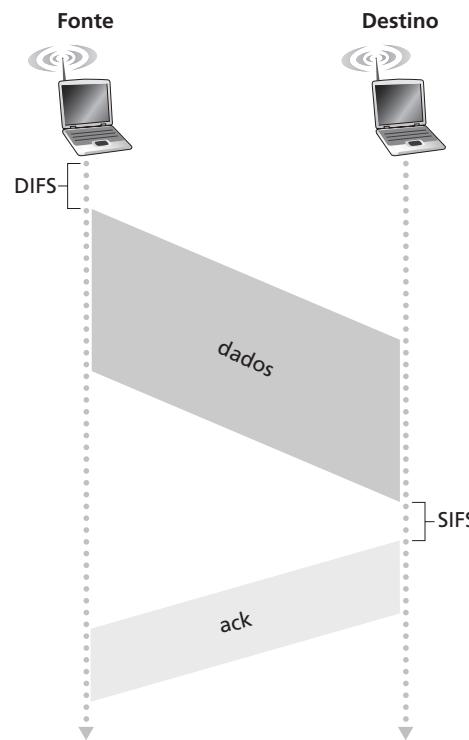


Figura 6.10 802.11 usa reconhecimentos de camada de enlace

4. Se receber um reconhecimento, a estação transmissora saberá que esse quadro foi corretamente recebido na estação de destino. Se a estação tiver outro quadro para transmitir, iniciará o protocolo CSMA/CA na etapa 2. Se não receber um reconhecimento, a estação transmissora entrará novamente na fase de backoff na etapa 2 e escolherá um valor aleatório dentro de um intervalo maior.

Lembre-se de que sob o protocolo de acesso múltiplo CSMA/CD (Seção 5.5.2), uma estação começa a transmitir tão logo percebe que o canal está ocioso. Com o CSMA/CA, entretanto, a estação priva-se de transmitir enquanto realiza a contagem regressiva, mesmo quando ela percebe que o canal está ocioso. Por que o CSMA/CD e o CDMA/CA adotam essas abordagens diferentes aqui?

Para responder a essa pergunta, vamos considerar um cenário com duas estações em que cada uma tem um quadro a transmitir, mas nenhuma transmite imediatamente porque percebe que uma terceira estação já está transmitindo. Com o CSMA/CD da Ethernet, cada uma das duas estações transmitiria tão logo detectasse que a terceira estação terminou de transmitir. Isso causaria uma colisão, o que não é uma questão séria em CSMA/CD, já que ambas as estações abortariam suas transmissões e assim evitariam a transmissão inútil do restante dos seus quadros. Entretanto, com 802.11 a situação é bem diferente. Como o 802.11 não detecta uma colisão nem aborta transmissão, um quadro que sofra uma colisão será transmitido integralmente. Assim, a meta do 802.11 é evitar colisões sempre que possível. Com esse protocolo, se duas estações perceberem que o canal está ocupado, ambas entrarão imediatamente em backoff aleatório e, esperamos, escolherão valores diferentes de backoff. Se esses valores forem, de fato, diferentes, assim que o canal ficar ocioso, uma das duas começará a transmitir antes da outra e (se as duas estações não estiverem ocultas uma da outra) a ‘estação perdedora’ ouvirá o sinal da ‘estação vencedora’, interromperá seu contador e não transmitirá até que a estação vencedora tenha concluído sua transmissão. Desse modo é evitada uma colisão dispendiosa. É claro que ainda podem ocorrer colisões com 802.11 nesse cenário: as duas estações podem estar ocultas uma da outra ou podem escolher valores de backoff aleatório próximos o bastante para que a transmissão da estação que se inicia primeiro tenha ainda de atingir a segunda estação. Lembre-se de que já vimos esse problema anteriormente em nossa discussão sobre algoritmos de acesso aleatório no contexto da Figura 5.14.

Terminais ocultos: RTS e CTS

O protocolo 802.11 MAC também inclui um esquema de reserva inteligente (mas opcional) que ajuda a evitar colisões mesmo na presença de terminais ocultos. Vamos estudar esse esquema no contexto da Figura 6.11, que mostra duas estações sem fio e um ponto de acesso. Ambas as estações sem fio estão dentro da faixa do AP (cuja área de cobertura é representada por um círculo sombreado) e ambas se associaram com o AP. Contudo, devido ao desvanecimento, as faixas de sinal de estações sem fio estão limitadas ao interior dos círculos sombreados mostrados na Figura 6.11. Assim, cada uma das estações sem fio está oculta da outra, embora nenhuma esteja oculta do AP.

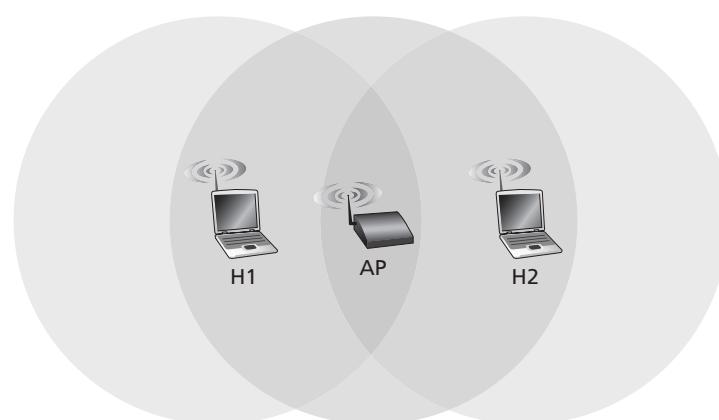


Figura 6.11 Exemplo de terminal oculto: H1 está oculto de H2, e vice-versa

Agora vamos considerar por que terminais ocultos podem ser problemáticos. Suponha que a estação H1 está transmitindo um quadro e, a meio caminho de sua transmissão, a estação H2 quer enviar um quadro para o AP. O H2, que não está ouvindo a transmissão de H1, primeiramente esperará um intervalo DIFS para, então, transmitir o quadro, resultando em uma colisão. Consequentemente, o canal será desperdiçado durante todo o período da transmissão de H1, bem como durante a transmissão de H2.

Para evitar esse problema, o protocolo IEEE 802.11 permite que uma estação utilize um quadro de controle **RTS** curto (*Request to Send* — solicitação de envio) e um quadro de controle **CTS** curto (*Clear to Send* — pronto para envio) para *reservar* acesso ao canal. Quando um remetente quer enviar um quadro DATA, ele pode enviar primeiramente um quadro RTS ao AP indicando o tempo total requerido para transmitir o quadro DATA e o quadro de reconhecimento (ACK). Quando o AP recebe o quadro RTS, responde fazendo a transmissão *broadcast* de um quadro CTS. Esse quadro CTS tem duas finalidades: dá ao remetente uma permissão explícita para enviar e também instrui as outras estações a não enviar durante o tempo reservado.

Assim, na Figura 6.12, antes de transmitir um quadro DATA, H1 primeiramente faz uma transmissão *broadcast* de um quadro RTS, que é ouvida por todas as estações que estiverem dentro do seu círculo de alcance, incluindo o AP. O AP então responde com um quadro CTS, que é ouvido por todas as estações dentro de sua faixa de alcance, incluindo H1 e H2. Como ouviu o CTS, a estação H2 deixa de transmitir durante o tempo especificado no quadro CTS. Os quadros RTS, CTS, DATA e ACK são mostrados na Figura 6.12.

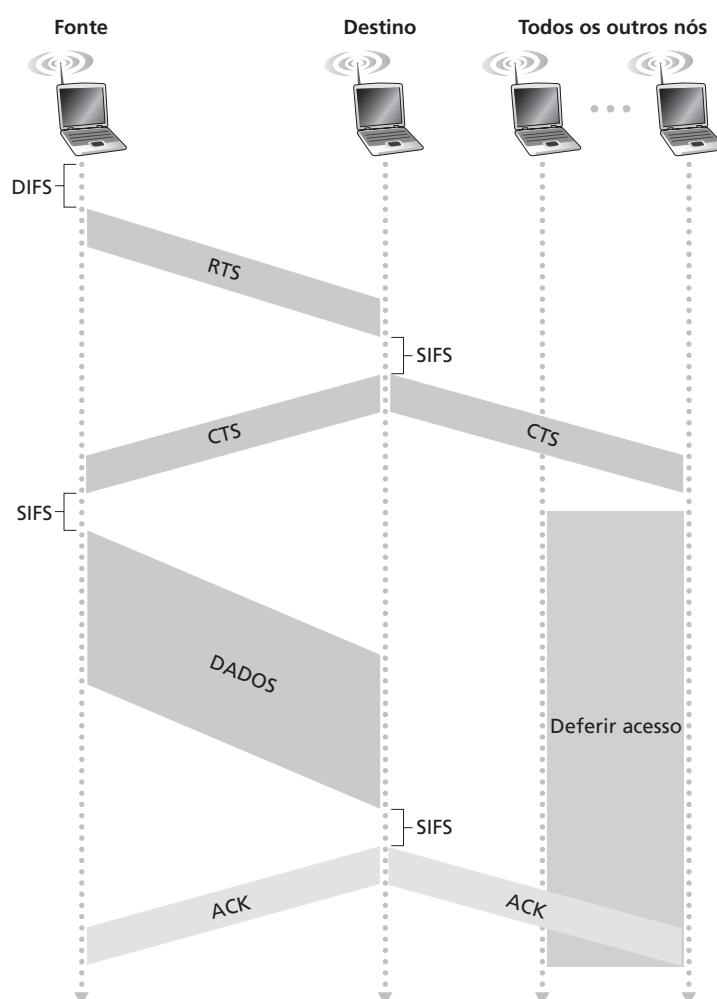


Figura 6.12 Prevenção de colisão usando os quadros RTS e CTS

A utilização dos quadros RTS e CTS pode melhorar o desempenho de dois modos importantes:

O problema da estação oculta é atenuado, visto que um quadro DATA longo é transmitido somente após o canal ter sido reservado.

Como os quadros RTS e CTS são curtos, uma colisão que envolva um quadro RTS ou CTS terá apenas a duração dos quadros RTS ou CTS curtos. Uma vez que os quadros RTS e CTS sejam corretamente transmitidos, os quadros DATA e ACK subsequentes deverão ser transmitidos sem colisões.

Aconselhamos o leitor a verificar o applet 802.11 no site Web deste livro. Esse applet interativo ilustra o protocolo CSMA/CA, incluindo a sequência de troca RTS/CTS.

Embora a troca RTS/CTS possa ajudar a reduzir colisões, também introduz atraso e consome recursos do canal. Por essa razão, a troca RTS/CTS é utilizada (quando utilizada) apenas para reservar o canal para a transmissão de um quadro DATA longo. Na prática, cada estação sem fio pode estabelecer um patamar RTS tal que a sequência RTS/CTS seja utilizada somente quando o quadro for mais longo do que o patamar. Para muitas estações sem fio, o valor default do patamar RTS é maior do que o comprimento máximo do quadro, de modo que a sequência RTS/CTS é omitida para todos os quadros DATA enviados.

Usando o 802.11 como enlace ponto-a-ponto

Até aqui nossa discussão focalizou a utilização do 802.11 em um cenário de múltiplo acesso. Devemos mencionar que, se dois nós tiverem, cada um, uma antena direcional, eles poderão dirigir suas antenas um para o outro e executar o protocolo 802.11 sobre o que é, essencialmente, um enlace ponto-a-ponto. Dado o baixo custo comercial do hardware 802.11, a utilização de antenas direcionais e uma maior potência de transmissão permitem que o 802.11 seja utilizado como um meio barato de prover conexões sem fio ponto-a-ponto por dezenas de quilômetros. [Raman, 2007] descreve uma dessas redes sem fio multissaltos que funciona nas planícies rurais do rio Ganges, na Índia, e que contém enlaces 802.11 ponto-a-ponto.

6.3.3 O quadro IEEE 802.11

Embora o quadro 802.11 tenha muitas semelhanças com um quadro Ethernet, ele também contém vários campos que são específicos para sua utilização para enlaces sem fio. O quadro 802.11 é mostrado na Figura 6.13. Os números acima de cada um dos campos no quadro representam os comprimentos dos campos em bytes; os números acima de cada um dos subcampos no campo de controle do quadro representam os comprimentos dos subcampos em bits. Agora vamos examinar os campos no quadro, bem como alguns dos subcampos mais importantes no campo de controle do quadro.

Campos de carga útil e de CRC

No coração do quadro está a carga útil, que consiste, tipicamente, em um datagrama IP ou em um pacote ARP. Embora o comprimento permitido do campo seja 2.312 bytes, normalmente ele é menor do que 1.500 bytes,

Quadro (os números indicam o comprimento do campo em bytes):

2	2	6	6	6	2	6	0-2312	2
Controle de quadro	Duração	Endereço 1	Endereço 2	Endereço 3	Controle de sequência	Endereço 4	Carga útil	CRC

Detalhamento do campo de controle do quadro (os números indicam o comprimento do campo em bits):

2	2	4	1	1	1	1	1	1	1	1	1
Versão do protocolo	Tipo	Subtipo	Para o AP	Do AP	Mais fragmentos	Nova tentativa	Gerenciamento de energia	Mais dados	WEP	Rsvd	

Figura 6.13 O quadro 802.11

contendo um datagrama IP ou um pacote ARP. Como um quadro Ethernet, um quadro 802.11 inclui uma verificação de redundância cíclica (CRC), de modo que o receptor possa detectar erros de bits no quadro recebido. Como já vimos, erros de bits são muito mais comuns em LANs sem fio do que em LANs cabeadas, portanto, aqui, CRC é ainda mais útil.

Campos de endereço

Talvez a diferença mais marcante no quadro 802.11 é que ele tem *quatro* campos de endereço e cada um pode conter um endereço MAC de 6 bytes. Mas por que quatro campos de endereço? Um campo de fonte MAC e um campo de destino MAC não são suficientes como são na Ethernet? Acontece que aqueles três campos de endereço são necessários para finalidades de interconexão em rede — especificamente, para mover o datagrama de camada de enlace de uma estação sem fio, passando por um AP, até uma interface de roteador. O quarto campo de endereço é usado quando APs encaminham quadros uns aos outros em modo ad hoc. Visto que estamos considerando apenas redes de infraestrutura, vamos concentrar nossa atenção nos três primeiros campos de endereço. O padrão 802.11 define esses campos como se segue:

- Endereço 2 é o endereço MAC da estação que transmite o quadro. Assim, se uma estação sem fio transmitir o quadro, o endereço MAC daquela estação será inserido no campo de endereço 2. De modo semelhante, se um AP transmitir o quadro, o endereço MAC do AP será inserido no campo de endereço 2.
- Endereço 1 é o endereço MAC da estação sem fio que deve receber o quadro. Assim, se uma estação móvel sem fio transmitir o quadro, o endereço 1 conterá o endereço MAC do AP de destino. De modo semelhante, se um AP transmitir o quadro, o endereço 1 conterá o endereço MAC da estação sem fio destinatária.
- Para entender o endereço 3, lembre-se de que o BSS (que consiste no AP e estações sem fio) é parte de uma sub-rede, e que essa sub-rede se conecta com outras sub-redes, via alguma interface de roteador.

Para compreender melhor a finalidade do endereço 3, vamos examinar um exemplo de interconexão em rede no contexto da Figura 6.14. Nessa figura há dois APs, cada um responsável por um certo número de estações sem fio. Cada um dos APs tem uma conexão direta com um roteador, que, por sua vez, se conecta com a Internet global. Devemos ter sempre em mente que um AP é um dispositivo de camada de enlace e, portanto, não ‘fala’ IP nem entende endereços IP. Agora, considere mover um datagrama da interface de roteador R1 até a estação sem fio H1. O roteador não está ciente de que há um AP entre ele e H1; da perspectiva do roteador, H1 é apenas um hospedeiro em uma das sub-redes às quais ele (o roteador) está conectado.

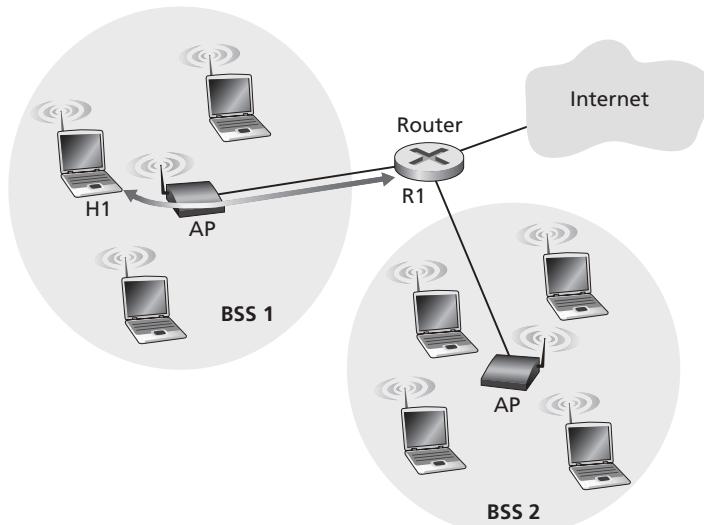


Figura 6.14 A utilização de campos de endereço em quadros 802.11: movendo um quadro entre H1 e R1

O roteador, que conhece o endereço IP de H1 (pelo endereço de destino do datagrama), utiliza ARP para determinar o endereço MAC de H1, exatamente como aconteceria em uma LAN Ethernet comum. Após obter o endereço MAC de H1, a interface de roteador R1 encapsula o datagrama em um quadro Ethernet. O campo de endereço de fonte desse quadro contém o endereço MAC de R1 e o campo de endereço de destino contém o endereço MAC de H1.

Quando o quadro Ethernet chega ao AP, este converte o quadro Ethernet 802.3 para um quadro 802.11 antes de transmiti-lo para o canal sem fio. O AP preenche o endereço 1 e o endereço 2 com o endereço MAC de H1 e seu próprio endereço MAC, respectivamente, como descrito acima. Para o endereço 3, o AP insere o endereço MAC de R1. Dessa maneira, H1 pode determinar (a partir do endereço 3) o endereço MAC da interface de roteador que enviou o datagrama para a sub-rede.

Agora considere o que acontece quando a estação sem fio H1 responde movendo um datagrama de H1 para R1.

H1 cria um quadro 802.11, preenchendo os campos de endereço 1 e 2 com o endereço MAC do AP e com o endereço MAC de H1, respectivamente, como descrito acima. Para o endereço 3, H1 insere o endereço MAC de R1.

Quando recebe o quadro 802.11, o AP o converte para um quadro Ethernet. O campo de endereço de fonte para esse quadro é o endereço MAC de H1 e o campo de endereço de destino é o endereço MAC de R1. Assim, o endereço 3 permite que o AP determine o endereço de destino MAC apropriado ao construir o quadro Ethernet.

Em resumo, o endereço 3 desempenha um papel crucial na interconexão do BSS com uma LAN cabeada.

Campos de número de sequência, duração e controle de quadro

Lembre-se de que, em 802.11, sempre que uma estação recebe corretamente um quadro de uma outra estação, devolve um reconhecimento. Como reconhecimentos podem ser perdidos, a estação emissora pode enviar várias cópias de um dado quadro. Como vimos na nossa discussão sobre o protocolo rdt2.1 (Seção 3.4.1), a utilização de números de sequência permite que o receptor distinga entre um quadro recém-transmitido e a retransmissão de um quadro anterior. Assim, o campo de número de sequência no quadro 802.11 cumpre aqui, na camada de enlace, exatamente a mesma finalidade que cumpria na camada de transporte do Capítulo 3.

Não se esqueça de que o protocolo 802.11 permite que uma estação transmissora reserve o canal durante um período que inclui o tempo para transmitir seu quadro de dados e o tempo para transmitir um reconhecimento. Esse valor de duração é incluído no campo de duração do quadro (tanto para quadros de dados quanto para os quadros RTS e CTS).

Como mostrado na Figura 6.13, o campo de controle de quadro inclui muitos subcampos. Diremos apenas umas poucas palavras sobre alguns dos subcampos mais importantes; se o leitor quiser uma discussão mais completa, aconselhamos que consulte a especificação 802.11 [Held, 2001; Crow, 1997; IEEE 802.11, 1999]. Os campos *tipo* e *subtipo* são usados para distinguir os quadros de associação, RTS, CTS, ACK e de dados. Os campos *de* e *para* são usados para definir os significados dos diferentes campos de endereço. (Esses significados mudam dependendo da utilização dos modos ad hoc ou de infraestrutura e, no caso do modo de infraestrutura, mudam dependendo de o emissor do quadro ser uma estação sem fio ou um AP.) Finalmente, o campo WEP (*wireless equivalent privacy*) indica se está sendo ou não utilizada criptografia. (A WEP é discutida no Capítulo 8.)

6.3.4 Mobilidade na mesma sub-rede IP

Para ampliar a faixa física de uma LAN sem fio, empresas e universidades frequentemente distribuirão vários BSSs dentro da mesma sub-rede IP. Isso naturalmente levanta a questão da mobilidade entre os BSSs — como estações sem fio passam ininterrupta e imperceptivelmente de um BSS para outro enquanto mantêm sessões TCP em curso? Como veremos nesta subseção, a mobilidade pode ser manipulada de uma maneira relativamente direta quando os BSSs são parte de uma sub-rede. Quando estações se movimentam entre sub-redes, são necessários protocolos de gerenciamento de mobilidade mais sofisticados, tais como os que estudaremos nas seções 6.5 e 6.6.

Agora vamos examinar um exemplo específico de mobilidade entre BSSs na mesma sub-rede. A Figura 6.15 mostra dois BSSs interconectados e um hospedeiro, H1, que se move entre BSS1 e BSS2. Como nesse exemplo o dispositivo de interconexão entre os dois BSSs não é um roteador, todas as estações nos dois BSSs, incluindo os APs, pertencem à mesma sub-rede IP. Assim, quando H1 se move de BSS1 para BSS2, ele pode manter seu endereço IP e todas as suas conexões TCP em curso. Se o dispositivo de interconexão fosse um roteador, então H1 teria de obter um novo endereço IP na sub-rede na qual estava percorrendo. Essa mudança de endereço interromperia (e, consequentemente, finalizaria) qualquer conexão TCP em curso em H1. Na Seção 6.6, veremos como um protocolo de mobilidade da camada de rede, como o IP móvel, pode ser usado para evitar esse problema.

Mas o que acontece especificamente quando H1 passa de BSS1 para BSS2? À medida que se afasta de AP1, o H1 detecta um enfraquecimento do sinal de AP1 e começa a fazer uma varredura em busca de um sinal mais forte. H1 recebe quadros de sinalização de AP2 (que em muitos ambientes empresariais e universitários terão o mesmo SSID do AP1). Então, H1 se desassocia de AP1 e se associa com AP2, mantendo, ao mesmo tempo, seu endereço IP e suas sessões TCP em curso.

Isso abrange o problema de transferência do ponto de vista do hospedeiro e do AP. Mas e quanto ao comutador na Figura 6.15? Como é possível saber que o hospedeiro se locomoveu de um hospedeiro a outro?

O leitor talvez se lembre de que, no Capítulo 5, dissemos que comutadores observam os endereços e constroem automaticamente suas tabelas de repasse. Essa característica de auto-aprendizagem funciona bem para movimentações ocasionais (por exemplo, quando um profissional é transferido de um departamento para outro); contudo, comutadores não são projetados para suportar usuários com alto grau de mobilidade, que querem manter conexões TCP enquanto se movimentam entre BSSs. Para avaliar este problema aqui, lembre-se de que, antes da movimentação, o comutador tem um registro em sua tabela de repasse que vincula o endereço MAC de H1 com a interface de saída do comutador por meio da qual o H1 pode ser alcançado. Se H1 estiver inicialmente em BSS1, então um datagrama destinado a H1 será direcionado a ele via AP1. Contudo, tão logo H1 se associe com BSS2, seus quadros deverão ser direcionados para AP2. Uma solução (na verdade um tanto forçada) é o AP2 enviar ao comutador um quadro Ethernet broadcast com o endereço de fonte de H1 logo após a nova associação. Quando o comutador recebe o quadro, atualiza sua tabela de repasse, permitindo que H1 seja alcançado via AP2. O grupo de padrões 802.11 está desenvolvendo um protocolo inter-AP para manipular essas e outras questões associadas.

6.3.5 Recursos avançados em 802.11

Finalizaremos nossa abordagem sobre 802.11 com uma breve discussão sobre capacidades avançadas encontradas nas redes 802.11. Como veremos, essas capacidades não são completamente especificadas no padrão 802.11, mas, em vez disso, são habilitadas por mecanismos especificados no padrão. Isso permite que fornecedores diferentes implementem essas capacidades usando suas próprias abordagens, aparentemente trazendo vantagens sobre a concorrência.

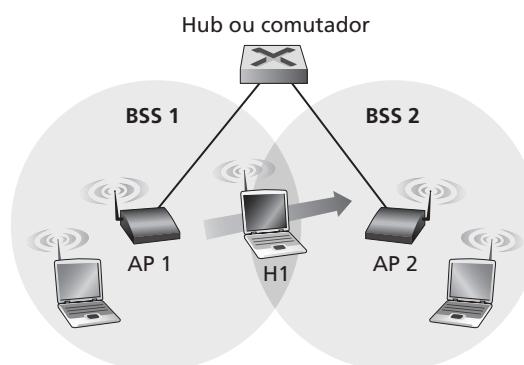


Figura 6.15 Mobilidade na mesma sub-rede

Adaptação da taxa 802.11

Vimos anteriormente na Figura 6.3 que as diferentes técnicas de modulação (com as diferentes taxas de transmissão que eles fornecem) são adequadas para cenários SNR diferentes. Considere, por exemplo, um usuário 802.11 móvel que está, inicialmente, a 20 metros de distância da estação-base, com uma alta razão sinal-ruído. Dado o alto SNR, o usuário pode se comunicar com essa estação usando uma técnica de modulação da camada física que provê altas taxas de transmissão enquanto mantém uma BER baixa. Esse usuário é um felizardo! Suponha agora que o usuário se torne móvel, se distanciando da estação-base, sendo que o SNR diminui à medida que a distância da estação-base aumenta. Neste caso, se a técnica de modulação usada no protocolo 802.11 que está operando entre a estação-base e o usuário não mudar, a BER será inaceitavelmente alta à medida que o SNR diminui e, consequentemente, nenhum quadro transmitido será recebido corretamente.

Por esta razão, algumas implementações de 802.11 possuem uma capacidade de adaptação da taxa que seleciona, de maneira adaptável, a técnica de modulação da camada física sobreposta a ser usada baseado em características de canal recentes ou atuais. A implementação do WaveLAN-II 802.11b da Lucent [Kamerman, 1997] provê múltiplas taxas de transmissão de dados. Se um nó enviar dois quadros seguidos sem receber um reconhecimento (uma indicação implícita de erros de bit no canal), a taxa de transmissão cai para a próxima taxa mais baixa. Se dez quadros seguidos forem reconhecidos, ou se um temporizador que registra o tempo desde que o último fallback expirar, a taxa de transmissão aumenta para a próxima taxa mais alta. Esse mecanismo de adaptação da taxa compartilha a mesma filosofia de “investigação” (refletido por recebimentos ACK); quando algo “ruim” acontece, a taxa de transmissão é reduzida. A adaptação da taxa 802.11 e o controle de congestionamento TCP são, deste modo, semelhantes à criança está sempre exigindo mais e mais de seus pais (digamos que doce, para uma criança e chegar mais tarde em casa, para os adolescentes) até eles finalmente dizerem “Chega!” e a criança desistir (para tentar novamente após a situação melhorar!). Um número de métodos também foi proposto para aperfeiçoar esse esquema básico de ajuste automático de taxa encontrado em [Holland, 2001; Lacage, 2004].

Gerenciamento de potência

A potência é uma fonte preciosa em aparelhos móveis e, assim, o padrão 802.11 provê capacidades de gerenciamento de potência, permitindo que os nós 802.11 minimizem o tempo de suas funções de percepção, transmissão e recebimento, e outros circuitos precisam estar “em funcionamento”. O gerenciamento de potência 802.11 opera da seguinte maneira. Um nó é capaz de alternar claramente entre os estados de dormência e despertar (como um aluno em sala de aula!). Um nó indica ao ponto de acesso que entrará no modo de dormência ajustando o bit de gerenciamento de potência no cabeçalho de um quadro 802.11 para 1. Um temporizador localizado no nó é, então, ajustado para acordar o nó antes de um AP ser programado para enviar seus quadros de sinalização (lembre-se de que, normalmente, um AP envia um quadro de sinalização a cada 100 milissegundos). Uma vez que o AP descobre, através do bit de transmissão de potência, que o nó vai dormir, ele (o AP) sabe que não deve enviar nenhum quadro àquele nó, e armazenará qualquer quadro destinado ao hospedeiro que está dormindo para transmissão posterior.

Um nó acordará logo após o AP enviar um quadro de sinalização, e rapidamente entrará no modo ativo (diferentemente do estudante sonolento, esse despertar leva apenas 250 microssegundos [Kamerman, 1997]!). Os quadros de sinalização enviados pelo AP contêm uma relação de nós cujos quadros foram armazenados no AP. Se não existirem quadros armazenados para o nó, ele pode voltar a dormir. Caso contrário, o nó pode, claramente, solicitar o envio dos quadros armazenados enviando uma mensagem de polling ao AP. Com um tempo de inter-sinalização de 100 milissegundos, um despertar de 250 milissegundos e um tempo semelhantemente pequeno para receber um quadro de sinalização e verificar que não haja quadros armazenados, um nó que não possui quadros para enviar ou receber pode dormir 99% do tempo, resultando em uma economia de energia significativa.

6.3.6 Mais além de 802.11: Bluetooth e WiMAX

Como ilustrado na Figura 6.2, o padrão Wi-Fi IEEE 802.11 é voltado para a comunicação entre aparelhos diferenciados de até 100 metros (exceto quando 802.11 é usado em configuração ponto a ponto com antena dire-

cional). Dois outros protocolos IEEE 802 — o Bluetooth (definido no padrão IEEE 802.15.1 [IEEE 802.15.2009]) e o WiMAX (definido no padrão IEEE 802.16 [IEEE 802.16d 2004; IEEE 802.16e 2005]) — são padrões para se comunicar a distâncias mais curtas e mais longas, respectivamente.

Bluetooth

Um rede IEEE 802.15.1 opera sobre uma curta faixa, a baixa potência, e a um custo baixo. Esta é, basicamente, uma tecnologia de “substituição de cabos” de baixa taxa, faixa curta e baixa potência para interconectar laptops, aparelhos periféricos, telefones celulares e PDAs, enquanto 802.11 é uma tecnologia de “acesso” de alta taxa, faixa média e alta potência. Por esta razão, as redes 802.15.1, às vezes, são denominadas rede pessoal sem fio (WPAN). As camadas de física e de enlace de 802.15.1 são baseadas na especificação do Bluetooth anterior para redes pessoais [Held, 2001; Bisdikian, 2001]. Redes 802.15.1 operam em faixa de rádio não licenciada de 2,4 GHz em modo TDM, com intervalos de tempo (time slots) de 625 microsegundos. Durante cada intervalo de tempo, um emissor transmite por um entre 79 canais, sendo que, de intervalo para intervalo, o canal muda de uma maneira conhecida, porém pseudoaleatória. Essa forma de saltar de canal em canal (hopping), conhecida como **frequency-hopping spread spectrum** (FHSS), espalha transmissões sobre o espectro de frequência ao longo do tempo. O 802.15.1 pode prover velocidades de dados de até 4 Mbps.

Redes 802.15.1 são redes ad hoc: não é preciso nenhuma infraestrutura de rede (por exemplo, um ponto de acesso) para interconectar dispositivos 802.15.1. Assim, esses dispositivos devem se organizar por si sós. Dispositivos 802.15.1 são primeiramente organizados em uma **picorrede** (piconet: pequena rede) de até oito dispositivos ativos, como mostra a Figura 6.16. Um desses dispositivos é designado como o mestre, e os outros agem como escravos. Na verdade, o nó mestre comanda a picorrede como um rei — seu relógio determina o tempo na picorrede, ele pode transmitir em cada intervalo de tempo de número ímpar e um escravo pode transmitir somente após o mestre ter se comunicado com ele no intervalo de tempo anterior e, mesmo assim, o escravo pode transmitir apenas para o mestre. Além dos dispositivos escravos, a rede também pode conter até 255 dispositivos estacionados. Esses dispositivos não podem se comunicar até que o nó mestre tenha mudado seus estados para ativos.

Se o leitor interessado quiser mais informações sobre WPANs 802.15.1, pode consultar as referências do Bluetooth [Held, 2001; Bisdikian, 2001] ou o site Web oficial do IEEE 802.15 [IEEE 802.15, 2009].

WiMAX

O WiMAX (Interoperabilidade Mundial para Acesso em Micro-ondas) pertence à família dos padrões IEEE 802.16 que visa a entregar dados sem fio a um grande número de usuários sobre uma ampla área a taxas que competem com as redes ADSL e modem a cabo. O padrão 802.16d é uma atualização do padrão 802.16a. O padrão 802.16e tem o objetivo de suportar a mobilidade a velocidades de 70-80 milhas por hora (por exemplo,

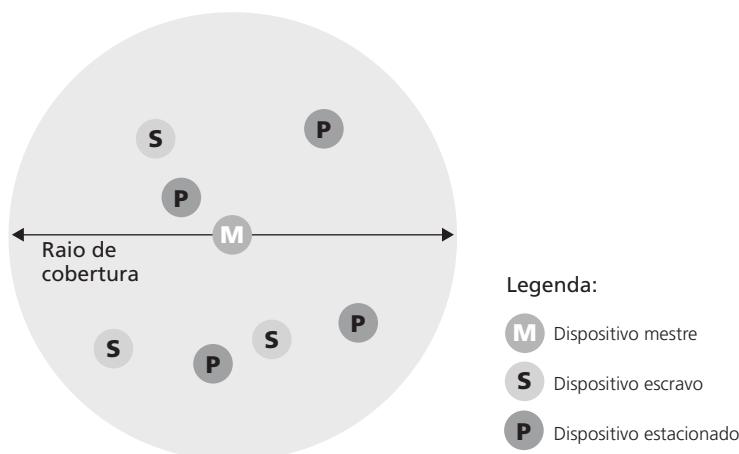


Figura 6.16 Uma picorrede Bluetooth

velocidades de rodovia na maioria dos países fora da Europa) e possui uma estrutura de enlace diferente para aparelhos pequenos e de recursos limitados, como PDAs, celulares e laptops.

A arquitetura 802.16 é baseada na noção de uma estação-base que serve uma grande quantidade de clientes (conhecida como estações de assinante) associados a essa estação-base. Neste sentido, o WiMAX se parece tanto com a infraestrutura da tecnologia Wi-Fi como com as redes de telefonia celular. A estação-base coordena a transmissão dos pacotes da camada de enlace nas direções de transmissão (downstream) (da estação-base para as estações de assinante) e recebimento (upstream) (das estações de assinante para a estação-base) de acordo com a estrutura do quadro do TDM mostrado na Figura 6.17. Usaremos aqui o termo “pacote” em vez do termo “quadro” (que usamos para 802.11 e para pacotes da camada de enlace) para diferenciar a unidade de dados na camada de enlace da estrutura do quadro do TDM mostrada na Figura 6.17. Desta maneira, o WiMAX opera em multiplexação de divisão do tempo (TDM), embora o tempo de framing seja variável, como discutido abaixo. Observamos que o WiMAX define também um modo de operação FDM, mas esse assunto não será tratado aqui.

No início do quadro, a estação-base primeiramente envia uma relação de mensagens MAP (protocolo de acesso de mídia) de transmissão informando as estações de assinante das propriedades da camada física (esquema de modulação, codificação e parâmetros para correção de erro) que serão utilizadas para transmitir rajadas de pacotes subsequentes dentro do quadro. Pode haver múltiplas rajadas dentro de um quadro, e múltiplos pacotes dentro de uma rajada destinados a uma determinada estação de assinante. Todos os pacotes dentro da rajada são transmitidos pela estação-base usando as mesmas propriedades da camada física. Entretanto, essas propriedades podem mudar de uma rajada para outra, permitindo que a estação-base escolha métodos de transmissão da camada física que não são adequados para a estação de assinante. A estação-base pode escolher o conjunto de receptores para os quais enviará durante esse quadro de acordo com as condições estimadas do canal atual para cada receptor. Essa forma de programação oportunista [Bender, 2000, Kulkarni, 2005] — combinando o protocolo da camada física às condições do canal entre o emissor e o receptor, e escolhendo os receptores para os quais os pacotes serão enviados de acordo com as condições do canal — permite que a estação-base aproveite ao máximo o meio sem fio. O padrão WiMAX não exige um conjunto particular de parâmetros da camada física que deve ser usado em determinada situação. Essa decisão é de responsabilidade do fornecedor do equipamento WiMAX e do operador de rede.

A estação-base do WiMAX também ajusta o acesso da estação de assinante para o canal de recebimento através do uso de mensagens UL-MAP. Essas mensagens controlam o tempo em que cada estação de assinante recebe acesso ao canal nos subquadros de uplink subsequentes. Novamente, o padrão WiMAX não exige nenhuma política específica para alocar o tempo do canal de uplink para um cliente — essa decisão é de responsabilidade do operador de rede. Em vez disso, o WiMAX provê os mecanismos (como mensagens de controle UL-MAP) para empregar uma política que poderia oferecer diferentes quantidades de acesso do tempo do canal a diferentes estações de assinante. As porções iniciais do subquadro de uplink são usadas para os assinantes transmitirem mensagens de controle de enlace de rádio, mensagens para solicitar admissão e autenticação na rede WiMAX, e mensagens protocolo de gerenciamento de alto nível, como o DHCP e o SNMP.

A Figura 6.18 mostra o formato para o pacote WiMAX MAC. O único campo que observamos aqui é do identificador de conexão no cabeçalho. O WiMAX é uma arquitetura orientada a conexão que permite que cada conexão tenha uma qualidade de serviço (QoS) associada, parâmetros de tráfego e outras informações. Como essa QoS será fornecida depende do operador de rede. O WiMAX provê mecanismos de baixo nível (por exemplo,

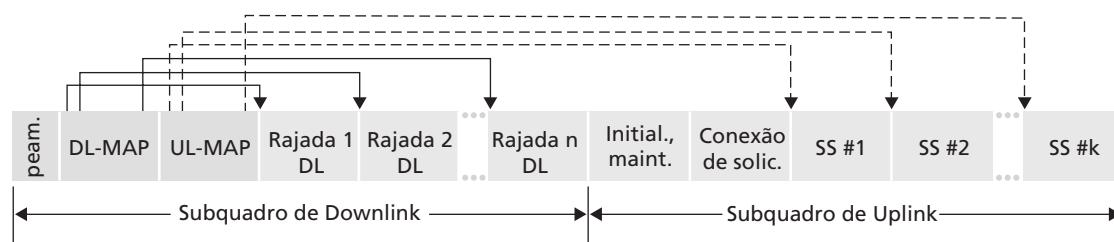


Figura 6.17 Estrutura de quadro 802.16 TDM

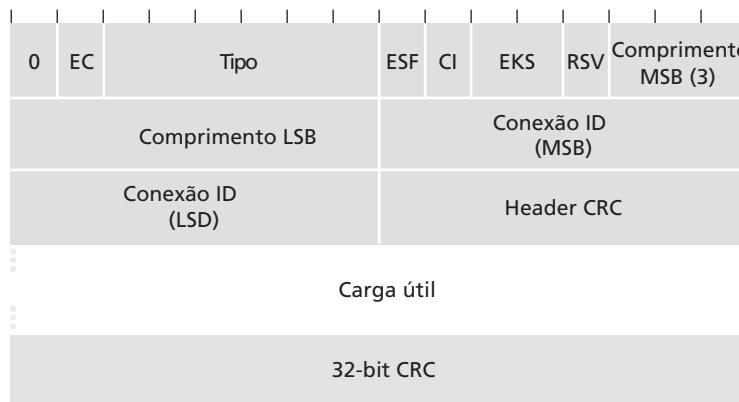


Figura 6.18 Pacote 802.16

avaliação de canal e campos de solicitação de admissão de conexão para transportar informações entre a estação-base e o hospedeiro), mas não oferece uma abordagem geral nem políticas para fornecer a QoS. Mesmo que cada estação de assinante normalmente tenha um endereço MAC de 48 bits (como nas redes 802.3 e 802.11), esse endereço MAC é mais bem visto como um identificador de equipamento em WiMAX, visto que a comunicação entre pontos de chegada é mapeada para um identificador de conexão (em vez de os endereços dos terminais emissores e receptores da conexão).

Nosso tratamento sobre o WiMAX foi necessariamente breve, e ainda há muitos assuntos, como gerenciamento de potência (um estado de dormência semelhante àquele em 802.11), transferências (handoffs), programação dependente do estado do canal de transmissões MAC PDU da estação-base, suporte QoS e segurança, que não pudemos tratar aqui. Com o padrão 802.16e ainda em desenvolvimento, os sistemas WiMAX continuarão a se desenvolver nos próximos anos. Enquanto esses padrões relacionados acima resultam em uma leitura “seca” desses e outros assuntos sobre o WiMAX, [Eklund, 2002; Cicconetti, 2006] apresentam pontos de vista claros sobre o WiMAX.

6.4 Acesso celular à Internet

Na seção anterior vimos como um hospedeiro da Internet pode acessar a Internet quando estiver dentro de um hotspot Wi-Fi, isto é, quando estiver nas vizinhanças de um ponto de acesso 802.11. Mas a área de cobertura da maioria dos hotspots Wi-Fi é pequena, entre 10 e 100 metros de diâmetro. O que fazer, então, quando precisamos desesperadamente de um acesso sem fio à Internet e não podemos acessar nenhum hotspot Wi-Fi?

Dado que a telefonia celular agora está presente por toda a parte, em muitas áreas no mundo inteiro, uma estratégia natural é estender redes celulares de modo que suportem não somente telefonia de voz, mas também acesso sem fio à Internet. Idealmente, esse acesso à Internet teria uma velocidade razoavelmente alta e proveria mobilidade imperceptível e ininterrupta, permitindo que usuários mantivessem suas sessões TCP enquanto em trânsito, por exemplo, em um ônibus ou em um trem. Com taxas de bits suficientemente altas entre a fonte e o usuário e vice-versa, o usuário poderia até mesmo manter sessões de videoconferência enquanto em trânsito. Esse cenário não é assim tão implausível. Quando escrevemos este livro (início de 2009), muitas provedoras de telefonia celular ofereciam a seus assinantes um serviço de acesso celular à Internet por menos de 50 dólares mensais com taxas de bits entre a provedora e o usuário e vice-versa na faixa de algumas centenas de kilobits por segundo. Taxas de dados de vários megabits por segundo estão se tornando acessíveis à medida que os serviços de dados de banda larga, como o HSDPA, são cada vez mais empregados.

Nesta seção, damos uma breve visão geral das tecnologias de acesso celular à Internet já existentes e emergentes. Mais uma vez, aqui focalizaremos o primeiro salto sem fio entre o telefone celular e a infraestrutura de

rede de telefonia cabeada; na Seção 6.7 consideraremos como chamadas são roteadas para um usuário que está se movimentando entre estações-base. Nossa breve discussão proverá, necessariamente, somente uma descrição simplificada e de alto nível de tecnologias celulares. É claro que a questão das comunicações celulares modernas é de grande amplitude e profundidade, e muitas universidades oferecem diversos cursos sobre o assunto. O leitor que desejar um conhecimento mais profundo da questão deve consultar [Goodman, 1997; Scourias, 1997; Kaaranen, 2001; Lin, 2001; Korhonen, 2003; Schiller, 2003; Scourias, 2007; Turner, 2009], bem como as referências particularmente excelentes e exaustivas em [Mouly, 1992].

6.4.1 Uma visão geral da arquitetura celular

Em nossa discussão sobre a arquitetura da rede celular nesta seção, adotaremos a terminologia dos padrões do Sistema Global para Comunicações Móveis (GSM). (Para os amantes de história, a sigla GSM, originalmente, derivou de Groupe Spécial Mobile, até a forma anglicizada ser adotada, mantendo as letras iniciais originais.) Nos anos 1980, os europeus reconheceram a necessidade de um sistema pan-europeu de telefonia celular digital que substituiria os diversos sistemas analógicos de telefonia celular incompatíveis, levando ao padrão GSM [Mouly, 1992]. Os europeus implantaram a tecnologia GSM com um grande sucesso no início dos anos 1990 e, desde então, o GSM cresceu e se tornou um gigante do mundo da telefonia celular, com mais de 80% de assinantes no mundo utilizando essa tecnologia.

Quando as pessoas falam sobre tecnologia celular, geralmente a classificam como pertencendo a uma das diversas “gerações”. As gerações mais antigas foram, basicamente, projetadas para o tráfego de voz. Os sistemas de primeira geração (1G) eram sistemas FDMA analógicos desenvolvidos especialmente para a comunicação somente por voz. Esses sistemas 1G quase não existem mais, tendo sido substituídos pelos sistemas 2G. Os sistemas originais 2G também foram projetados para voz, sendo estendidos, mais tarde, para suportar dados (por exemplo, a Internet), bem como o serviço de voz. Os sistemas 3G, que estão sendo implementados atualmente, também suportam voz e dados, mas com uma ênfase crescente em capacidades de dados e enlaces de acesso via rádio com velocidade maior.

Arquitetura de rede celular, 2G: conexões por voz à rede telefônica

O termo *celular* refere-se ao fato de que uma área geográfica é dividida em várias áreas de cobertura geográfica, conhecidas como **células**, ilustrado como hexágonos à esquerda da Figura 6.19. Assim como estudamos o padrão Wi-Fi 802.11 na Seção 6.3.1, o GSM possui sua nomenclatura específica. Cada célula contém uma estação-base de transceptor (BTS) que transmite sinais para, e recebe sinais de, estações móveis dentro de sua célula. A área de cobertura de uma célula depende de muitos fatores, incluindo potência da BTS, potência de transmissão de aparelhos do usuário, da estação móvel, obstáculos, como edifícios, na célula, e altura das antenas da estação-base. Embora a Figura 6.19 mostre cada célula com uma estação-base de transceptor posicionada no seu meio, hoje, muitos sistemas posicionam a BTS em pontos de interseção de três células, de modo que uma única estação-base com antenas direcionais possa atender a três células.

O padrão GMS para sistemas celulares 2G utilizam FDM/TDM (rádio) para a interface ar. Lembre-se de que no Capítulo 1, com FDM puro, o canal é dividido em um número de bandas de frequência, sendo que cada banda se dedica a uma chamada. Lembre-se também de que, no Capítulo 1, com FDM puro, o tempo é dividido em quadros, sendo cada quadro dividido em intervalos e cada chamada sendo destinada ao uso de um intervalo específico no quadro rotativo. Nos sistemas combinados FDM/TDM, o canal é dividido em um número de sub-bandas de frequência; dentro de cada sub-banda, o tempo é dividido em quadros e intervalos. Desse modo, para um sistema combinado FDM/TDM, se o canal for dividido em F sub-bandas e o tempo em T intervalos, então o canal poderá suportar chamadas simultâneas $F \cdot T$.

Os sistemas GSM consistem em bandas de frequência de 200-kHz e cada banda suporta oito chamadas TDM. O GSM codifica a voz a 13 kbps e 12,2 kbps. Um padrão concorrente do GSM, o IS-95 CDMA e seu sucessor CDMA 2000, utiliza o acesso múltiplo de divisão de código (consulte Seção 6.2.1) e não a abordagem combinada FDM/TDM, tornando os telefones dos usuários GSM inoperantes em uma rede IS-95 e vice-versa.

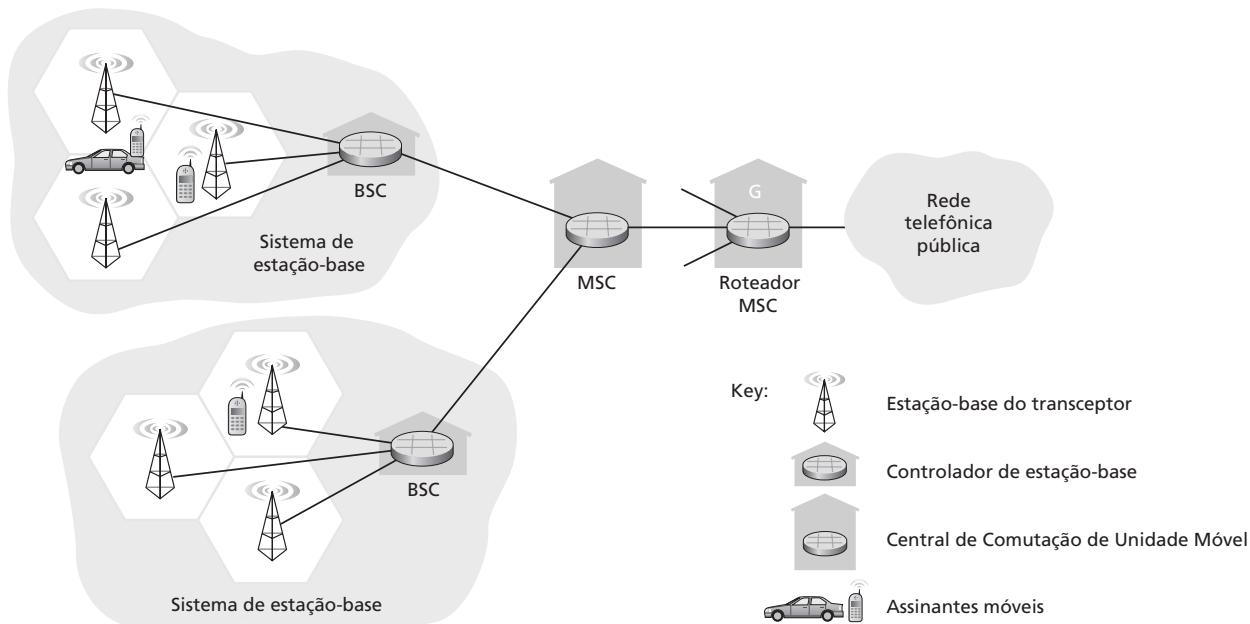


Figura 6.19 Componentes da arquitetura de rede celular 2G GSM

Um controlador de estação-base (BSC) da rede GSM pode estar fisicamente situado com uma BTS, mas, normalmente, uma única BSC prestará serviço a dezenas de estações-base do transceptor. O papel da BSC é alojar os canais de rádio da BTS a assinantes móveis, realizar *paging* (encontrar a célula na qual reside um usuário móvel) e realizar transferência de usuários móveis — um assunto que abordaremos, em poucas palavras, na Seção 6.7.2. O controlador de estação-base e suas estações-base de transceptor, coletivamente, constituem um sistema de estação-base (BSS) GSM.

Como veremos na Seção 9.7, a central de comutação de unidade móvel (MSC) interpreta o papel central na contabilidade e autorização do usuário (por exemplo, determinar se um aparelho móvel permitiu a conexão à rede celular), estabelecimento e interrupção de chamada, e transferências. Normalmente, uma única MSC conterá até cinco BSCs, resultando em aproximadamente 200K assinantes por MSC. Uma rede de provedor celular terá um número de MSCs, possuindo MSCs especiais conhecidas como roteadores das MSCs, conectando a rede de provedor celular à maior rede telefônica pública.

Arquitetura de rede celular, 2,5G e 3G: estendendo a Internet para assinantes celulares

Até agora, nossa discussão se focou em conectar usuários de voz celular à rede telefônica pública. Os usuários móveis estão cada vez mais acessando a Internet através da rede celular por meio de iPhones, Blackberries, laptops e mais. Uma maneira de fazê-lo, usando somente a infraestrutura 2G mostrada na Figura 6.19, é utilizar uma conexão telefônica celular, como uma conexão de discagem para um ISP, assim como muitos usuários, nos anos 1990, usaram suas linhas telefônicas para prover acesso discado a um ISP. A desvantagem desse método, entretanto, são as lentas taxas de transmissão (normalmente dezenas de kbps e, muitas vezes, menor) disponíveis em uma conexão de discagem celular. De maneira ideal, gostaríamos de estender o alcance do IP ao próprio sistema de estação-base utilizando linhas de alta largura de banda e, então, múltiplos canais de voz, ou redes de acesso via rádio aperfeiçoados, para conectar o usuário móvel ao sistema de estação-base a altas taxas. Este é, precisamente, o método utilizado nos sistemas celulares 2G e 3G.

A Figura 6.20 mostra a arquitetura da rede celular para o 2,5G GSM, uma extensão do 2G GSM para prover máximo acesso à Internet com alta velocidade. O método utilizado pelos projetistas do 2,5G GSM é evidente: deixe o núcleo da rede telefônica celular GSM intacto. Isso acontece fornecendo acesso à Internet como uma funcionalidade complementar, e não como uma funcionalidade integrada (e, assim, exigindo mudanças) ao

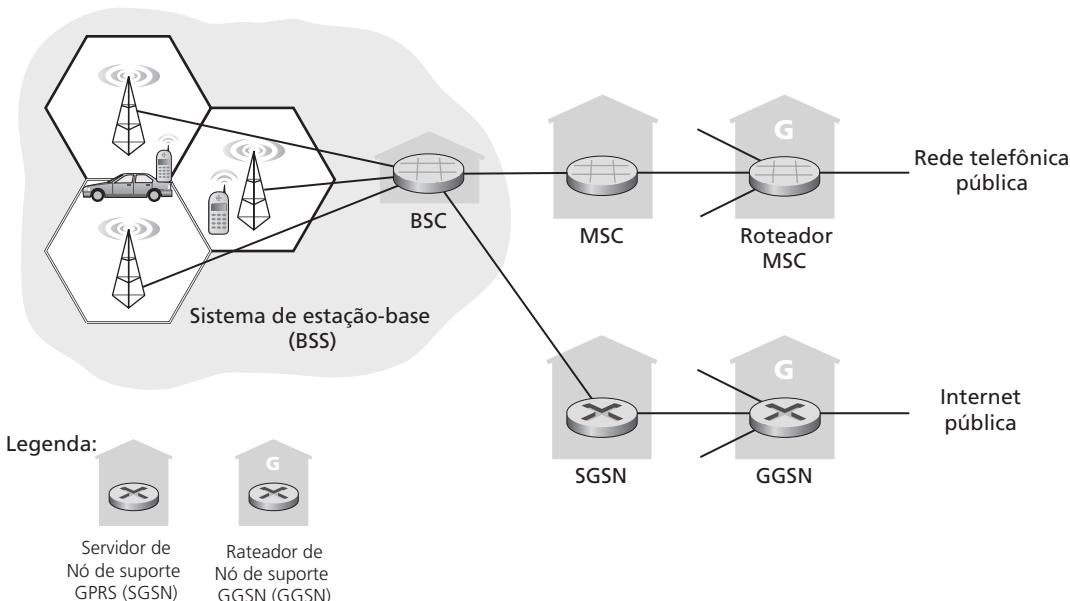


Figura 6.20 Arquitetura de rede de dado e voz celular 2,5G GSM

núcleo da rede telefônica celular existente. A capacidade complementar é implementada na rede de acesso via rádio, a BSC, e por meio da introdução de uma rede de nós distintos chamados de Servidor de Nó de Suporte GPRS (SGSN).

Na BSC, os canais FDM/TDM que conduzem o datagrama IP na interface aérea são encaminhados da BSC ao SGSN, que se comunica com a MSC para realizar a autorização do usuário, transferência e outras funções. Além desse tráfego de sinalização, os datagramas IP da BSC são encaminhadas para/de a Internet pelo SGSN. Na rede de acesso via rádio, o Serviço de Pacote de Rádio Geral (GPRS) foi introduzido no 2,5G GSM para permitir que um usuário utilize, de forma dinâmica, canais múltiplos de rádio para dados IP, resultando em taxas de até 115 Kbps. Seguindo os passos do GPRS, a Taxa de Dados Optimizada para Evolução Global (EDGE) foi introduzida para aumentar as capacidades de taxa de dados da rede GSM/GPRS até 384 kbps. Um excelente panorama sobre EDGE encontra-se em [Ericsson, 2009].

Sistemas celulares 3G devem prover serviço de telefonia (bem como de comunicação de dados a velocidades significativamente mais altas do que seus contrapartes 2,5G. Em particular, sistemas 3G devem prover, obrigatoriamente:

- 144 kbps em velocidades de automóveis.
- 384 kbps para uso estacionário em ambiente externo ou velocidades de quem anda a pé.
- 2 Mbps em ambiente interno.

O Serviço Universal de Telecomunicações Móveis (UMTS), uma das tecnologias 3G mais conhecidas, é uma evolução do 2,5G GSM para suportar as capacidades 3G. A arquitetura da rede UMTS apropriou-se expressivamente da arquitetura da rede GSM estabelecida. Ela deixa, especificamente, as redes de dados 2,5G de voz celular existentes na Figura 6.20 em funcionamento, assim como a rede 2,5G o fez com as redes existentes. Uma mudança significativa no UMTS é que, em vez de usar o método FDMA/TDMA do GSM, o UMTS utiliza uma técnica CDMA denominada *Direct Sequence Wideband CDMA* (CDMA de banda larga de sequência direta) (DS-WCDMA) dentro de intervalos TDMA (com quadros de intervalos de TDMA acessíveis em frequências múltiplas — uma utilização interessante de todos os métodos de compartilhamento de canal que identificamos anteriormente!). Essa mudança exige uma nova rede celular de acesso sem fio operando paralelamente com a rede BSS mostrada na Figura 6.20. O serviço de dados associado à especificação do WCDMA é conhecido como HSDPA/HSUPA (Pacote de Acesso de Alta Velocidade Downlink/Uplink) e promete taxas de dados de até 14 Mbps. Mais detalhes referentes às redes 3G podem ser encontrados no site do Projeto de Parceria da 3^a Geração (3GPP) [3GPP, 2009].



História

Celular móvel 3G versus LANs sem fio

Muitas operadoras de telefonia celular móvel estão disponibilizando sistemas celulares móveis 3G, com taxas de dados de 2 Mbps em ambiente fechado e 384 kbps, ou maior, ao ar livre. Os sistemas 3G estão sendo disponibilizados em faixas licenciadas de radiofrequência, sendo que o preço que algumas operadoras pagam aos governos pela licença é uma quantia considerável. Os sistemas 3G permitirão que os usuários acessem a Internet a partir de localizações remotas enquanto em trânsito, de modo semelhante ao acesso a telefones celulares existente hoje. Por exemplo, a tecnologia 3G permitirá que um usuário acesse informações sobre estradas enquanto dirige, ou informações sobre a programação de cinema enquanto estiver tomando sol na praia. Não obstante, hoje muitos especialistas estão começando a questionar se a tecnologia 3G terá sucesso, dados o seu custo e a concorrência da tecnologia de LAN sem fio. Em particular, esses especialistas argumentam que:

- A infraestrutura emergente de LAN sem fio logo se tornará quase ubíqua. As LANs sem fio IEEE 802.11, que operam a taxas de 54 Mbps, estão sendo amplamente disponibilizadas. Quase todos os computadores portáteis e PDAs já virão da fábrica equipados com placas de LAN 802.11. Além disso, as novidades em equipamentos de Internet — como câmeras sem fio e molduras para fotografias — também usarão pequenas placas de LAN sem fio de baixa potência.
- O WiMAX, que estudamos na Seção 6.3.6, promete serviços de dados em amplas áreas a usuários móveis de vários megabits por segundo ou mais. A Sprint Nextel está fazendo um investimento multibilionário na implementação do WiMAX.
- Estações-base de LANs sem fio também podem manipular equipamentos de telefonia móvel. Os futuros telefones podem ser capazes de se conectar ou a uma rede de telefonia celular ou a uma rede IP utilizando o serviço voz sobre IP semelhante ao Skype, ultrapassando, assim, os serviços de dados 3G e de voz celular da operadora.

É claro que muitos outros especialistas acham que a tecnologia 3G não somente será o maior sucesso, mas também revolucionará drasticamente o modo como trabalhamos e vivemos. É claro que ambas, Wi-Fi e 3G, podem se tornar tecnologias sem fio dominantes, com equipamentos sem fio em trânsito que selecionam automaticamente a tecnologia de acesso que provê o melhor serviço em sua localização física corrente.

Até junho de 2007, 200 milhões de assinantes 3G estavam conectados. Isso é somente 6,7% dos três bilhões de assinantes mundiais da telefonia móvel. Nos países onde o 3G chegou primeiro — Japão e Coreia do Sul — mais da metade de todos os assinantes usa a tecnologia 3G. Na Europa, o país que lidera é a Itália, com um terço de assinantes 3G. Outros países que lideram são o Reino Unido, Áustria, Austrália e Cingapura.

Com a criação e implementação de várias gerações de especificações 3G a caminho, a tecnologia 4G sem fio pode ficar para trás? A resposta é tanto sim como não. Não existe uma definição formal sobre como serão os sistemas 4G, e mesmo que haja fornecedores produzindo equipamento (por exemplo, WiMAX), isso já excede o desempenho dos sistemas 3G. Certamente, no momento em que os sistemas 4G forem definidos e implementados, eles operarão a velocidades mais altas do que os sistemas 3G, alcançando 1 Gbps ou mais; protocolos da Internet mais integrados; e tendem a se focar em serviços de localização, multimídia e segurança.

6.5 Gerenciamento da mobilidade: princípios

Após estudarmos a natureza *sem fio* dos enlaces de comunicação em uma rede sem fio, é hora de voltarmos nossa atenção à *mobilidade* que esses enlaces sem fio possibilitam. No sentido mais amplo, um nó móvel é aquele que muda seu ponto de conexão com a rede ao longo do tempo. Como o termo *mobilidade* adquiriu muitos

significados no mundo da computação e também no mundo da telefonia, será proveitoso, antes de mais nada, considerar diversas dimensões da mobilidade com um certo detalhe.

Do ponto de vista da camada de rede, até que ponto um usuário é móvel? Um usuário fisicamente móvel apresentará à camada de rede conjuntos de desafios muito diferentes dependendo de como ele se movimenta entre pontos de conexão com a rede. Em uma extremidade do espectro na Figura 6.21, um usuário pode carregar consigo um laptop equipado com uma placa de interface de rede sem fio dentro de um edifício. Como vimos na Seção 6.3.4, esse usuário *não* é móvel da perspectiva da camada de rede. Além do mais, se o usuário se associar com o mesmo ponto de acesso independentemente de localização, então ele não será móvel nem mesmo do ponto de vista da camada de enlace.

Na outra extremidade do espectro, considere o usuário que está dentro de um BMW, correndo pela estrada a 150 quilômetros por hora, passando por várias redes de acesso sem fio e querendo manter uma conexão TCP ininterrupta com uma aplicação remota durante a viagem. Esse usuário é *definitivamente* móvel! Entre esses extremos está um usuário que leva seu laptop de uma localização (por exemplo, escritório ou quarto de dormir) a outra (por exemplo, lanchonete, biblioteca) e quer se conectar à rede na nova localização. Esse usuário também é móvel (embora menos do que o motorista do BMW!), mas não precisa manter uma conexão ativa enquanto se movimenta entre pontos de conexão com a rede. A Figura 6.21 ilustra esse espectro de mobilidade do usuário da perspectiva da camada de rede.

Qual é a importância de o endereço do nó móvel permanecer sempre o mesmo? Com telefonia móvel, o número de seu telefone — essencialmente o endereço de camada de rede do seu aparelho — permanece o mesmo quando você transita entre uma provedora de rede de telefonia móvel e outra. Um laptop também terá de manter o mesmo endereço IP enquanto se movimenta entre redes IP?

A resposta a essa pergunta dependerá muito da aplicação que está sendo executada. Para o motorista da BMW que quer manter uma conexão TCP ininterrupta com uma aplicação remota enquanto voa pela estrada, seria conveniente manter o mesmo endereço IP. Lembre-se de que dissemos, no Capítulo 3, que uma aplicação de Internet precisa conhecer o endereço IP e o número de porta da entidade remota com a qual está se comunicando. Se uma entidade móvel puder manter seu endereço IP enquanto estiver em trânsito, a mobilidade torna-se invisível do ponto de vista da aplicação. Essa transparência é de grande valor — uma aplicação não precisa se preocupar com uma potencial mudança de endereço IP e o mesmo código de aplicação servirá igualmente a conexões móveis e não móveis. Na próxima seção veremos que o IP móvel provê essa transparência, permitindo que um nó móvel mantenha seu endereço IP permanente enquanto se movimenta entre redes.

Por outro lado, um usuário móvel menos sofisticado poderia querer simplesmente desligar seu laptop no escritório, levá-lo para casa, ligá-lo novamente e continuar trabalhando em casa. Se o laptop funciona primordialmente como um cliente em aplicações cliente-servidor (por exemplo, enviar/receber e-mails, navegar pela Web, usar Telnet com um hospedeiro remoto), o endereço IP particular utilizado pelo laptop não é muito importante. Em particular, é possível passar muito bem com um endereço temporariamente alocado ao laptop pelo ISP ao qual a residência está ligada. Na Seção 4.4 vimos que o DHCP já provê essa funcionalidade.

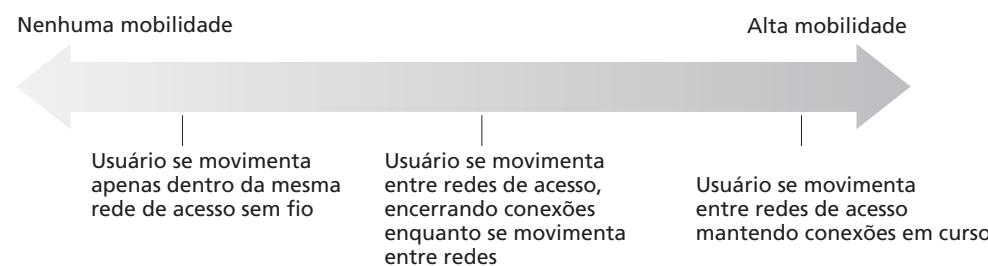


Figura 6.21 Vários graus de mobilidade do ponto de vista da camada de rede

Qual é a infraestrutura cabeada de suporte disponível? Em todos os quatro cenários descritos, admitimos implicitamente que existe uma infraestrutura fixa à qual o usuário móvel pode se conectar, por exemplo, a rede ISP à qual a residência está ligada, a rede de acesso sem fio no local de trabalho ou as redes de acesso sem fio ao longo da rodovia. E se tal infraestrutura não existir? Se dois usuários estiverem a uma distância que permita comunicação, eles poderão estabelecer uma conexão de rede na ausência de qualquer outra infraestrutura de camada de rede? As redes ad hoc proveem exatamente essas capacidades. Essa área, que está em rápido desenvolvimento e representa a pesquisa de ponta em redes móveis, está além do escopo deste livro. [Perkins, 2000] e as páginas Web do grupo de trabalho Mobile Ad Hoc Network (manet) da IETF apresentam um tratamento completo do assunto.

Para ilustrar as questões que envolvem permitir que um usuário mantenha conexões em curso enquanto se movimenta entre redes, vamos considerar uma analogia com seres humanos. Um adulto de 20 e poucos anos que sai da casa de seus pais torna-se móvel, pois passa a morar em uma série de quartos e/ou apartamentos e está sempre mudando de endereço. Se uma velha amiga quiser entrar em contato com ele, como conseguirá encontrar o endereço de seu amigo móvel? Uma maneira comum de fazer isso é entrar em contato com a família, já que um jovem móvel costuma informar seus novos endereços à família (nem que seja apenas para que seus pais possam lhe enviar dinheiro para ajudar a pagar o aluguel!). A residência da família, com seu endereço permanente, torna-se aquele único lugar a que outros podem se dirigir como uma primeira etapa para estabelecer comunicação com o jovem móvel. As comunicações posteriores da velha amiga podem ser indiretas (por exemplo, pelo envio de uma carta primeiramente à casa dos pais, que então encaminharão a carta ao jovem móvel) ou diretas (por exemplo, a velha amiga utiliza o endereço informado pelos pais para enviar uma carta diretamente a seu amigo móvel).

Em um ambiente de rede, a residência permanente de um nó móvel (tal como um laptop ou um PDA) é conhecida como **rede nativa** (home network) e a entidade dentro dessa rede que executa o gerenciamento de funções de mobilidade em nome do nó móvel (e que serão discutidas mais adiante) é conhecida como **agente nativo** (home agent). A rede na qual o nó móvel está residindo atualmente é conhecida como **rede externa** (foreign network) ou **rede visitada** (visited network), e a entidade dentro da rede externa que auxilia o nó móvel no gerenciamento das funções de mobilidade discutidas adiante é conhecida como **agente externo** (foreign agent). No caso de profissionais móveis, suas redes nativas possivelmente seriam as redes das empresas em que trabalham, enquanto a rede visitada poderia ser a rede de um colega que estão visitando. Um **correspondente** é a entidade que quer se comunicar com o nó móvel. A Figura 6.22 ilustra esses conceitos, bem como conceitos de endereçamento considerados mais adiante. Observe que, na Figura 6.22, os agentes são colocados junto aos roteadores (por exemplo, como processos que funcionam em roteadores), mas, alternativamente, poderiam estar funcionando em outros hospedeiros ou servidores na rede.

6.5.1 Endereçamento

Já observamos que, para que a mobilidade do usuário seja transparente para aplicações de rede, é desejável que um nó móvel mantenha seu endereço quando transita de uma rede para outra. Quando um nó móvel residir em uma rede externa, todo o tráfego enviado ao endereço permanente do nó agora precisará ser roteado para a rede externa. Como isso pode ser feito? Uma opção é a rede externa anunciar para todas as outras redes que o nó móvel agora reside em sua rede, o que poderia ser feito mediante a costumeira troca de informações de roteamento interdomínios e intradomínios e exigiria poucas mudanças na infraestrutura de roteamento existente. A rede externa poderia simplesmente anunciar a seus vizinhos que tem uma rota altamente específica para o endereço permanente do nó móvel (isto é, essencialmente, informar a outras redes que ela tem o caminho correto para rotear datagramas para o endereço permanente do nó móvel; ver Seção 4.4). Esses vizinhos então propagariam essa informação de roteamento por toda a rede como parte do procedimento normal de atualização de informações de roteamento e tabelas de repasse. Quando o nó móvel sair de uma rede externa e se juntar a uma outra, a nova rede externa anunciaría uma nova rota, altamente específica, até o nó móvel e a antiga rede externa retiraria suas informações referentes ao nó móvel.

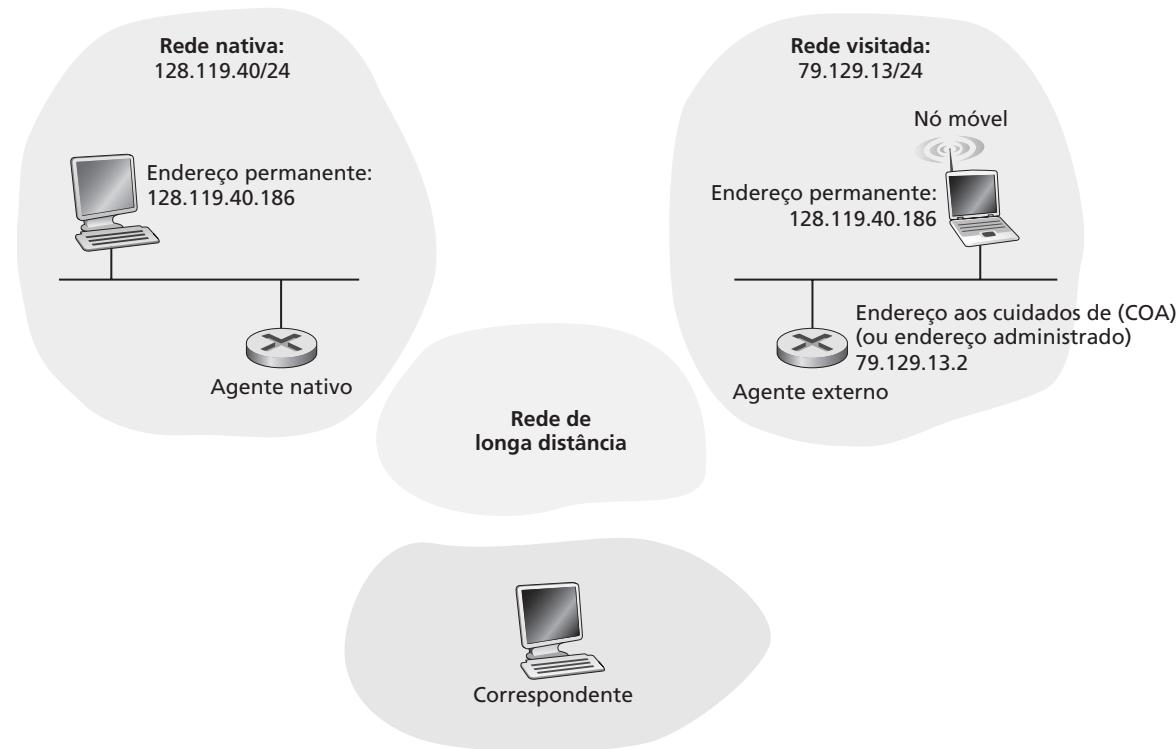


Figura 6.22 Elementos iniciais de uma arquitetura de rede móvel

Esse procedimento resolve dois problemas de uma vez só e o faz sem promover mudanças significativas na infraestrutura da camada de rede. Outras redes conhecem a localização do nó móvel e é fácil rotear datagramas para o nó móvel, visto que as tabelas de repasse dirigirão datagramas à rede externa. Uma desvantagem significativa, contudo, é a da escalabilidade. Se a responsabilidade pelo gerenciamento da mobilidade tivesse de recair sobre os roteadores da rede, eles teriam de manter registros em tabelas de repasse para potencialmente milhões de nós móveis e atualizar esses registros à medida que os nós se movimentassem. Algumas desvantagens adicionais serão exploradas nos problemas ao final deste capítulo.

Uma abordagem alternativa (que tem sido adotada na prática) é passar a funcionalidade de mobilidade do núcleo da rede para a borda da rede — um tema recorrente em nosso estudo da arquitetura da Internet. Um modo natural de fazer isso é por meio da rede nativa do nó móvel. De maneira muito semelhante ao modo como os pais daquele jovem de 20 e poucos anos monitoram a localização de seu filho, o agente nativo na rede nativa do nó móvel pode monitorar a rede externa na qual o nó móvel reside. Certamente será preciso um protocolo (ou um agente externo representando o nó móvel) entre o nó móvel e o agente nativo para atualizar a localização do nó móvel.

Agora vamos considerar o agente externo com mais detalhes. A abordagem conceitualmente mais simples, mostrada na Figura 6.22, é localizar agentes externos nos roteadores de borda na rede externa. Um dos papéis do agente externo é criar o denominado **endereço aos cuidados de** (*care-of-address* — COA) **ou endereço administrado** para o nó móvel, sendo que a parte da rede do endereço COA combinaria com a parte da rede do endereço da rede externa. Assim, há dois endereços associados com um nó móvel, seu **endereço permanente** (análogo ao endereço da família do nosso jovem móvel) e seu endereço COA, às vezes denominado **endereço externo** (análogo ao endereço da casa onde nosso jovem móvel está residindo atualmente). No exemplo da Figura 6.22, o endereço permanente do nó móvel é 128.119.40.186. Quando está visitando a rede 79.129.13/24, o nó móvel tem um COA 79.129.13.2. Um segundo papel desempenhado pelo agente externo é informar ao agente nativo que o nó móvel está residindo em sua rede (do agente externo) e tem o dado endereço COA. Veremos, em breve, que o COA pode ser utilizado para ‘rerrotear’ datagramas para o nó móvel via seu agente externo.

Embora tenhamos separado a funcionalidade do nó móvel e o agente externo, vale a pena observar que o nó móvel também pode assumir a responsabilidade do agente externo. Por exemplo, o nó móvel poderia obter um COA na rede externa (por exemplo, utilizando um protocolo como o DHCP) e ele mesmo informar seu COA ao agente nativo.

6.5.2 Roteamento para um nó móvel

Agora já vimos como um nó móvel obtém um COA e como é possível informar esse endereço ao agente nativo. Mas conseguir que o agente nativo conheça o COA resolve apenas parte do problema. Como datagramas devem ser endereçados e repassados para o nó móvel? Visto que somente o agente nativo (e não os roteadores no âmbito total da rede) conhece a localização do nó móvel, já não será mais suficiente simplesmente endereçar um datagrama para o endereço permanente do nó móvel e enviá-lo para a infraestrutura de camada de rede. É preciso fazer algo mais. Duas abordagens podem ser identificadas, as quais denominaremos roteamento indireto e roteamento direto.

Roteamento indireto para um nó móvel

Vamos considerar primeiramente um correspondente que quer enviar um datagrama a um nó móvel. Na abordagem de **roteamento indireto** o correspondente simplesmente endereça o datagrama ao endereço permanente do nó móvel, envia o datagrama para a rede e fica alegremente inconsciente quanto ao nó móvel residir em sua rede nativa ou estar visitando uma rede externa; assim, a mobilidade é completamente transparente para o correspondente. Esses datagramas são primeiramente roteados, como sempre, para a rede local do nó móvel. Isso é ilustrado na etapa 1 da Figura 6.23.

Agora vamos voltar nossa atenção ao agente nativo. Além de ser responsável por interagir com um agente externo para monitorar o COA do nó móvel, o agente nativo tem uma outra função muito importante. Sua segunda

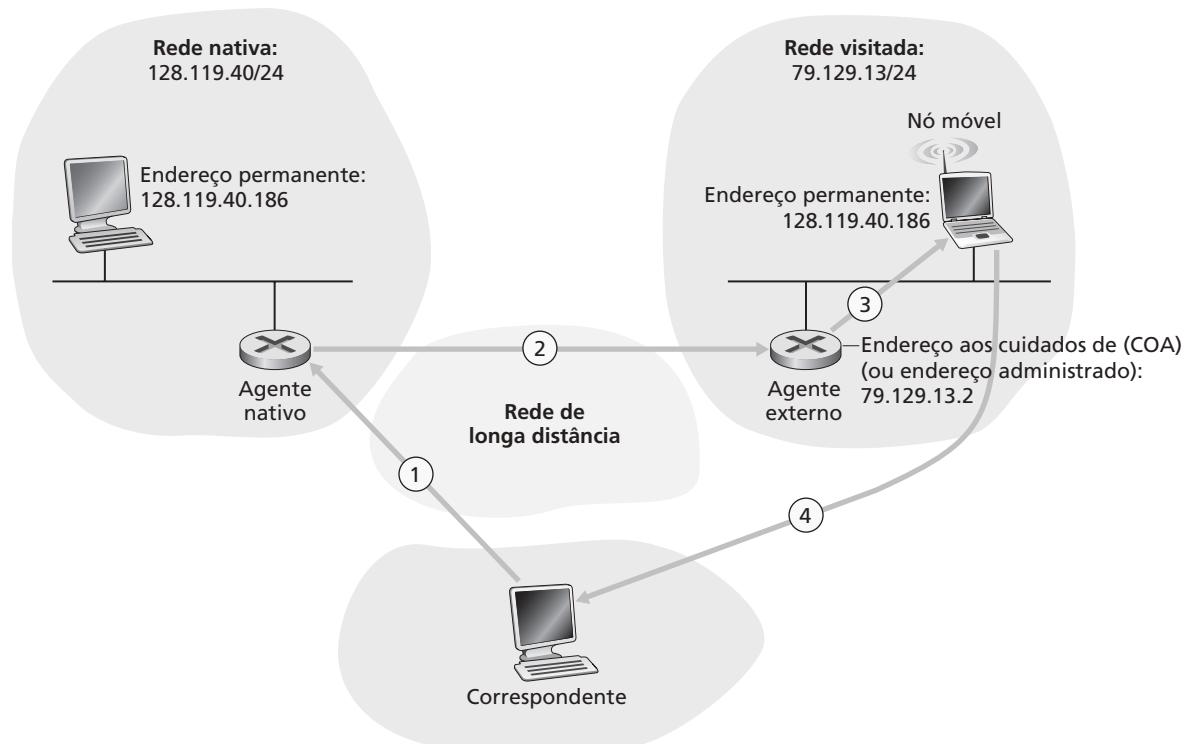


Figura 6.23 Repasse indireto para um nó móvel

tarefa é ficar à espreita de datagramas que chegam e são endereçados a nós cuja rede nativa é a rede do agente nativo, mas que estão correntemente residindo em uma rede externa. O agente nativo intercepta esses datagramas e então os repassa a um nó móvel por um processo de duas etapas. Primeiramente o datagrama é repassado para o agente externo usando o COA do nó móvel (etapa 2 na Figura 6.23) e então é repassado do agente externo para o nó móvel (etapa 3 na Figura 6.23).

É instrutivo considerar esse roteamento com mais detalhes. O agente nativo precisará endereçar o datagrama usando o COA do nó móvel, de modo que a camada de rede roteará o datagrama para a rede externa. Por outro lado, é desejável deixar intacto o datagrama do correspondente, visto que a aplicação que recebe o datagrama deve ficar inconsciente de que o datagrama foi repassado por meio do agente nativo. Ambas as metas podem ser cumpridas fazendo com que o agente nativo **encapsule** o datagrama original completo do correspondente dentro de um novo (e maior) datagrama. Esse datagrama maior é endereçado e entregue ao COA do nó móvel. O agente externo ‘proprietário’ do COA receberá e desencapsulará o datagrama, isto é, removerá o datagrama original do correspondente de dentro daquele datagrama maior de encapsulamento e repassará o datagrama original (etapa 3 na Figura 6.23) para o nó móvel. A Figura 6.24 mostra um datagrama original de um correspondente sendo enviado para a rede local, um datagrama encapsulado sendo enviado ao agente externo e o datagrama original sendo entregue ao nó móvel. O leitor observador notará que o encapsulamento/desencapsulamento descrito aqui é idêntico à noção de implementação de túnel discutida no Capítulo 4 no contexto do IP multicast e do IPv6.

A seguir, vamos considerar como um nó móvel envia datagramas a um correspondente. Isso é muito simples, pois o nó móvel pode endereçar seu datagrama *diretamente* ao correspondente (usando seu próprio endereço permanente como o endereço de origem e o endereço do correspondente como o endereço de destino). Visto que o nó móvel conhece o endereço do correspondente, não há necessidade de rotear o datagrama de volta por meio do agente nativo. Isso é mostrado como etapa 4 na Figura 6.23.

Vamos resumir o que discutimos sobre roteamento indireto relacionando as novas funcionalidades de camada de rede exigidas para suportar mobilidade.

Um protocolo nó móvel–agente externo. O nó móvel fará seu registro no agente externo ao se conectar à rede externa. De modo semelhante, um nó móvel cancelará seu registro no agente externo quando sair da rede externa.

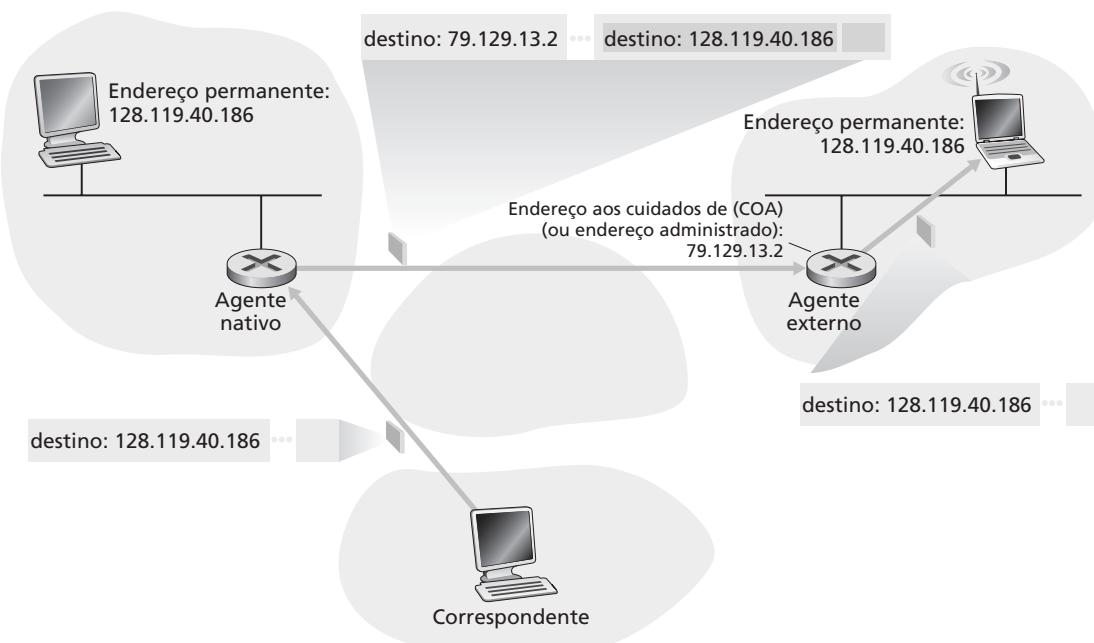


Figura 6.24 Encapsulamento e desencapsulamento



Um protocolo de registro agente externo–agente nativo. O agente externo registrará o COA do nó móvel no agente nativo. Um agente externo não precisa cancelar explicitamente um COA quando um nó móvel sai da rede porque o registro subsequente de um novo COA se encarregará disso quando o nó móvel passar para uma nova rede.

Um protocolo de encapsulamento de datagrama para o agente nativo. Encapsulamento e repasse do datagrama original do correspondente dentro de um datagrama endereçado ao COA.

Um protocolo de desencapsulamento de datagrama para o agente externo. Extração do datagrama original do correspondente de dentro do datagrama de encapsulamento e repasse do datagrama original ao nó móvel.

A discussão anterior provê todas as peças — agentes externos, agente nativo e roteamento indireto — de que um nó móvel necessita para manter uma conexão em curso enquanto transita entre redes. Como um exemplo de como essas peças se encaixam, admita que o nó móvel está ligado à rede externa A, registrou um COA no agente local na rede A e está recebendo datagramas que estão sendo roteados indiretamente por meio de seu agente nativo. O nó móvel agora passa para a rede externa B e se registra no agente externo na rede B, que informa ao agente nativo o novo COA do nó móvel. Desse ponto em diante, o agente nativo rerroteará datagramas para a rede externa B. No que diz respeito ao correspondente, a mobilidade é transparente — datagramas são roteados por meio do mesmo agente nativo antes e depois de o nó móvel mudar de rede. No que diz respeito ao agente nativo, não há nenhuma disruptão no fluxo de datagramas — datagramas que chegam são primeiramente repassados para a rede externa A; após a mudança no COA, datagramas são repassados para a rede externa B. Mas o nó móvel verá um fluxo ininterrupto de datagramas ao se movimentar entre redes? Contanto que seja pequeno o tempo transcorrido entre o desligamento do nó móvel da rede A (ponto em que ele não pode mais receber datagramas via A) e sua conexão com a rede B (ponto em que registrará um novo COA no agente nativo da rede), poucos datagramas serão perdidos. Lembre-se de que dissemos, no Capítulo 3, que conexões fim-a-fim podem sofrer perda de datagramas devido a congestionamento na rede. Por conseguinte, a perda ocasional de datagramas dentro de uma conexão quando um nó se move entre redes não é, de maneira alguma, um problema catastrófico. Se for preciso uma comunicação livre de perdas, mecanismos de camadas superiores recuperarão a perda de dados, quer essas perdas resultem do congestionamento da rede ou da mobilidade do usuário.

Uma abordagem indireta de roteamento é utilizada no padrão IP móvel [RFC 3344], como discutiremos na Seção 6.6.

Roteamento direto para um nó móvel

A abordagem do roteamento indireto ilustrada na Figura 6.23 sofre de uma ineficiência conhecida como **problema do roteamento triangular** — datagramas endereçados ao nó móvel devem ser roteados primeiramente para o agente nativo e em seguida para a rede externa, mesmo quando existir uma rota muito mais eficiente entre o correspondente e o nó móvel. No pior caso, imagine um usuário móvel que está visitando a rede externa de um colega. Os dois estão sentados lado a lado e trocando dados pela rede. Datagramas do correspondente (nesse caso, do colega do visitante) são roteados para o agente nativo do usuário móvel e, então, novamente de volta para a rede externa!

O **roteamento direto** supera a ineficiência do roteamento triangular, mas o faz à custa de complexidade adicional. Na abordagem do roteamento direto, um **agente correspondente** na rede do correspondente primeiramente aprende o COA do nó móvel. Isso pode ser realizado fazendo com que o agente correspondente consulte o agente nativo, admitindo que (como é o caso no roteamento indireto) o nó móvel tem um valor atualizado para seu COA registrado no seu agente nativo. Também é possível que o próprio correspondente execute a função do agente externo, exatamente como um nó móvel poderia executar a função do agente externo. Essa situação é ilustrada como as etapas 1 e 2 na Figura 6.25. O agente correspondente então implementa um túnel para os datagramas diretamente até o COA do nó móvel, de modo análogo ao túnel executado pelo agente nativo, etapas 3 e 4 da Figura 6.25.

Conquanto supere o problema do roteamento triangular, o roteamento direto introduz dois importantes desafios adicionais:

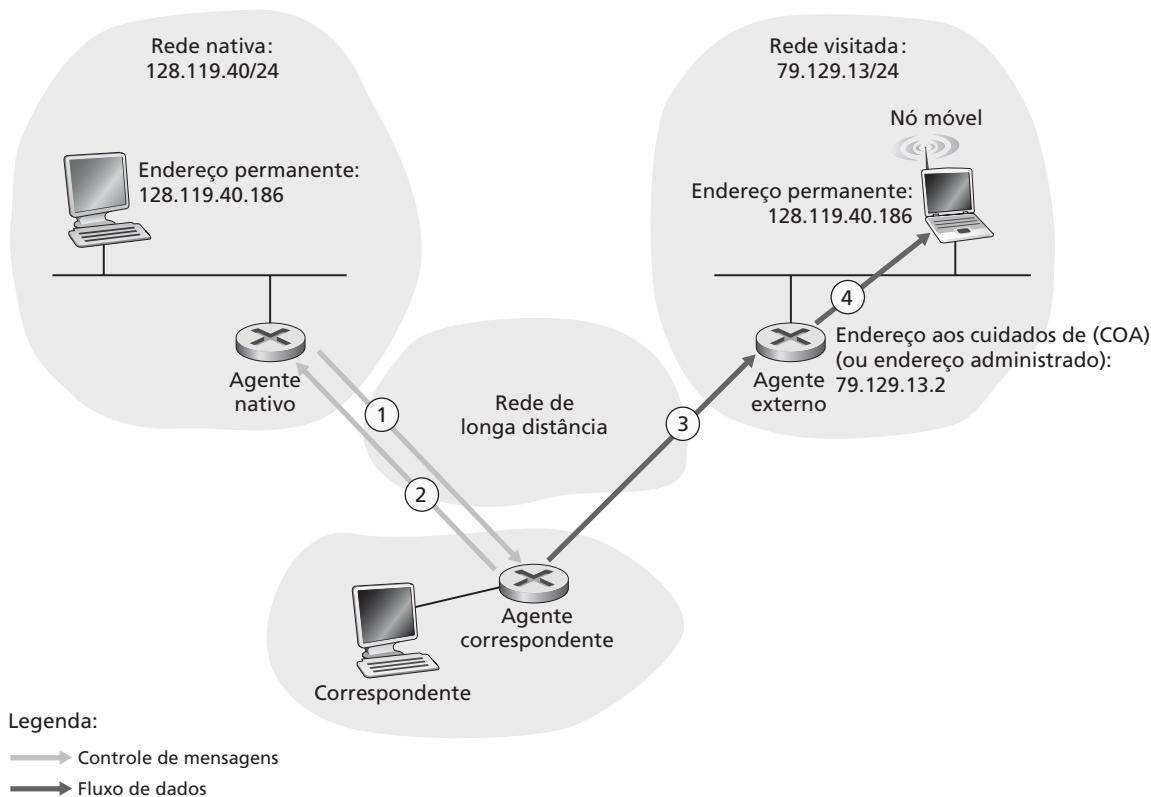


Figura 6.25 Roteamento direto para um usuário móvel

É preciso um **protocolo de localização de usuário móvel** para o agente correspondente consultar o agente nativo de modo a obter o COA do nó móvel (etapas 1 e 2 da Figura 6.25).

Quando o nó móvel passa de uma rede externa para outra, como os dados são repassados, agora, para a nova rede externa? No caso do roteamento indireto, esse problema era facilmente resolvido atualizando-se o COA mantido pelo agente nativo. Todavia, com roteamento direto, o agente correspondente consulta o COA junto ao agente nativo apenas uma vez, no início da sessão. Assim, atualizar o COA do agente nativo, embora necessário, não será suficiente para resolver o problema do roteamento de dados para a nova rede externa do nó móvel.

Uma solução seria criar um novo protocolo para notificar a mudança de COA ao correspondente. Uma solução alternativa, que, como veremos, é adotada na prática em redes GSM, funciona da seguinte maneira: suponha que estão sendo repassados dados correntemente para o nó móvel na rede externa onde ele estava localizado quando a sessão foi iniciada (etapa 1 na Figura 6.26). O agente externo naquela rede externa onde o nó móvel foi encontrado pela primeira vez será denominado **agente externo âncora**. Quando o nó móvel passar para uma nova rede externa (etapa 2 na Figura 6.26), ele se registrará junto ao novo agente externo (etapa 3) e esse novo agente externo fornecerá ao agente externo âncora o novo COA do nó móvel (etapa 4). Quando o agente externo âncora receber um datagrama encapsulado para um nó móvel que já deixou a rede, ele então poderá reencapsular o datagrama e repassá-lo para o nó móvel (etapa 5) usando o novo COA. Se, mais tarde, o nó móvel passar para mais uma outra rede externa, o agente externo nessa nova rede visitada então contatará o agente externo âncora para estabelecer repasse para essa nova rede externa.

6.6 IP móvel

A arquitetura e os protocolos da Internet para suporte de mobilidade, conhecidos coletivamente como IP móvel, estão definidos primariamente no RFC 3344 para IPv4. O IP móvel é um protocolo flexível que suporta

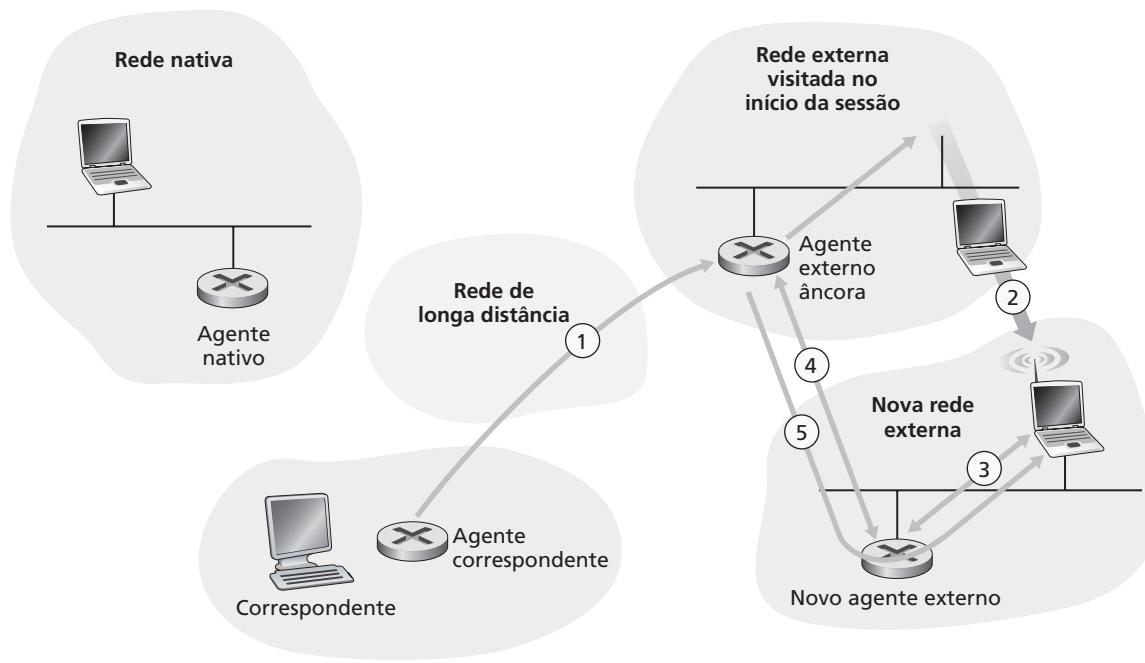


Figura 6.26 Transferência móvel entre redes com roteamento direto

muitos modos de operação diferentes (por exemplo, operação com ou sem um agente externo) várias maneiras para agentes e nós móveis descobrirem uns aos outros, utilização de um único COA ou de vários COAs e diversas formas de encapsulamento. Por isso, o IP móvel é um protocolo complexo cuja descrição detalhada exigiria um livro inteiro; um desses livros é [Perkins, 1998b]. Aqui, nossa modesta meta é prover uma visão geral dos aspectos mais importantes do IP móvel e ilustrar sua utilização em alguns cenários comuns.

A arquitetura do IP móvel contém muitos dos elementos que acabamos de considerar, incluindo os conceitos de agentes nativos, agentes externos, endereços administrados e encapsulamento/desencapsulamento. O padrão corrente [RFC 3344] especifica a utilização de roteamento indireto para o nó móvel.

O padrão IP móvel consiste em três partes principais:

- *Descoberta de agente.* O IP móvel define os protocolos utilizados por um agente nativo ou por um agente externo para anunciar seus serviços a nós móveis e protocolos para que os nós móveis solicitem os serviços de um agente externo ou nativo.
- *Registro no agente nativo.* O IP móvel define os protocolos usados pelo nó móvel e/ou agente externo para registrar e anular os registros de COAs no agente local de um nó móvel.
- *Roteamento indireto de datagramas.* O padrão também define a maneira pela qual datagramas são repassados para nós móveis por um agente nativo, incluindo regras para repassar datagramas, regras para manipular condições de erro e diversas formas de encapsulamento [RFC 2003, RFC 2004].

Considerações de segurança têm destaque em todo o padrão IP móvel. Por exemplo, a autenticação de um nó móvel é claramente necessária para impedir que um usuário mal-intencionado registre no agente nativo um falso endereço administrado, o que poderia fazer com que datagramas endereçados a um endereço IP sejam direcionados ao usuário mal-intencionado. O IP móvel consegue segurança usando muitos dos mecanismos que examinaremos no Capítulo 8, portanto, não consideraremos a questão da segurança de endereços na discussão a seguir.

Descoberta de agente

Um nó IP móvel que está chegando a uma nova rede, quer esteja se conectando a uma rede externa ou retornando à sua rede nativa, tem de aprender a identidade do correspondente externo ou do agente nativo. Realmente, é a descoberta de um novo agente externo, com um novo endereço de rede, que permite que a camada

de rede em um nó móvel aprenda que ele passou para uma nova rede externa. Esse processo é conhecido como **descoberta de agente**. A descoberta de agente pode ser realizada de duas maneiras: via anúncio de agente ou via solicitação de agente.

No caso do **anúncio de agente**, um agente externo ou nativo anuncia seus serviços usando uma extensão do protocolo de descoberta de roteador existente [RFC 1256]. O agente faz *broadcast* periódico de uma mensagem ICMP tendo 9 no campo de tipo (descoberta de roteador) em todos os enlaces aos quais está conectado. A mensagem de descoberta de roteador contém o endereço IP do roteador (isto é, o agente), o que permite que um nó móvel aprenda o endereço IP do agente. A mensagem de descoberta de roteador também contém uma extensão de anúncio de agente de mobilidade que contém informações adicionais de que o nó móvel necessita. Entre os campos mais importantes na extensão estão os seguintes:

- **Bit do agente nativo (H)**. Indica que o agente é um agente nativo para a rede na qual reside.
- **Bit de agente externo (F)**. Indica que o agente é um agente externo para a rede na qual reside.
- **Bit de registro obrigatório (R)**. Indica que um usuário móvel nessa rede *deve* se registrar em um agente externo. Em particular, um usuário móvel não pode obter um endereço administrado na rede externa (por exemplo, usando DHCP) e assumir a funcionalidade de agente externo para si mesmo sem se registrar no agente externo.
- **Bits de encapsulamento M, G**. Indicam se será utilizada uma forma de encapsulamento que não seja o encapsulamento IP em IP.
- **Campos de endereços administrados (COA)**. Uma lista de um ou mais endereços administrados fornecida pelo agente externo. No exemplo logo a seguir, o COA estará associado com o agente externo, que receberá datagramas enviados ao COA e então os repassará para o nó móvel adequado. O usuário móvel selecionará um desses endereços como seu COA ao se registrar no seu agente nativo.

A Figura 6.27 ilustra alguns dos principais campos na mensagem de anúncio de agente.

Com **solicitação de agente**, um nó móvel que quer conhecer agentes sem esperar para receber um anúncio de agente pode emitir uma mensagem de solicitação de agente em *broadcast*, que é, simplesmente, uma mensagem ICMP cujo valor de tipo é 10. Ao receber a solicitação, um agente emite um anúncio *unicast* de agente diretamente ao nó móvel, que, então, procederá como se tivesse recebido um anúncio não solicitado.

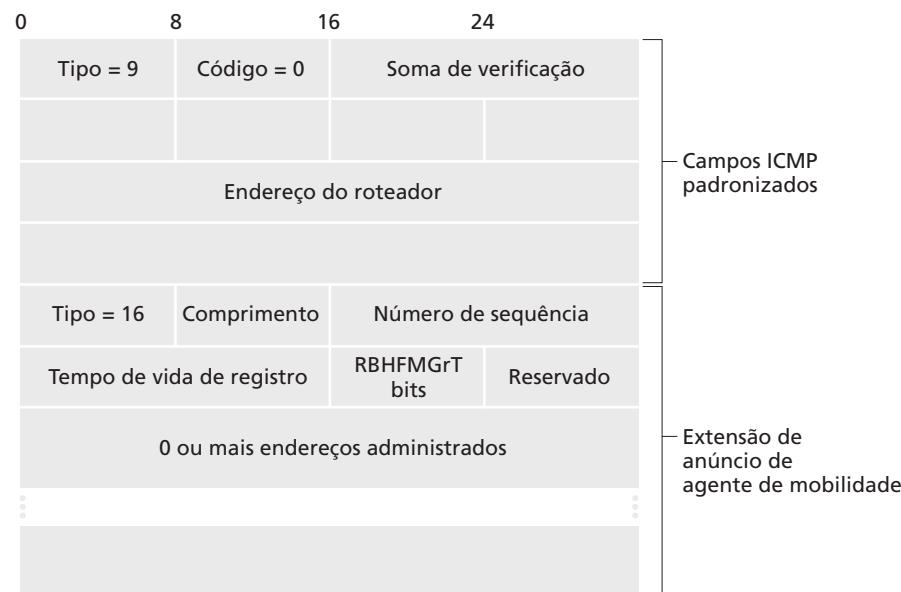


Figura 6.27 Mensagem ICMP de descoberta de roteador com extensão de anúncio de agente de mobilidade

Registro no agente nativo

Tão logo um nó móvel IP tenha recebido um COA, esse endereço deve ser registrado no agente nativo, o que pode ser feito via o agente externo (que, então, registra o COA no agente nativo) ou diretamente pelo próprio nó móvel IP. A seguir, consideramos o primeiro caso. Há quatro etapas envolvidas.

1. Após o recebimento de um anúncio de agente externo, um nó móvel envia uma mensagem de registro IP móvel ao agente externo. A mensagem de registro é carregada dentro de um datagrama UDP e enviada à porta 434. A mensagem leva um COA anunciado pelo agente externo, o endereço do agente nativo (HA), o endereço permanente do nó móvel (MA), o tempo de vida de registro requerido e uma identificação de registro de 64 bits. O tempo de vida de registro requerido é o número de segundos durante os quais o registro será válido. Se o registro não for renovado no agente nativo dentro do tempo de vida especificado, ele se tornará inválido. O identificador de registro age como um número de sequência e serve para combinar uma resposta de registro recebida com uma solicitação de registro, como discutiremos mais adiante.
2. O agente externo recebe a mensagem de registro e registra o endereço IP permanente do nó móvel. Agora o agente externo sabe que deve procurar datagramas que contenham um datagrama encapsulado cujo endereço de destino combine com o endereço permanente do nó móvel. O agente externo então envia uma mensagem de registro IP móvel (novamente dentro de um datagrama UDP) à porta 434 do agente nativo. A mensagem contém o COA, o HA, o MA, o formato de encapsulamento requisitado, o tempo de vida de registro requisitado e a identificação do registro.
3. O agente nativo recebe a requisição de registro e verifica sua autenticidade e correção. O agente nativo vincula o endereço IP permanente do nó móvel ao COA; no futuro, datagramas que chegarem ao agente nativo e endereçados ao nó móvel serão encapsulados e enviados por túnel até o COA. O agente nativo envia uma resposta de registro IP móvel contendo o HA, o MA, o tempo de vida de registro vigente e a identificação de registro da solicitação que está sendo atendida com essa resposta.
4. O agente externo recebe a resposta de registro e então a repassa ao nó móvel.

Nesse ponto o registro está concluído e o nó móvel pode receber datagramas enviados a seu endereço permanente. A Figura 6.28 ilustra essas etapas. Note que o agente nativo especifica um tempo de vida menor do que o tempo de vida requisitado pelo nó móvel.

Um agente externo não precisa anular explicitamente um registro de COA quando um nó móvel sai da rede. Isso ocorrerá automaticamente quando o nó móvel passar para uma nova rede (seja uma outra rede externa, seja sua rede nativa) e registrar um novo COA.

O padrão IP móvel permite muitos cenários e capacidades adicionais além das que acabamos de descrever. O leitor interessado deve consultar [Perkins, 1998b; RFC 3344].

6.7 Gerenciamento de mobilidade em redes celulares

Agora que acabamos de examinar como a mobilidade é gerenciada em redes IP, vamos voltar nossa atenção a redes cujo histórico de suporte à mobilidade é ainda mais longo — redes de telefonia celular. Enquanto na Seção 6.4 focalizamos o enlace sem fio do primeiro salto em redes celulares, aqui focalizaremos a mobilidade utilizando a arquitetura GSM de rede celular [Goodman, 1997; Mouly, 1992; Scourias, 1997; Kaaranen 2001; Korhonen, 2003; Turner, 2009] como objeto de nosso estudo, visto que é uma tecnologia madura e amplamente disponibilizada. Como no caso do IP móvel, veremos que vários dos princípios fundamentais que identificamos da Seção 6.5 estão incorporados à arquitetura de rede do GSM.

Do mesmo modo que o IP móvel, o GSM adota uma abordagem de roteamento indireto (veja Seção 6.5.2), primeiramente roteando a chamada do correspondente para a rede nativa do nó móvel e daí para a rede visitada. Em terminologia GSM a rede nativa do nó móvel é denominada **rede pública terrestre móvel nativa (PLMN nativa)**. Visto que o acrônimo PLMN é um pouco complicado, e sempre firmes na nossa decisão de evitar uma sopa de letrinhas, denominaremos a rede PLMN nativa GSM simplesmente **rede nativa**. A rede nativa é a provedora celular da qual o usuário móvel é assinante (isto é, a provedora que cobra mensalmente do usuário pelo

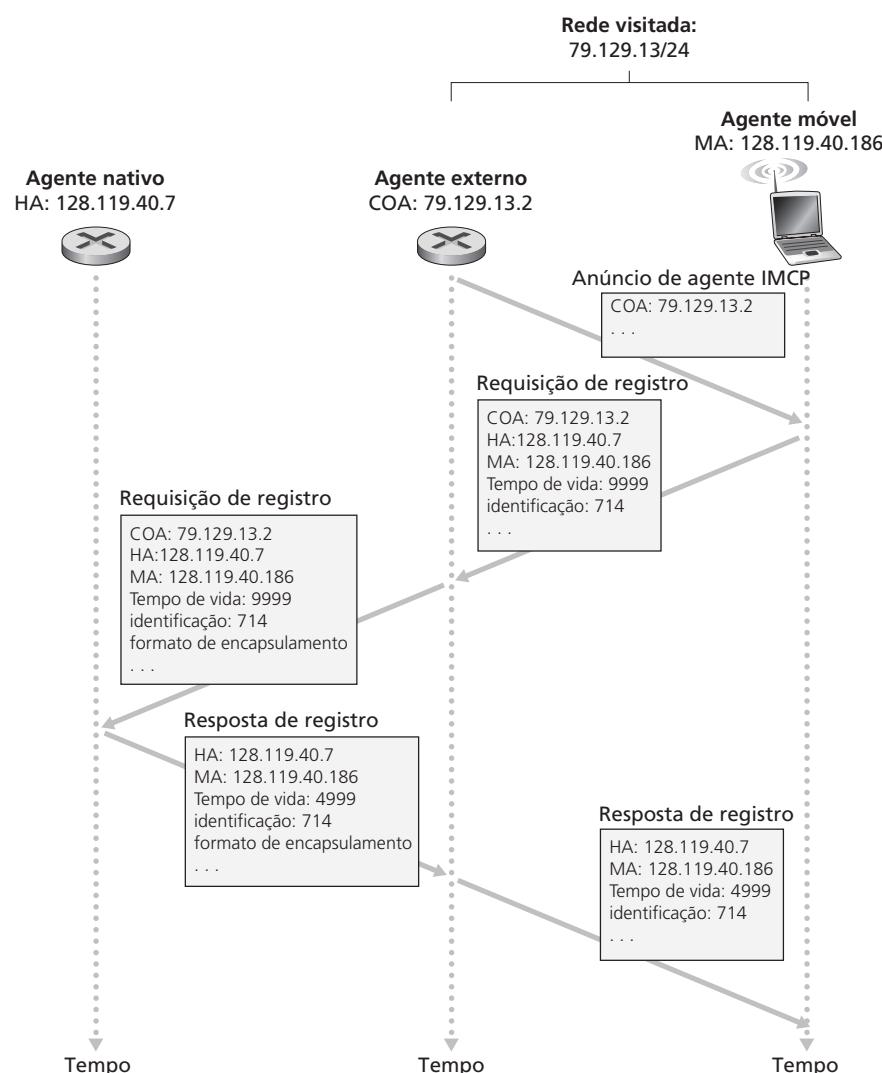


Figura 6.28 Anúncio de agente e registro IP móvel

serviço celular). A PLMN visitada, que denominaremos simplesmente **rede visitada**, é a rede na qual o nó móvel está residindo correntemente.

Como no caso do IP móvel, as responsabilidades das redes nativas e visitadas são bastante diferentes.

A rede nativa mantém um banco de dados conhecido como **registro nativo de localização** (*home location register* — HLR), que contém o número permanente do telefone celular e as informações do perfil do assinante para cada um de seus assinantes. O importante é que o HLR também contém informações sobre as localizações correntes desses assinantes. Isto é, se um usuário móvel estiver correntemente em trânsito pela rede celular de uma outra provedora, o HLR conterá informações suficientes para obter (por meio de um processo que descreveremos em breve) um endereço na rede visitada para o qual deverá ser roteada uma chamada ao usuário móvel. Como veremos, quando é feita uma chamada para um usuário móvel, um comutador especial na rede nativa, conhecido como **Central de Comutação para Portal de Serviços Móveis** (*Gateway Mobile Services Switching Center* — GMSC), é contatado por um correspondente. Mais uma vez, conforme nossa decisão de evitar a sopa de letrelinhas, denominaremos a GMSC por um termo mais descritivo, uma **MSC nativa**.

A rede visitada mantém um banco de dados conhecido como **registro de localização de visitantes** (*visitor location register* — VLR). O VLR contém um registro para cada usuário móvel que está

correntemente na porção da rede atendida pelo VLR. Assim, os registros do VLR vêm e vão à medida que usuários entram e saem da rede. Um VLR usualmente está localizado juntamente com a central de comutação móvel (MSC) que coordena o estabelecimento de uma chamada de e para a rede visitada.

Na prática, a rede celular de uma provedora servirá como uma rede nativa para seus assinantes e como uma rede visitada para usuários móveis que são assinantes de outra provedora de serviços celulares.

6.7.1 Roteando chamadas para um usuário móvel

Agora estamos prontos para descrever como é estabelecida uma chamada para um usuário GSM em uma rede visitada. Consideraremos um exemplo simples, representado na Figura 6.24; cenários mais complexos são descritos em [Mouly, 1992]. As etapas, como ilustradas na Figura 6.29, são as seguintes:

1. O correspondente disca o número do telefone do usuário móvel. Esse número, em si, não se refere a uma determinada linha telefônica ou localização (afinal, o número do telefone é fixo, mas o usuário é móvel!). Os dígitos iniciais do número são suficientes para identificar globalmente a rede nativa do usuário móvel. A chamada é roteada desde o correspondente, passa através do PSTN e chega até a MSC na rede nativa do usuário móvel. Esse é o primeiro trecho da chamada.
2. A MSC nativa recebe a chamada e interroga o HLR para determinar a localização do usuário móvel. No caso mais simples, o HLR retorna o **número roaming da estação móvel** (*mobile station roaming number* — MSRN), que denominaremos **número de roaming**. Note que esse número é diferente do número permanente do telefone móvel, que é associado com a rede nativa do usuário móvel. O número de roaming é efêmero: é designado temporariamente a um usuário móvel quando ele entra em uma rede visitada. O número de roaming desempenha um papel semelhante ao do endereço administrado em IP móvel e, tal como o COA, é invisível para o correspondente e para o usuário móvel. Se o HLR não tiver o número de roaming, ele retornará o endereço do VLR na rede visitada. Nesse caso (que não é mostrado na Figura 6.29), a MSC nativa precisará consultar o VLR para obter o número de roaming do nó móvel. Mas, antes de mais nada, como o HLR obtém o número de roaming ou o endereço VLR? O que acontece a esses valores quando o usuário móvel passa para uma outra rede visitada? Em breve consideraremos essas perguntas importantes.

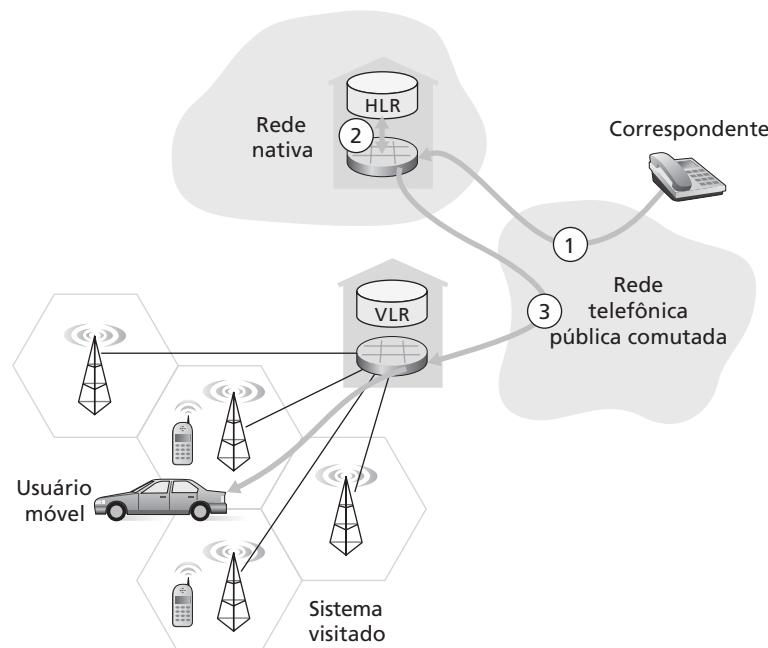


Figura 6.29 Estabelecendo uma chamada para um usuário móvel: roteamento indireto

3. Dado o número de roaming, a MSC nativa estabelece o segundo trecho da chamada através da rede até a MSC na rede visitada. A chamada está concluída — foi roteada do correspondente até a MSC nativa e daí para a MSC visitada, e desta até a estação-base que atende o usuário móvel.

Uma questão não resolvida na etapa 2 é como o HLR obtém informação sobre a localização do usuário móvel. Quando um telefone móvel é ligado ou entra em uma parte de uma rede visitada que é coberta por um novo VLR, ele deve se registrar na rede visitada. Isso é feito por meio da troca de mensagens de sinalização entre o usuário móvel e o VLR. O VLR visitado, por sua vez, envia uma mensagem de requisição de atualização de localização ao HLR do usuário móvel. Essa mensagem informa o HLR do número de roaming no qual o usuário móvel pode ser contatado ou o endereço do VLR (que então pode ser consultado mais tarde para obter o número do usuário móvel). Como parte dessa troca, o VLR também obtém do HLR informações sobre o assinante do usuário móvel e determina quais serviços (se houver algum) devem ser prestados ao usuário móvel pela rede visitada.

6.7.2 Transferências em GSM (handoffs)

Uma **transferência** (handoff) ocorre quando uma estação móvel muda sua associação de uma estação-base para outra durante uma chamada. Como mostra a Figura 6.30, uma chamada de telefone móvel é roteada inicialmente (antes da transferência) para o usuário móvel por meio de uma estação-base (a qual denominaremos estação-base antiga) e, após a transferência, por meio de uma outra estação-base (a qual denominaremos nova estação). Note que uma transferência entre estações-base resulta não somente em transferência/recepção de/para um telefone móvel e uma nova estação-base, mas também no rerroteamento da chamada em curso de um ponto de comutação dentro da rede para a nova estação-base. Vamos admitir inicialmente que as estações-base antiga e nova compartilham a mesma MSC e que o rerroteamento ocorre nessa MSC.

Há diversas razões possíveis para ocorrer transferência, incluindo (1) o sinal entre a estação-base corrente e o usuário móvel pode ter-se deteriorado a tal ponto que a chamada corre perigo de ser descartada e (2) uma célula pode ter ficado sobrecarregada, manipulando um grande número de chamadas. Esse congestionamento pode ser aliviado transferindo usuários móveis para células próximas menos congestionadas.

Enquanto está associado com uma estação-base, um usuário móvel mede periodicamente a potência de um sinal de sinalização emitido por sua estação-base corrente, bem como de sinais de sinalização emitidos por estações-base próximas que ele pode ‘ouvir’. Essas medições são passadas uma ou duas vezes por segundo para a estação-base corrente do usuário móvel. A transferência em GSM é iniciada pela estação-base antiga com base nessas medições, nas cargas correntes de usuários móveis em células próximas e outros fatores [Mouly, 1992]. O padrão GSM não determina o algoritmo específico a ser utilizado por uma estação-base para decidir se realiza ou não uma transferência.

A Figura 6.31 ilustra as etapas envolvidas quando uma estação-base decide transferir um usuário móvel:

1. A antiga estação-base (BS) informa à MSC visitada que deve ser feita uma transferência e a BS (ou possível conjunto de BSs) para a qual o usuário móvel deve ser transferido.

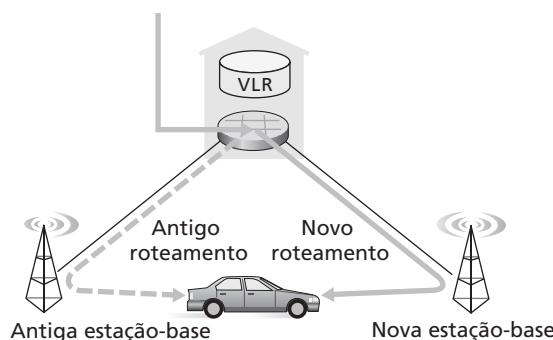


Figura 6.30 Cenário de transferência entre estações-base que têm uma MSC em comum

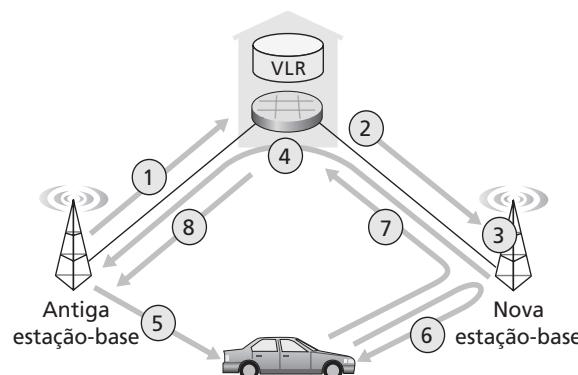


Figura 6.31 Etapas da execução de uma transferência entre estações-base que têm uma MSC em comum

2. A MSC visitada inicia o estabelecimento do caminho até a nova BS, alocando os recursos necessários para carregar a chamada rerroteada e sinalizando à nova BS que uma transferência está prestes a ocorrer.
3. A nova BS reserva e ativa um canal de rádio para ser utilizado pelo usuário móvel.
4. A nova BS devolve um sinal à MSC visitada e à antiga BS, indicando que foi estabelecido um caminho entre a MSC visitada e a nova BS e que o usuário móvel deve ser informado da transferência iminente. A nova BS provê todas as informações de que o usuário móvel necessitará para se associar com a nova BS.
5. O usuário móvel é informado de que deve realizar uma transferência. Note que, até esse ponto, o usuário móvel está alegremente inconsciente de que a rede estava preparando o terreno para uma transferência (por exemplo, reservando um canal na nova BS e alocando um caminho entre a MSC visitada e a nova BS).
6. O usuário móvel e a nova BS trocam uma ou mais mensagens para ativar totalmente o novo canal na nova BS.
7. O móvel envia à nova BS uma mensagem de conclusão de transferência que é transmitida até a MSC visitada. Então, a MSC visitada rerroteia a chamada em curso para o usuário móvel, via a nova BS.
8. Os recursos reservados ao longo do caminho até a antiga BS são liberados.

Vamos concluir nossa discussão sobre transferência considerando o que acontece quando o usuário móvel passa para uma BS que está associada com uma MSC *diferente* da antiga BS e o que acontece quando essa transferência interMSCs ocorre mais de uma vez. Como mostra a Figura 6.32, o GSM define a noção de uma **MSC âncora**. A MSC âncora é a MSC visitada pelo usuário móvel logo no início de uma chamada; assim, a MSC âncora não muda durante a chamada. Por toda a duração da chamada e independentemente do número de transferências interMSCs realizadas pelo usuário móvel, a chamada é roteada desde a MSC nativa até a MSC âncora e, então, da MSC âncora até a MSC visitada, onde o usuário móvel está correntemente localizado. Quando um usuário móvel passa da área de cobertura de uma MSC para a área de cobertura de uma outra MSC, a chamada em curso é rerroteada desde a MSC âncora até a nova MSC visitada, que contém a nova estação-base. Assim, durante o tempo todo há, no máximo, três MSCs (a MSC local, a MSC âncora e a MSC visitada) entre o correspondente e o usuário móvel. A Figura 6.32 ilustra o roteamento de uma chamada entre as MSCs visitadas por um usuário móvel.

Em vez de manter um único salto desde a MSC âncora até a MSC corrente, uma abordagem alternativa seria simplesmente encadear as MSCs visitadas pelo móvel, fazendo com que uma MSC antiga transmitisse a chamada em curso até a nova MSC cada vez que o móvel passasse para uma nova MSC. Esse encadeamento de MSC pode realmente ocorrer em redes celulares IS-41, com uma etapa opcional de minimização de caminho para remover MSCs entre a MSC âncora e a nova MSC visitada [Lin, 2001].

Vamos encerrar nossa discussão sobre o gerenciamento da mobilidade GSM fazendo uma comparação entre gerenciamento de mobilidade em GSM e em IP Móvel. A comparação apresentada na Tabela 6.2 indica que, embora redes IP e celulares sejam fundamentalmente diferentes sob muitos aspectos, compartilham um número surpreendente de elementos funcionais comuns e abordagens gerais para o gerenciamento da mobilidade.

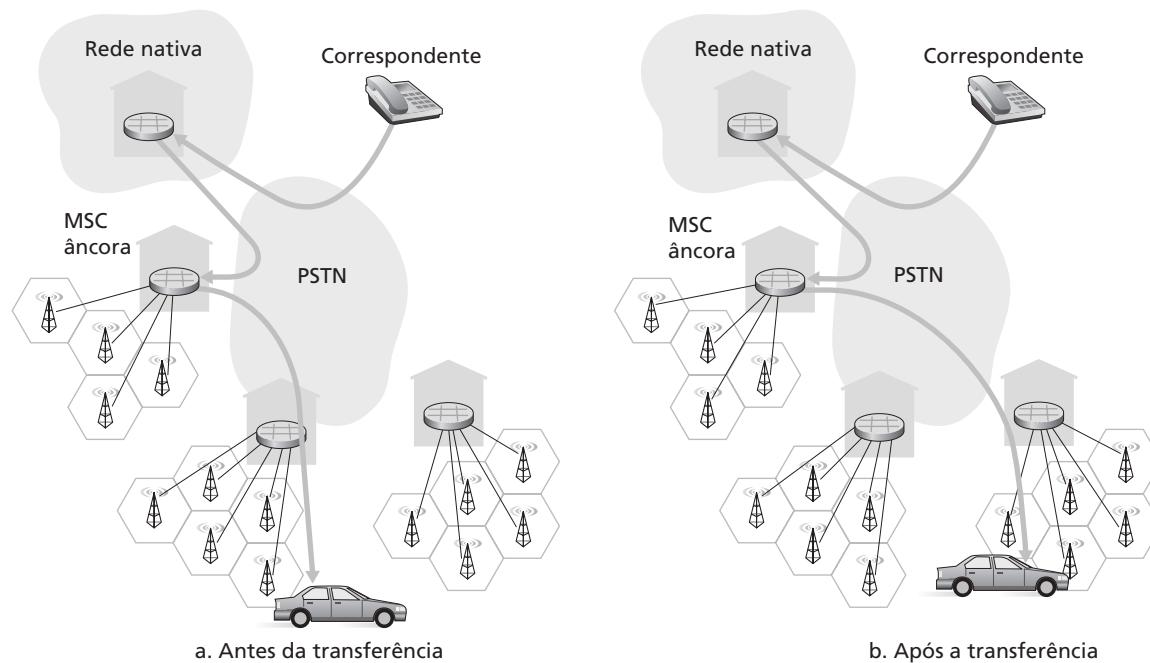


Figura 6.32 Reroteamento via a MSC âncora

6.8 Sem fio e mobilidade: impacto sobre protocolos de camadas superiores

Neste capítulo, vimos que redes sem fio são significativamente diferentes de suas contrapartes cabeadas tanto na camada de enlace (como resultado de características de canais sem fio como desvanecimento, propagação multivias e terminais ocultos) quanto na camada de rede (como resultado de usuários móveis que mudam seus pontos de conexão com a rede). Mas há diferenças importantes nas camadas de transporte e de aplicação? É tentador pensar que essas diferenças seriam minúsculas, visto que a camada de rede provê o mesmo modelo de serviço de entrega de melhor esforço às camadas superiores tanto em redes cabeadas quanto em redes sem fio. De modo semelhante, se protocolos como TCP ou UDP são usados para prover serviços de camada de transporte a aplicações tanto em redes cabeadas como em redes sem fio, então a camada de aplicação também deve permanecer

Elemento do GSM	Comentário sobre o elemento do GSM	Elemento do IP móvel
Sistema nativo	Rede à qual pertence o número de telefone permanente do usuário	Rede nativa
Central de comutação de unidade móvel ou simplesmente MSC nativa, Registro nativo de localização (HLR)	MSC nativa: ponto de contato para obter endereço roteável de usuário móvel. HLR: banco de dados no sistema nativo que contém número de telefone permanente, informações de perfil, localização corrente de usuário móvel, informações de assinatura	Agente nativo
Sistema visitado	Rede, exceto o sistema nativo, onde o usuário móvel está residindo correntemente	Rede visitada
Central de serviços de comutação de unidade móvel visitada, Registro de Localização de Visitante (VLR)	MSC visitada: responsável por estabelecer chamadas de/para nós móveis em células associadas com MSC. VLR: registro temporário em banco de dados em sistema visitado, contendo informações de assinatura para cada usuário móvel visitado	Agente externo
Número de roaming de estação móvel (MSRN) ou simplesmente número de roaming	Endereço roteável para segmento de chamada telefônica entre MSC nativa e MSC visitada, que não é visível nem para o usuário móvel nem para o correspondente	Endereço administrado

Tabela 6.2 Semelhanças entre a mobilidade em IP móvel e em GSM

sem mudança. Nossa intuição está certa em um sentido — TCP e UDP podem operar, e realmente operam, em redes com enlaces sem fio. Por outro lado, protocolos de transporte em geral e o TCP em particular às vezes podem ter desempenhos muito diferentes em redes cabeadas e em redes sem fio, e é neste particular, em termos de desempenho, que as diferenças se manifestam. Vamos ver por quê.

Lembre-se de que o TCP retransmite um segmento que é perdido ou corrompido no caminho entre remetente e destinatário. No caso de usuários móveis, a perda pode resultar de congestionamento de rede (transbordamento de buffer de roteador) ou de transferência (por exemplo, de atrasos no rerroteamento de segmentos para um novo ponto de conexão do usuário à rede). Em todos os casos, o ACK receptor-emissor do TCP indica somente que um segmento não foi recebido intacto; o remetente não sabe se o segmento foi perdido devido a congestionamento, durante transferência, ou devido a erros de bits detectados. Em todos os casos, a resposta do remetente é a mesma — retransmitir o segmento. A resposta do controle de congestionamento do TCP também é a mesma em todos os casos — o TCP reduz sua janela de congestionamento, como discutimos na Seção 3.7. Reduzindo incondicionalmente sua janela de congestionamento, o TCP admite implicitamente que a perda de segmento resulta de congestionamento e não de corrupção ou transferência de usuário. Vimos na Seção 6.2 que erros de bits são muito mais comuns em redes sem fio do que em redes cabeadas. Quando ocorrem esses erros de bits ou quando ocorre perda na transferência de usuário, na realidade não há nenhuma razão para que o remetente TCP reduza sua janela de congestionamento (reduzindo, assim, sua taxa de envio). Na verdade, é bem possível que os buffers de roteadores estejam vazios e que pacotes estejam fluindo ao longo de caminhos fim a fim desimpedidos, sem congestionamento.

Entre o início e meados da década de 1990, pesquisadores perceberam que, dadas as altas taxas de erros de bits em enlaces sem fio e a possibilidade de perdas devido à transferência de usuários, a resposta do controle de congestionamento do TCP poderia ser problemática em um ambiente sem fio. Há duas classes gerais de abordagens possíveis para tratar esse problema:

Recuperação local. Os protocolos de recuperação local recuperam erros de bits quando e onde (por exemplo, no enlace sem fio) eles ocorrem (por exemplo, o protocolo ARQ 802.11, que estudamos na Seção 6.3, ou abordagens mais sofisticadas que utilizam ARQ e também FEC [Ayanoglu, 1995]);

Remetente TCP ciente de enlaces sem fio. Em abordagens de recuperação local, o remetente TCP fica alegremente inconsciente de que seus segmentos estão atravessando um enlace sem fio. Uma abordagem alternativa é o remetente e o receptor ficarem cientes da existência de um enlace sem fio, para distinguir entre perdas por congestionamento que ocorrem na rede cabeada e corrupção/perdas que ocorrem no enlace sem fio e invocar controle de congestionamento somente em resposta a perdas por congestionamento na rede cabeada. [Balakrishnan, 1995] investiga vários tipos de TCP, admitindo que sistemas finais possam fazer essa distinção. [Wei, 2004] investiga técnicas para distinguir entre perdas nos segmentos cabeados e sem fio de um caminho fim a fim.

Abordagens de conexão dividida. Nesta abordagem [Bakre, 1995], a conexão fim a fim entre o usuário móvel e o outro ponto terminal é dividida em duas conexões da camada de transporte: uma do hospedeiro móvel ao ponto de acesso sem fio, e uma do ponto de acesso sem fio ao outro ponto terminal de comunicação (admitiremos, aqui, um usuário cabeado). A conexão fim a fim é, então, formada por uma concatenação de uma parte sem fio e uma parte cabeada. A camada de transporte sobre um segmento sem fio pode ser uma conexão padrão TCP [Bakre, 1995], ou principalmente um protocolo de recuperação de erro personalizado em uma do UDP. [Yavatkar, 1994] analisa o uso de um protocolo de repetição seletiva da camada de transporte através de uma conexão sem fio. As medidas relatadas em [Wei, 2006] indicam que conexões TCP divididas são amplamente usadas em redes de dados celulares, e que podem ser feitos aperfeiçoamentos significativos através do uso dessas conexões.

Aqui, nosso tratamento do TCP em enlaces sem fio foi necessariamente breve. Aconselhamos o leitor a consultar as referências se quiser mais detalhes sobre essa área de pesquisa.

Agora que já consideramos protocolos de camada de transporte, vamos considerar em seguida o efeito do sem fio e da mobilidade sobre protocolos de camada de aplicação. Uma consideração importante aqui é que enlaces sem fio muitas vezes têm larguras de banda relativamente baixas, como vimos na Figura 6.2. Consequentemente, aplicações que operam por enlaces sem fio, particularmente por enlaces celulares sem fio, devem tratar a largura

de banda como uma mercadoria escassa. Por exemplo, um servidor Web que serve conteúdo a um browser Web que está rodando em um telefone 3G provavelmente não conseguirá prover o mesmo conteúdo rico em imagens que oferece a um browser que está rodando sobre uma conexão cabeada. Embora enlaces sem fio proponham desafios na camada de aplicação, a mobilidade que eles criam também torna possível um rico conjunto de aplicações dependentes de localização e cientes de contexto [Chen, 2000]. Em termos mais gerais, redes sem fio e redes móveis desempenharão um papel fundamental na concretização dos ambientes de computação ubíquos do futuro [Weiser, 1991]. É justo dizer que vimos somente a ponta do iceberg quando se trata do impacto de redes sem fio e móveis sobre aplicações em rede e seus protocolos!

6.9 Resumo

Redes sem fio e móveis revolucionaram a telefonia e também estão causando um impacto cada vez mais profundo no mundo das redes de computadores. Com o acesso à infraestrutura da rede global que oferecem — desimpedido, a qualquer hora, em qualquer lugar — estão não somente aumentando a ubiquidade do acesso a redes, mas também habilitando um novo conjunto muito interessante de serviços dependentes de localização. Dada a crescente importância das redes sem fio e móveis, este capítulo focalizou os princípios, as tecnologias de enlace e as arquiteturas de rede para suportar comunicações sem fio e móveis.

Iniciamos este capítulo com uma introdução às redes sem fio e móveis, traçando uma importante distinção entre os desafios propostos pela natureza *sem fio* dos enlaces de comunicação desse tipo de rede, e pela *mobilidade* que esses enlaces permitem. Isso nos permitiu isolar, identificar e dominar melhor os conceitos fundamentais em cada área. Focalizamos primeiramente a comunicação sem fio, considerando as características de um enlace sem fio na Seção 6.2. Nas seções 6.3 e 6.4 examinamos os aspectos de camada de enlace do padrão IEEE 802.11 para LANs sem fio (Wi-Fi), do padrão WiMAX 802.16, do padrão Bluetooth 802.15.1 e do acesso celular à Internet. Então, voltamos nossa atenção para a questão da mobilidade. Na Seção 6.5 identificamos diversas formas de mobilidade e verificamos que há pontos ao longo desse espectro que propõem desafios diferentes e admitem soluções diferentes. Consideramos os problemas de localização e roteamento para um usuário móvel, bem como abordagens para transferir o usuário móvel que passa dinamicamente de um ponto de conexão com a rede para outro. Examinamos como essas questões foram abordadas no padrão IP móvel e em GSM nas seções 6.6 e 6.7, respectivamente. Finalmente, na Seção 6.8 consideramos o impacto causado por enlaces sem fio e pela mobilidade sobre protocolos de camada de transporte e aplicações em rede.

Embora tenhamos dedicado um capítulo inteiro ao estudo de redes sem fio e redes móveis, seria preciso todo um livro (ou mais) para explorar completamente esse campo tão animador e que está se expandindo tão rapidamente. Aconselhamos o leitor a se aprofundar mais nesse campo consultando as muitas referências fornecidas neste capítulo.



Exercícios de fixação

Capítulo 6 Questões de revisão

Seção 6.1

- O que significa para uma rede sem fio estar operando no “modo de infraestrutura”? Se a rede não estiver no modo de infraestrutura, em qual modo ela está e qual é a diferença entre esse modo de operação e o modo de infraestrutura?
- Quais são os quatro tipos de redes sem fio identificadas em nossa taxonomia na Seção 6.1? Quais desses tipos de redes sem fio você utilizou?

Seção 6.2

- Quais são as diferenças entre os seguintes tipos de falhas no canal sem fio: atenuação de percurso, propagação multivias, interferência de outras fontes?
- Um nó móvel se distancia cada vez mais de uma estação-base. Quais são as duas atitudes que uma estação-base poderia tomar para garantir que a probabilidade de perda de um quadro transmitido não aumente?



Seção 6.3

5. Descreva o papel dos quadros de sinalização em 802.11.
6. Verdadeiro ou falso: antes de uma estação 802.11 transmitir um quadro de dados, ela deve primeiramente enviar um quadro RTS e receber um quadro CTS correspondente.
7. Por que são usados reconhecimentos em 802.11, mas não em Ethernet cabeada?
8. Falso ou verdadeiro: Ethernet e 802.11 usam a mesma estrutura de quadro.
9. Descreva como funciona o limiar RTS.
10. Suponha que os quadros RTS e CTS IEEE 802.11 fossem tão longos quanto os quadros padronizados DATA e ACK. Haveria alguma vantagem em usar os quadros CTS e RTS? Justifique sua resposta.
11. A Seção 6.3.4 discute mobilidade 802.11, na qual uma estação sem fio passa de um BSS para outro dentro da mesma sub-rede. Quando os APs estão interconectados com um comutador, um AP pode precisar enviar um quadro com um endereço MAC fingido para fazer com que o comutador transmita quadros adequadamente. Por quê?
12. Quais são as diferenças entre o dispositivo mestre em uma rede Bluetooth e uma estação-base em uma rede 802.11?
13. Falso ou verdadeiro: Em WiMAX, a estação-base deve transmitir a todos os nós a uma mesma taxa de canal.
14. Qual o significado de “programação oportunista” em WiMAX?
15. Aprendemos na Seção 6.3.2 que há dois padrões 3G importantes: UMTS e CDMA-2000. A quais padrões

2G e 2,5G cada um desses dois padrões deve sua linhagem?

Seção 6.5 a 6.6

16. Se um nó tem uma conexão sem fio à Internet, esse nó precisa ser móvel? Explique. Suponha que um usuário portando um laptop ande com ele por sua casa, e sempre acesse a Internet por meio do mesmo ponto de acesso. Em relação à rede, este é usuário móvel? Explique.
17. Qual é a diferença entre um endereço permanente e um endereço aos cuidados de (COA)? Quem determina o endereço aos cuidados de?
18. Considere uma conexão TCP através de um IP Móvel. Falso ou verdadeiro: a fase da conexão TCP entre o correspondente e o hospedeiro móvel percorre a rede doméstica móvel, mas a fase de transferência de dados está diretamente entre o correspondente e o hospedeiro móvel, pulando a rede doméstica,

Seção 6.7

19. Quais são os objetivos do HRL e VLR nas redes GSM? Quais elementos de IP móvel são semelhantes ao HLR e ao VLR?
20. Qual é o papel da MSC âncora em redes GSM?

Seção 2.8

21. Quais são os métodos que podem ser realizados para evitar que um único enlace sem fio reduza o desempenho de uma conexão TCP fim a fim da camada de transporte?



Problemas

1. Considere o exemplo do remetente CDMA único na Figura 6.5. Qual seria a saída do remetente (para os 2 bits de dados mostrados) se o código do remetente CDMA fosse $(1, -1, 1, -1, 1, -1, 1, -1)$?
2. Considere o remetente 2 na Figura 6.6. Qual é a saída do remetente para o canal (antes de ser adicionada ao sinal vindo do remetente 1) $Z_{i,m}^2$?
3. Suponha que o receptor na Figura 6.6 queira receber os dados que estão sendo enviados pelo remetente 2. Mostre (por cálculo) que o receptor pode, na verdade, recuperar dados do remetente 2 do sinal agregado do canal usando o código do remetente 2.
4. Para o exemplo sobre dois emissores, dois receptores, dê um exemplo de dois códigos CDMA, que con-

têm os valores 1 e -1 , que não permitem que dois receptores extraiam os bits originais transmitidos por dois emissores CDMA.

5. Suponha que dois ISPs fornecem acesso Wi-Fi em um determinado café, e que cada um deles opera seu próprio AP e tem seu próprio bloco de endereços IP.
 - a. Suponha ainda mais, que, por acidente, cada ISP configurou seu AP para operar no canal 11. O protocolo 802.11 falhará totalmente nessa situação? Discuta o que acontece quando duas estações, cada uma associada com um ISP diferente, tentam transmitir ao mesmo tempo.
 - b. Agora suponha que um AP opera no canal 1 e outro no canal 11. Como suas respostas diferem uma da outra?

6. Na etapa 4 do protocolo CSMA/CA, uma estação que transmite um quadro com sucesso inicia o protocolo CSMA/CA para um segundo quadro na etapa 2, e não na etapa 1. Quais seriam as razões que os projetistas do CSMA/CA provavelmente tinham em mente para fazer com que essa estação não transmitisse o segundo quadro imediatamente (se o canal fosse percebido como ocioso)?
7. Suponha que uma estação 802.11b seja configurada para sempre reservar o canal com a sequência RTS/CTS. Suponha que essa estação repentinamente queira transmitir 1.000 bytes de dados e que todas as outras estações estão ociosas nesse momento. Calcule o tempo requerido para transmitir o quadro e receber o reconhecimento como uma função de SIFS e DIFS, ignorando atraso de propagação e admitindo que não haja erros de bits.
8. Considere o cenário mostrado na Figura 6.33, no qual existem quatro nós sem fios, A, B, C e D. A cobertura de rádio dos quatro nós é mostrada pelas formas ovais mais escuras; todos os nós compartilham a mesma frequência. Quando A transmite, ele pode ser ouvido/recebido por B; quando B transmite, ele só pode ser ouvido/recebido por A e C; quando C transmite, B e D podem ouvir/receber de C; quando D transmite, somente C pode ouvir/receber de D.

Agora suponha que cada nó possua um estoque infinito de mensagens que ele queira enviar para cada um dos outros nós. Se o destinatário da mensagem não for um vizinho imediato, então a mensagem deve ser retransmitida. Por exemplo, se A quer enviar para D, uma mensagem de A deve ser primeiro enviada a B, que, então, envia a mensagem a C, e este a D. O tempo é dividido em intervalos, com um tempo de transmissão de mensagem de exatamente um intervalo de tempo, como em um slotted Aloha, por exemplo. Durante um intervalo, um nó pode fazer uma das seguintes opções: (i) enviar uma mensagem; (ii) receber uma mensagem (se, exatamente, uma mensagem estiver sendo enviada a ele), (iii) permanecer silencioso. Como sempre, se um nó ouvir duas ou mais transmissões simultâneas, ocorrerá uma colisão e nenhuma das mensagens transmitidas é recebida com sucesso. Você pode admitir aqui que não existem erros de bits e, dessa forma, se uma mensagem for enviada, ela será recebida corretamente pelos que estão dentro do rádio de transmissão do emissor.

- a. Suponha que um controlador onisciente (ou seja, um controlador que sabe o estado de cada nó na rede) possa comandar cada nó a fazer o que ele (o controlador onisciente) quiser, ou seja, enviar uma mensagem, receber uma mensagem,

ou permanecer silencioso. Dado esse controlador onisciente, qual é a taxa máxima a qual uma mensagem de dados pode ser transferida de C para A, sabendo que não existem outras mensagens entre nenhuma outra dupla remetente/destinatária?

- b. Suponha que A envie uma mensagem a B, e D envie uma mensagem a C. Qual é a taxa máxima combinada à qual as mensagens de dados podem fluir de A a B e de D a C?
- c. Suponha agora que A envie uma mensagem a B, e C envie uma mensagem a D. Qual é a taxa máxima combinada à qual as mensagens de dados podem fluir de A a B e de C a D?
- d. Suponha agora que os enlaces sem fio sejam substituídos por enlaces cabeados. Repita as questões de "a" a "c" neste cenário cabeado.
- e. Agora suponha que estejamos novamente em um cenário sem fio e que para cada mensagem de dados enviada do remetente ao destinatário, este envie de volta uma mensagem ACK para o remetente (como no TCP, por exemplo). Suponha também que cada mensagem ACK possua um intervalo. Repita as questões de "a" a "c" para este cenário.

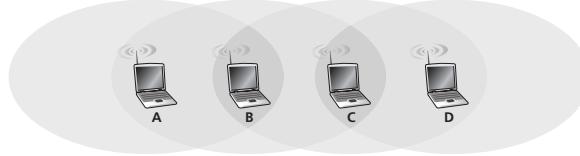


Figura 6.33 Cenário para o Problema 8

9. Descreva o formato do quadro Bluetooth 802.15.1. Você precisará de uma leitura complementar para encontrar essa informação. Existe algo no formato do quadro que basicamente limita o número de nós ativos para oito em uma rede 802.15.1? Explique.
10. Considere o seguinte cenário WiMAX ideal. O subquadro de downstream (veja Figura 6.17) é dividido em intervalos de tempo, com N intervalos downstream por subquadro e todos os intervalos de tempo têm o mesmo comprimento. Existem quatro nós, A, B, C e D, alcançáveis da estação-base a taxas de 10 Mbps, 5 Mbps, 2,5 Mbps e 1 Mbps, respectivamente no canal downstream. A estação-base possui infinitos dados para enviar a cada um dos nós e pode enviar para qualquer um dos quatro nós durante qualquer intervalo de tempo no subquadro de downstream.
 - a. Qual é a taxa máxima à qual a estação-base pode enviar aos nós, admitindo que ela pode enviar a qualquer nó de sua escolha durante cada intervalo de tempo? Sua solução é justa? Explique e defina o que você quis dizer com "justo".

- b.** Se há requisito de equidade que cada nós deve receber uma quantidade igual de dados durante cada quadro de downstream, qual é a taxa média de transmissão pela estação-base (para todos os nós) durante o subquadro de downstream? Explique como você chegou a esta resposta.
- c.** Suponha que, como critério de equidade, qualquer nó possa receber, no máximo, duas vezes tantos dados quanto qualquer outro nó durante o subquadro. Qual é a taxa média de transmissão pela estação-base (para todos os nós) durante o subquadro de downstream? Explique como você chegou a esta resposta.
- 11.** Na Seção 6.5, uma solução proposta que permitia que usuários móveis mantivessem seu endereço IP à medida que transitavam entre redes externas era fazer com que uma rede externa anunciasse ao usuário móvel uma rota altamente específica e usasse a infraestrutura de roteamento existente para propagar essa informação por toda a rede. Uma das preocupações que identificamos foi a escalabilidade. Suponha que, quando um usuário móvel passe de uma rede para outra, a nova rede externa anuncie uma rota específica para o usuário móvel e a antiga rede externa retire sua rota. Considere como informações de roteamento se propagam em um algoritmo vetor de distâncias (particularmente para o caso de roteamento interdomínios entre redes que abrangem o globo terrestre).
- a.** Outros roteadores conseguirão rotear datagramas imediatamente para a nova rede externa tão logo essa rede comece a anunciar sua rota?
- b.** É possível que roteadores diferentes acreditem que redes externas diferentes contenham o usuário móvel?
- c.** Discuta a escala temporal segundo a qual outros roteadores na rede finalmente aprenderão o caminho até os usuários móveis.
- 12.** Suponha que o correspondente na Figura 6.22 fosse móvel. Faça um desenho esquemático da infraestrutura adicional de camada de rede que seria necessária para rotear o datagrama do usuário móvel original até o correspondente (que agora é móvel). Mostre a estrutura do(s) datagrama(s) entre o usuário móvel original e o correspondente (agora móvel), como na Figura 6.23.
- 13.** Em IP móvel, que efeito terá a mobilidade sobre atrasos fim a fim de datagramas entre a fonte e o destino?
- 14.** Considere o exemplo de encadeamento discutido no final da Seção 6.7.2. Suponha que um usuário móvel visite as redes externas A, B e C, e que um correspondente inicie uma conexão com o usuário móvel enquanto este reside na rede externa A. Relacione a sequência de mensagens entre agentes externos e entre agentes externos e o agente nativo, enquanto o usuário passa da rede A para a rede B e para a rede C. Em seguida, suponha que não é executado encadeamento e que as mudanças no endereço administrado do usuário móvel devem ser notificadas explicitamente ao correspondente (bem como ao agente nativo). Relacione a sequência de mensagens que seria necessário trocar nesse segundo cenário.
- 15.** Considere dois nós móveis em uma rede externa que tem um agente externo. É possível que os dois nós móveis utilizem o mesmo endereço administrado em IP móvel? Explique sua resposta.
- 16.** Quando discutimos como o VLR atualizava o HLR com informações sobre a localização corrente de usuários móveis, quais eram as vantagens e as desvantagens de fornecer ao HLR o MSRN em vez do endereço do VLR?



Questões dissertativas

- 1.** Relacione cinco produtos existentes hoje no mercado que forneçam uma interface Bluetooth ou 802.15.
- 2.** O serviço 3G sem fio está disponível em sua região? Quando custa? Quais aplicações ele suporta?
- 3.** Que tipos de problemas você observou como usuário do IEEE 802.11? Como os projetos do 802.11 podem evoluir para superar esses problemas?
- 4.** Faça uma pesquisa na Web para se informar sobre os testes de implementação do WiMAX. Quão abrangentes foram esses testes? Quais vazões foram atingidas a quais distâncias? Para quantos usuários?
- 5.** Faça uma pesquisa na Web para se informar sobre a implementação de EVDO e HSDPA. Qual deles foi o mais empregado até agora? Em que lugar?



Wireshark Lab

No Companion Website deste livro, www.aw.com/kurose_br, você encontrará um Wireshark Lab, em inglês,

para este capítulo, que captura e estuda os quadros 802.11 trocados entre um laptop sem fio e um ponto de acesso.

Entrevista

Charlie Perkins

Charles E. Perkins é um técnico associado na WiChorus, que investiga novas técnicas para a aplicação dos protocolos de gerenciamento de mobilidade da Internet a novas gerações de mídia sem fio, como o WiMAX e LTE. Essas tecnologias já começaram a transformar a banda larga sem fio em realidade, surgindo novas oportunidades para expandir a Internet e fornecer conteúdo multimídia sob demanda. Ele é o editor de documento para o grupo de trabalho IP móvel da Internet Engineering Task Force (IETF) e é autor ou coautor de documentos de padrões para os grupos de trabalho mip4, mip6, manet, mext, dhc, seamoby (Seamless Mobility) e autoconf de grupos de trabalho. É editor de vários jornais do ACM e IEEE relacionados à rede sem fio. Na WiChorus, ele possui um envolvimento contínuo com desenvolvimentos avançados empregando Mobile IP, IPv6 e outros protocolos baseados no IETF.



Charles escreveu e editou livros sobre IP Móvel e Redes Ad Hoc e publicou vários estudos e artigos premiados nas áreas de redes móveis, redes ad hoc, otimização de rota para redes móveis, descoberta de recursos e configuração automática para computadores móveis. Ele é um dos criadores do MobiHoc, sendo o coordenador geral e o coordenador do comitê de programa. Charles participou do Conselho de Arquitetura da Internet do seu IETF e em vários comitês do National Research Council, bem como em diversos conselhos de avaliação técnica para o Army Research Lab e o programa Swiss MICS. Recentemente, ele foi o coordenador geral do workshop WiNS-DR e MASS 2006.

O que o fez se decidir pela especialização em sem fio/mobilidade?

Meu envolvimento com redes sem fio e mobilidade foi uma consequência natural do meu trabalho em projetos na IBM Research no final da década de 1980. Nós tínhamos enlaces de rádio e estávamos tentando construir um equipamento ao estilo do 'ThinkPad' (digamos, por exemplo, um Palm Pilot) com conectividade sem fio e reconhecimento de escrita.

Construímos uma solução simples (mais tarde denominada 'Mobile IP' (IP Móvel)) e percebemos que ela funcionava. Aproveitando nossa experiência com o Mobile IP, projetamos uma modificação rápida e efetiva para o RIP — e que também funcionou muito bem. Aqui, 'funcionar' quer dizer que as aplicações rodavam bem sem nenhuma modificação e que o desempenho da rede não foi prejudicado por nossos novos projetos. Essas propriedades ficaram conhecidas como 'transparência da aplicação' e 'escalabilidade'. É claro que funcionar em laboratório é muito diferente de alcançar sucesso comercial, e ambas as tecnologias têm muito potencial ainda não realizado comercialmente.

Qual foi seu primeiro emprego no setor de computação?

Trabalhei na TRW Controls, em Houston, Texas. Foi uma mudança drástica em relação ao estudo universitário. Uma coisa que aprendi na TRW Controls é como o suporte de software é precário até mesmo para os sistemas mais críticos de controle de utilidades. Esses sistemas deviam controlar o fluxo de energia elétrica em redes de enorme potência e o modo como era construído o software subjacente era de arrepia os cabelos. E mais, os cronogramas eram sempre apertados e os programadores, profundamente céticos em relação às suas condições de trabalho e às intenções da diretoria. Era preciso projetar o sistema inteiro a partir do zero. Não tenho muitas razões para achar que as coisas mudaram nesses últimos 30 anos, especialmente considerando os eventos recentes que cercaram o apagão de 2003. Na verdade, dada a desregulamentação, certamente ficaram até piores. Para mim foi uma felicidade sair da TRW Controls e passar para a Tektronix (Tek Labs).

Qual parte do seu trabalho lhe propõe mais desafios?

A parte do meu trabalho que me propõe mais desafios é entender o que devo fazer para ajudar minha empresa. E também considero como parte do meu trabalho moldar as tecnologias sem fio com as quais entro em contato e fornecer às pessoas serviços melhores e uma experiência diária mais agradável. Minha empresa trabalha no ramo de prover equipamento de infraestrutura para conectividade sem fio de alta velocidade. Além de desenvolvendo os documentos de padrão relevantes, espero encontrar maneiras de simplificar os sistemas resultantes aplicando várias técnicas relacionadas ao IETF. Fazer isso de uma maneira a também maximizar o potencial de benefício das tecnologias que desenvolvemos, cada dia se torna um desafio. Ainda há muito para ser feito e as oportunidades são imensas.

Em um nível técnico mais detalhado, no qual certamente me sinto mais à vontade, tento resolver problemas de protocolos de rede de modo a aplicar o mínimo de carga possível sobre os equipamentos sem fio (e suas baterias!) e provocar a mínima inconveniência aos usuários. Interconectar os aparelhos sem fio de hoje com a Internet por meio de tecnologias sem fio de alta velocidade é extremamente interessante tecnicamente e oferece potencial ilimitado de sucesso comercial para quem conseguir encontrar os caminhos certos para seguir em frente. Além disso, estamos começando a encarar um desafio notável já que nosso espaço de endereço IPv4 está no rumo certo de se esgotar em dois ou três anos. O IPv6 provou ser mais difícil de empregar do que o que foi previsto há dez anos, mesmo as especificações básicas estando completamente desenvolvidas.

Em sua opinião, qual é o futuro da tecnologia sem fio?

O setor, como um todo, está passando por tremendas mudanças e ainda não há nenhum final à vista. Estão surgindo novas tecnologias sem fio de alta velocidade que poderão causar efeitos práticos imprevisíveis e poderiam mudar fundamentalmente a sociedade. Nossas expectativas atuais em relação à privacidade e às limitações à nossa capacidade de nos comunicar uns com os outros (voz, imagem e dados) poderão ser irreconhecíveis dentro de dez anos. À medida que as empresas adotam cada vez mais comunicações sem fio, é bem possível que novas medidas de segurança sejam tomadas, o que mudaria significativamente nossa experiência no local de trabalho.

Parece bem claro que conseguiremos alocar mais espaço a vários esquemas de comunicações por rádio, cujas velocidades poderão ser muito altas. A minha expectativa é que logo será muito viável para as comunidades oferecer a seus cidadãos comunicações sem fio de alta velocidade completas, como se a cidade inteira fosse uma rede local. Isso poderia ter um efeito revigorante sobre o sentimento de comunidade que há muito tempo se perdeu em nossa sociedade, pelo menos nos Estados Unidos. É claro que a comunidade ainda demandaria acesso à Internet. A capacidade de armazenamento em disco está crescendo tão rápido, a preços tão econômicos que já podemos carregar no bolso o Wikipédia inteiro e provavelmente todos os números de telefone do mundo, sem mencionar os acervos pessoais inéditos de livros, músicas e filmes.

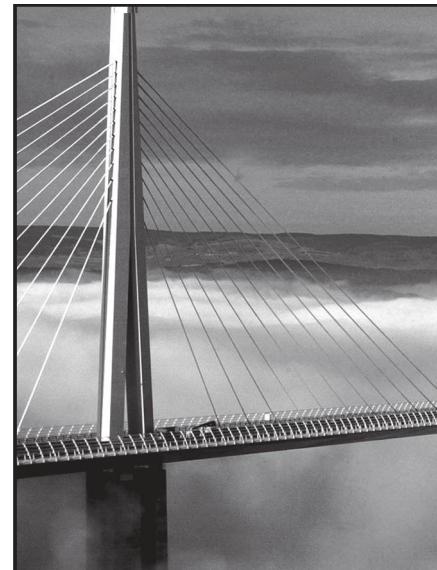
A tecnologia sem fio está acelerando o crescimento da Internet. À medida que equipamentos sem fio ficam mais baratos, as comunicações pela Internet estão por toda a parte (em plugues de ouvido, em jogos multiusuário, em leitoras de bilhetes de metrô), o que motivará novas aplicações e novas soluções de segurança.





Capítulo 7

Redes multimídia



Estamos todos testemunhando o desdobramento mundial de aplicações de vídeo e áudio na Internet. Milhares de sites — incluindo CNN, Rhapsody, Napster, MSN, AOL, Yahoo! — disponibilizam o envio de informação multimídia com conteúdo de áudio e vídeo. O YouTube e outros sites de compartilhamento de vídeos permitem que os usuários vejam — por pedido — videoclips que foram carregados por outros usuários. Milhões de usuários usam o Skype regularmente para suas necessidades relacionadas a telefonia e a video-conferência. E alguns canais tradicionais da televisão estão sendo transmitidos pela Internet, permitindo que usuários da Internet assistam os canais de televisão de todos os cantos do mundo. Esse crescimento explosivo das aplicações multimídia da Internet é, primariamente, resultado do aumento da penetração do acesso residencial a banda larga e do acesso sem fio de alta velocidade (como a WiFi). Como estudado na Seção 1.2, as taxas de acesso de banda larga continuarão a aumentar, e a possibilitar a distribuição de novas e empolgantes aplicações multimídias.

As exigências de funcionamento das aplicações multimídia divergem significativamente daquelas tradicionais aplicações elásticas, como e-mails, navegação na web, login remoto e download e compartilhamento de arquivo (as quais estudamos no Capítulo 2). Em particular, diferentemente das aplicações elásticas, as aplicações multimídia, são muito mais sensíveis a atrasos ponto a ponto e a variações de atrasos, mas podem tolerar casuais perdas de informação.

Começaremos esse Capítulo com uma taxonomia das aplicações multimídia na Seção 7.1. Veremos que uma aplicação pode ser classificada como *fluxo de áudio/vídeo armazenado*, *fluxo áudio/vídeo ao vivo* e *áudio/vídeo interativo de tempo real*. Mais adiante veremos que cada uma dessas classificações de aplicativos têm um conjunto de exigências de funcionamento para a rede. Na seção 7.2 examinaremos o fluxo de áudio/vídeo armazenado com mais atenção. Na Seção 7.3, analisaremos técnicas de níveis de aplicação que podem aprimorar o desempenho das aplicações multimídia nos melhores esforços de funcionamento de Internet atual, e na Seção 7.4 cobriremos os vários protocolos de multimídia usados na Internet atual. Na Seção 7.5 investigaremos os mecanismos dentro da rede que podem ser usados para diferenciar uma classificação de tráfego (por exemplo, aplicações que toleram atrasos, como as multimídia) de outras (por exemplo, aplicações elásticas, como FTP), e fornecer um funcionamento diferenciado entre várias classificações de tráfego. Finalmente, na Seção 7.6, levaremos em conta um caso onde a rede precisa fornecer *garantias* de desempenho a uma aplicação — por exemplo, uma ligação telefônica baseada em pacote IP terá o mesmo desempenho do que uma ligação feita de um circuito comutado de uma rede de telefonia. Veremos que isso precisará da introdução de novos mecanismos e protocolos de rede.



7.1 Aplicações de rede multimídia

Em nossa discussão sobre requisitos de serviço de aplicação no Capítulo 2, identificamos algumas linhas ao longo das quais esses requisitos podem ser classificados. Duas dessas linhas — considerações quanto à temporização e à tolerância à perda de dados — são particularmente importantes para aplicações de multimídia em rede. Considerações de temporização são importantes porque muitas aplicações de multimídia são altamente **sensíveis a atraso**. Veremos em breve que, em muitas aplicações de multimídia, pacotes que sofrem atrasos de remetente a destinatário de mais do que algumas centenas de milissegundos são essencialmente inúteis ao receptor. Por outro lado, aplicações de multimídia em rede são, em sua maioria, **tolerantes à perda** — perdas ocasionais causam somente pequenas perturbações na recepção de áudio e vídeo, e essas perdas podem ser parcial ou totalmente encobertas. Essas características de sensibilidade a atraso e de tolerância à perda são claramente diferentes daquelas das aplicações elásticas, como a Web, e-mail, FTP e Telnet. Para aplicações elásticas, atrasos longos são incômodos, mas não particularmente prejudiciais, e a completude e a integridade dos dados transferidos são de suma importância.

7.1.1 Exemplos de aplicações multimídia

A Internet pode comportar uma grande variedade de aplicações de multimídia interessantes. Nesta subseção, consideraremos três classes abrangentes de aplicações de multimídia: áudio e vídeo de fluxo contínuo armazenados, áudio e vídeo de fluxo contínuo ao vivo e áudio e vídeo interativos em tempo real.

Neste capítulo *não* consideraremos aplicações cujos dados são transportados antes e reproduzidos depois, tal como um arquivo MP3, que é descarregado completamente por meio de uma aplicação de compartilhamento de arquivos P2P antes de ser ouvido. Na verdade, aplicações desse tipo são elásticas, isto é, são aplicações de transferência de arquivos que não têm nenhum requisito especial quanto a atrasos. Examinamos transferência de arquivos (HTTP e FTP) e sistemas de compartilhamento de arquivos P2P no Capítulo 2.

Áudio e vídeo de fluxo contínuo armazenados

Nessa classe de aplicações, clientes requisitam, sob demanda, arquivos de áudio e vídeo comprimidos que estão armazenados em servidores. Milhares de páginas fornecem fluxos armazenados de áudio e vídeo hoje em dia, incluindo a CNN, a Microsoft Video e o YouTube. Essa classe de aplicações tem três características fundamentais:

Mídia armazenada. O conteúdo multimídia foi pré-gravado e está armazenado num servidor. Como a mídia é gravada antecipadamente, o usuário no cliente pode fazer pausa, voltar, avançar ou escolher itens no índice de conteúdo da gravação. O tempo decorrido entre a solicitação de uma dessas ações pelo cliente e a execução da ação deve ser da ordem de 1 a 10 segundos para que a resposta seja considerada aceitável.

Fluxo contínuo. Em uma aplicação de áudio/vídeo armazenado, normalmente o cliente inicia a reprodução do áudio/vídeo alguns segundos após começar a receber o arquivo do servidor. Isso significa que o cliente estará reproduzindo áudio/vídeo de uma parte do arquivo ao mesmo tempo que está recebendo do servidor partes do arquivo que estão mais à frente. Essa técnica, conhecida como **fluxo contínuo** (streaming), evita ter de descarregar o arquivo inteiro (e incorrer em atraso potencialmente longo) antes de começar a reproduzi-lo. Há muitos produtos para reprodução de multimídia de fluxo contínuo, entre eles o RealPlayer, da RealNetworks [RealNetworks, 2009], o QuickTime, da Apple [QuickTime, 2009] e o Windows Media, da Microsoft [Microsoft Windows Media Player, 2009].

Reprodução contínua. Assim que se inicia a reprodução do conteúdo da multimídia, ela deve prosseguir de acordo com a temporização original da gravação. Portanto, os dados devem ser recebidos do servidor a tempo de ser reproduzidos no cliente; caso contrário, os usuários podem ser vítimas de atrasos de buffer. Embora aplicações de mídia armazenada tenham requisitos de reprodução contínua, ainda assim suas limitações de atraso fim a fim são menos severas do que as limitações para aplicações ao vivo, interativas, como a telefonia por Internet e a videoconferência (veja a seguir).



História

IPTV

O conteúdo televisivo vem tradicionalmente sendo distribuído por micro-ondas terrestres fibra híbrida coaxial(HFC), e canais de sátelites geoestacionários (veja Seção 1.2). Mas na era da Internet atual, existe um interesse enorme no IPTV — isto é, distribuir conteúdo televisivo pela Internet.

Um dos desafios do IPTV é lidar com a imensa quantidade de banda passante necessária, particularmente na fonte do servidor. Por exemplo, considere a transmissão simultânea de um importante evento esportivo, como uma partida da Copa do Mundo, de um único servidor da Internet a 100 milhões de usuários. Se a velocidade do vídeo é, modestamente, de 1 Mbps, então a banda passante exigida pelo servidor seria de chocantes 100 terabits/seg! Sendo assim, a distribuição clássica servidor-cliente é inviável. Se um multicast IP fosse implementado através da Internet seria muito mais fácil para o IPTV se tornar realidade. Outra alternativa seria distribuir o vídeo através de uma rede multicast sobreposta, como aquelas que são oferecidas pelas redes de distribuição de conteúdo (Content Distribution Networks — CDNs)(Veja a Seção 7.3).

No entanto, outra opção é usar uma distribuição par-a-par, através do qual cada par que recebe um canal de televisão também ajuda na redistribuição do canal aos outros pares. Talvez o maior chamativo desse tipo de abordagem seja o preço baixo de distribuição: se os pares coletivamente fornecem uma largura de banda upstream suficiente, pouca largura de banda de rede será necessária (talvez poucos múltiplos da taxa do vídeo). Por ter um custo tão barato, qualquer pessoa com uma Webcam poderia distribuir um programa ao vivo a milhões de usuários por um preço ínfimo!

Até hoje, um determinado número de sistemas como o BitTorrent P2P IPTV desfrutaram de uma distribuição bem-sucedida. O pioneiro da área, o CoolStreaming, divulgou que teve mais de 4.000 usuários simultâneos em 2003 [CoolStreaming 2005]. Mais recentemente, outros sistemas, incluindo o PPLive e o ppstream, relataram êxito, com milhares de usuários simultâneos assistindo a canais com velocidades de 300 kbps e 1 Mbps. Nestes sistemas como o BitTorrent, pares formam uma rede dinâmica sobreposta e trocam blocos de vídeo com outras redes sobrepostas. Será interessante acompanhar como o IPTV se sairá nos próximos 5 ou 10 anos. Que tecnologia de sobreposição será usada: CDN ou P2P, ou um híbrido dos dois? E uma fração significativa dos fãs da Copa do Mundo assistirão às partidas da Copa de 2014 pela Internet?

Áudio e vídeo de fluxo contínuo ao vivo

Essa classe de aplicação é semelhante à transmissão tradicional de rádio e televisão, exceto que a transmissão é realizada pela Internet. Essas aplicações permitem que um usuário receba uma transmissão de rádio ou televisão ao vivo de qualquer parte do mundo (por exemplo, um dos autores deste livro sempre ouve suas emissoras de rádio favoritas da Filadélfia quando está viajando. O outro autor ouviu regularmente as transmissões ao vivo dos jogos de seu amado time de basquete universitário quando morou na França por um ano). Estas aplicações são frequentemente reconhecidas como um rádio da Internet e IPTV. Hoje existem milhares de estações de rádio sendo transmitidas pela Internet e um número significativo de implementações de IPTV (ver quadro complementar).

Como o fluxo contínuo de áudio e vídeo ao vivo não é armazenado, um cliente não pode adiantar o programa que está recebendo. Contudo, com armazenamento local de dados recebidos, outras operações interativas, como pausa e retrocesso São possíveis. No caso de aplicações ao vivo de transmissão broadcast, frequentemente há muitos clientes recebendo o mesmo programa de áudio/vídeo. A distribuição de áudio/vídeo ao vivo para muitos receptores pode ser realizada eficientemente utilizando as técnicas de multicasting IP que estudamos na Seção 4.7.No entanto, atualmente a distribuição de áudio/vídeo ao vivo é mais comumente realizada por meio de aplicações em camadas multicast (usando P2P ou CDN) através de fluxos unicast múltiplos separados servidor-cliente. Como acontece com a multimídia armazenada de fluxo contínuo, ela requer reprodução contínua, embora as limitações de temporização sejam menos severas do que para aplicações interativas em tempo real. Atrasos de até dezenas de segundos desde o instante em que o usuário requisita a entrega/reprodução de uma transmissão ao vivo até o início da reprodução podem ser tolerados.

Áudio e vídeo interativos em tempo real

Essa classe de aplicações permite que as pessoas usem áudio/vídeo para se comunicar entre si em tempo real. Áudio interativo em tempo real pela Internet é frequentemente denominado **telefone por Internet**, já que, da perspectiva do usuário, é semelhante ao tradicional serviço telefônico por comutação de circuitos. O telefone por Internet pode fornecer, potencialmente, serviços telefônicos de central privada de comutação telefônica (PBX), serviços locais e de longa distância, a custo muito baixo. Também pode facilitar a disponibilização de novos serviços que não são suportados com facilidade pelas redes tradicionais de comutação de circuitos, como detectores de presença, comunicação em grupo, integração telefone/Web, filtragem de chamadas e outros mais. Hoje, há muitos produtos de telefonia por Internet disponíveis. Por exemplo, usuários do Skype podem fazer chamadas por voz de computador para telefone e de computador para computador. Com vídeo interativo em tempo real, também denominado videoconferência, indivíduos se comunicam visualmente, assim como oralmente. Há também muitos produtos de vídeo interativo em tempo real disponíveis para a Internet, incluindo o NetMeeting da Microsoft, vídeo Skype, e vários produtos da Polycom. Note que, em uma aplicação de áudio/vídeo interativo em tempo real, um usuário pode falar ou se mover a qualquer instante. Para uma conversação com interação entre vários usuários, o atraso entre o momento em que um usuário fala ou se move e o momento em que a ação se manifesta nos hospedeiros receptores deve ser menor do que algumas centenas de milissegundos. No caso da voz, atrasos menores do que 150 milissegundos não são percebidos pelo ouvido humano, atrasos entre 150 e 400 milissegundos podem ser aceitáveis e atrasos que excedem 400 milissegundos podem resultar em conversas frustrantes, se não completamente ininteligíveis.

7.1.2 Obstáculos para a multimídia na Internet de hoje

Lembre-se de que o protocolo IP disponibilizado na Internet de hoje provê um **serviço de melhor esforço** a todos os datagramas que transporta. Em outras palavras, a Internet faz seu melhor esforço para transportar cada datagrama do remetente ao receptor o mais rapidamente possível, mas não faz nenhuma promessa sequer sobre o atraso fim a fim para um pacote individual. Tampouco faz promessas quanto à variação do atraso de pacote dentro de uma corrente de pacotes. Como TCP e UDP executam sobre IP, nenhum desses protocolos pode dar alguma garantia contra atraso às aplicações requisitantes. Devido à falta de qualquer esforço especial para entregar pacotes a tempo, é um problema extremamente desafiador desenvolver aplicações de rede multimídia de sucesso para a Internet. Contudo, a multimídia pela Internet alcançou um sucesso significativo até o momento, porém limitado. Por exemplo, a recepção em fluxo contínuo de áudio/ vídeo armazenado com atrasos de 5 a 10 segundos na interatividade com o usuário agora é comum na Internet. Mas, durante os horários de pico de tráfego, o desempenho pode ser insatisfatório, sobretudo quando os enlaces intervenientes estão congestionados (como os congestionados enlaces transoceânicos).

Telefone por Internet e vídeo interativo em tempo real também encontraram um uso amplo; por exemplo, existem normalmente sete milhões de usuários do Skype online em qualquer momento. Voz e vídeo interativos em tempo real impõem rígidas limitações ao atraso de pacote e à variação de atraso do pacote. **Variação de atraso** de pacote significa que os pacotes experimentam atrasos diferentes dentro da mesma corrente de pacotes. Voz e vídeo em tempo real funcionam bem em regiões onde a oferta de largura de banda é abundante e, portanto, o atraso e a variação de atraso são mínimos. Mas a qualidade pode se deteriorar até níveis inaceitáveis assim que a corrente de pacotes de voz e vídeo atinge um enlace moderadamente congestionado.

O projeto de aplicações de multimídia seria certamente menos complicado se houvesse algum tipo de classificação dos serviços na Internet em primeira e segunda classe, pela qual pacotes de primeira classe teriam número limitado e receberiam serviço prioritário em filas de roteadores. Esse serviço de primeira classe poderia ser satisfatório para aplicações sensíveis ao atraso. Mas, até agora, a Internet adotou preferencialmente uma postura igualitária em relação ao escalonamento de pacotes em filas de roteadores. Todos os pacotes recebem serviço igual; nenhum pacote, incluindo os pacotes de áudio e vídeo, sensíveis ao atraso, recebe prioridade especial nas filas de roteadores. Não importa quanto dinheiro você tenha ou quão importante você seja: você tem de se juntar ao fim da fila e esperar sua vez! Na segunda metade deste capítulo, examinaremos as arquiteturas propostas que visam a acabar com essa limitação.

Assim, por enquanto temos de viver com o serviço de melhor esforço. Mas, dada essa limitação, podemos tomar diversas decisões quanto ao projeto e utilizar algumas artimanhas para melhorar a qualidade de uma aplicação de rede multimídia percebida pelo usuário. Por exemplo, podemos enviar áudio e vídeo sobre UDP e, assim, evitar a baixa vazão do TCP quando este entra em sua fase de partida lenta. Podemos retardar a reprodução no receptor em 100 milissegundos ou mais para reduzir os efeitos da variação de atraso induzida pela rede. Podemos colocar marca de tempo nos pacotes no remetente, de modo que o receptor saiba quando eles devem ser reproduzidos. Para áudio/vídeo armazenados, podemos adquirir dados antecipadamente durante a reprodução, quando houver capacidade de arquivo e largura de banda extra disponíveis no cliente. Podemos até enviar informações redundantes para atenuar os efeitos da perda de pacotes induzida pela rede. Examinaremos muitas dessas técnicas no restante da primeira metade deste capítulo.

7.1.3 Como a Internet deveria evoluir para dar melhor suporte à multimídia?

Hoje, há um debate extraordinário — e, às vezes, feroz — sobre como a Internet deveria evoluir para melhor atender o tráfego multimídia com suas limitações rígidas de temporização. Em um extremo, alguns pesquisadores argumentam que devem ser feitas modificações fundamentais na Internet para que aplicações possam reservar explicitamente largura de banda fim a fim e dessa forma receber uma *garantia* em seu desempenho fim-fim. Uma garantia definitiva significa que a aplicação receberá a qualidade exigida de serviço (QoS) com certeza. Uma garantia branda significa que a aplicação terá grande probabilidade de receber sua qualidade exigida de serviço. Eles acham, por exemplo, que, se um usuário quiser fazer uma chamada telefônica pela Internet do hospedeiro A para o hospedeiro B, a aplicação de telefone por Internet do usuário deveria poder reservar largura de banda explicitamente em cada enlace ao longo da rota entre os dois hospedeiros. Mas permitir que as aplicações façam reservas e exigir que a rede honre essas reservas implica algumas grandes mudanças. Primeiramente, precisamos de um protocolo que reserve largura de banda de enlace no caminho entre os remetentes e seus receptores, em nome das aplicações. Em segundo lugar, devemos modificar políticas de escalonamento nas filas dos roteadores, de modo que as reservas de largura de banda possam ser honradas. Com essas novas políticas de escalonamento, nem todos os pacotes receberiam o mesmo tratamento; em vez disso, quem reserva (e pagar) mais, recebe mais. Em terceiro lugar, para honrar as reservas, as aplicações devem dar à rede uma descrição do tráfego que pretendem enviar para dentro dela. A rede deve, então, inspecionar o tráfego de cada aplicação para assegurar que ela cumpra o que descreveu. Finalmente, a rede deve dispor de meios para determinar se tem largura de banda suficiente para suportar qualquer nova solicitação de reserva. Esses mecanismos, quando combinados, exigem novos e complexos programas nos hospedeiros e roteadores, bem como novos tipos de serviços. Examinaremos esses mecanismos mais detalhadamente na Seção 7.6.

No outro extremo, alguns pesquisadores argumentam que não é necessário fazer nenhuma modificação fundamental no serviço de melhor esforço e nos protocolos subjacentes da Internet. Em vez disso, defendem uma abordagem liberal:

À medida que a demanda aumenta, os ISPs (de nível mais alto e de nível mais baixo) ampliarão suas redes para atendê-la. Especificamente, os provedores adicionarão mais largura de banda e capacidade de comutação para oferecer desempenho satisfatório de atraso e perda de pacotes dentro de suas redes [Huang, 2005]. Desse modo, essas empresas oferecerão melhor serviço a seus clientes (usuários e ISPs clientes), que serão traduzidos em receitas mais altas resultantes de um número maior de clientes e de tarifas de serviços também mais altas. Para certificar que as aplicações multimídia recebem o serviço adequado, até no caso de sobrecarga, um ISP pode superdimensionar a banda e a capacidade de comutação. Com uma previsão de tráfego justa e um suprimento de banda larga, uma garantia de QoS branda pode ser feita.

Redes de distribuição de conteúdo (*content distribution networks* — CDNs) duplicariam conteúdo armazenado e o colocariam nas bordas da Internet. Dado que uma grande fração do tráfego que flui pela Internet é conteúdo armazenado (páginas Web, MP3, vídeo), as CDNs podem aliviar significativamente as cargas de tráfego nos ISPs e nas interfaces de formação de pares (peering) entre ISPs. Além do mais, CDNs oferecem um serviço diferenciado para provedores de conteúdo: os que pagam pelos serviços de

uma CDN podem entregar conteúdo mais rapidamente e com mais eficiência. Estudaremos CDNs mais adiante neste capítulo, na Seção 7.3.

No caso de tráfego de fluxo contínuo ao vivo (como a transmissão de um evento esportivo) que está sendo enviado a milhões de usuários simultaneamente, podem ser disponibilizadas **redes multicast de sobreposição**. Uma rede *multicast* de sobreposição consiste em usuários hospedeiros e, possivelmente, servidores dedicados espalhados Internet. Esses hospedeiros, servidores e os enlaces lógicos entre eles formam, coletivamente, uma rede de sobreposição que transmite tráfego *multicast* (veja Seção 4.7) da origem até milhões de usuários. Diferentemente do *multicast IP*, no qual a função *multicast* é manipulada por roteadores na camada do IP, em redes de sobreposição o *multicast* ocorre na camada de aplicação. Por exemplo, o hospedeiro de origem poderia enviar uma corrente a três servidores de sobreposição; cada um desses servidores pode transmitir a corrente para mais três servidores de sobreposição; o processo continua, criando, sobre a rede IP subjacente, uma árvore de distribuição. A transmissão *multicast* de tráfego muito requisitado, ao vivo, por redes de sobreposição pode reduzir a carga total de tráfego na Internet em relação à distribuição *unicast*.

Entre o campo da reserva de largura de banda e o campo da abordagem liberal, há ainda um terceiro — o campo dos serviços diferenciados (DiffServ). Esse campo quer fazer mudanças relativamente pequenas nas camadas de rede e de transporte e introduzir esquemas simples de determinação de preços e inspeção na borda da rede (isto é, na interface entre o usuário e o ISP do usuário). A ideia é introduzir um pequeno número de classes de tráfego (possivelmente apenas duas), designar cada datagrama a uma das classes, atribuir diferentes níveis de serviço nas filas de roteadores aos datagramas conforme suas classes e cobrar dos usuários segundo a classe de pacotes que os roteadores estão enviando pela rede. Abordaremos serviços diferenciados na Seção 7.5.

Estas três abordagens diferentes para lidar com o tráfego de multimídia — obter o máximo do serviço de melhor esforço, QoS diferencial e QoS garantida — são resumidas na Tabela 7.1, e vista nas Seções 7.3, 7.5 e 7.6 respectivamente.

7.1.4 Compressão de áudio e vídeo

Antes que áudio e vídeo possam ser transmitidos por uma rede de computadores, eles devem ser digitalizados e comprimidos. A necessidade de digitalização é óbvia: redes de computadores transmitem bits; portanto, toda informação transmitida deve ser representada como uma sequência de bits. A compressão é importante porque áudio e vídeo não comprimidos consomem uma quantidade extraordinária de capacidade de armazenamento e de largura de banda; remover as redundâncias inerentes aos sinais digitalizados de áudio e vídeo pode reduzir, em muitas vezes, a quantidade de dados que precisa ser armazenada e transmitida. Como exemplo, sem compressão, uma única imagem constituída de 1.024 (com cada pixel codificado em 24 bits — 8 bits para cada uma das cores vermelha, verde e azul), exige 3 Mbytes de armazenamento. Levaria 7 minutos para enviar essa imagem por um enlace de 64 kbps. Se a imagem for comprimida na modesta razão de compressão de 10:1, a necessidade de armazenamento será reduzida para 300 kbytes e o tempo de transmissão também diminuirá por um fator de 10.

Abordagem	Localização da Unidade	Garantia	Implementação atual	Complexidade	Mecanismos
Obter o máximo do serviço de melhor esforço	Nenhuma	Nenhuma, ou branda	Qualquer local	Mínima	Supporte de camada de aplicação, CDN, suprimento extra
QoS diferencial	Classes de fluxos	Nenhuma, ou branda	Alguns locais	Média	Escalonamento, regulação
QoS garantida	Fluxos individuais	Branda ou definitiva, quando o fluxo é admitido	Poucos locais	Alta	Escalonamento, regulação, admissão de chamada e sinalização

Tabela 7.1 Três abordagens para suportar aplicações de multimídias

O tópico sobre compressão de áudio e vídeo é vasto. Áreas ativas de pesquisa nesse campo já existem há mais de 50 anos, e agora há literalmente centenas de técnicas e padrões populares para compressão tanto de áudio quanto de vídeo. Muitas universidades tem cursos completos de compressão de áudio e vídeo. Portanto, apresentamos aqui uma breve introdução de alto nível ao assunto.

Compressão de áudio na Internet

Um sinal analógico de áudio continuamente variável (que poderia emanar de voz ou de música) é normalmente convertido em um sinal digital como segue:

- O sinal analógico de áudio é primeiramente amostrado a alguma taxa fixa; por exemplo, 8.000 amostras por segundo. O valor de cada amostra é um número real arbitrário.
- Cada uma das amostras é então arredondada para um valor entre um número finito de valores. Essa operação é denominada **quantização**. O número de valores finitos — denominados valores de quantização — é tipicamente uma potência de 2, por exemplo, 256 valores de quantização.
- Cada um dos valores de quantização é representado por um número fixo de bits. Por exemplo, se houver 256 valores de quantização, então cada valor — e, portanto, cada amostra — será representado por 1 byte. Cada uma das amostras é convertida à sua representação por bits. As representações por bits de todas as amostras são, então, concatenadas em conjunto para formar a representação digitalizada do sinal.

Como exemplo, se um sinal de áudio for amostrado a uma taxa de 8.000 amostras por segundo e cada amostra for quantizada e representada por 8 bits, então o sinal digital resultante terá uma taxa de 64.000 bits por segundo. Esse sinal digital pode então ser reconvertido — isto é, decodificado — em um sinal analógico para ser reproduzido. Contudo, o sinal analógico decodificado é characteristicamente diferente do sinal de áudio original. Aumentando a taxa de amostragem e o número de valores de quantização, o sinal decodificado pode se aproximar do sinal analógico original. Assim, há uma clara permuta entre a qualidade do sinal decodificado e as exigências de armazenamento e largura de banda do sinal digital.

A técnica básica de codificação que acabamos de descrever é denominada **modulação por codificação de pulso** (*pulse code modulation* — PCM). A codificação de voz frequentemente usa PCM, com uma taxa de amostragem de 8.000 amostras por segundo e 8 bits por amostra, o que dá uma taxa de 64 kbps. O disco compacto de áudio (CD) também usa PCM, com taxa de amostragem de 44.100 amostras por segundo e 16 bits por amostra; isso dá uma taxa de 705,6 kbps para mono e 1,411 Mbps para estéreo.

Uma taxa de bits de 1,411 Mbps para música estéreo ultrapassa a maioria das taxas de acesso, e mesmo os 64 kbps para voz ultrapassam a taxa de acesso de um modem discado. Por essas razões, voz e música codificadas em PCM raramente são usadas na Internet. Em vez disso, são utilizadas técnicas de compressão para reduzir a taxa de bits. Entre as técnicas de compressão populares para voz estão **GSM** (13 kbps), **G.729** (8 kbps) e **G.723.3** (6,4 e 5,3 kbps), e um grande número de técnicas proprietárias. Uma técnica de compressão popular para música estéreo com qualidade próxima à do CD é a **MPEG 1 de camada 3**, mais conhecida como **MP3**. Codificadores MP3 normalmente comprimem para taxas de 96 kbps, 128 kbps ou 112 kbps e causam pouquíssima degradação de som. Quando um arquivo de MP3 é fragmentado em pedaços, cada pedaço ainda é reproduzível. Esse formato de arquivo sem cabeçalho permite que arquivos de música em MP3 sejam transmitidos pela Internet em fluxo contínuo (admitindo que a taxa de bits da reprodução e a velocidade da conexão com a Internet sejam compatíveis). O padrão de compressão MP3 é complexo — usa máscara psicoacústica, redução de redundância e buffer com reservatório de bits.

Compressão de vídeo na Internet

Um vídeo é uma sequência de imagens que normalmente são apresentadas a uma taxa constante; por exemplo, 24 ou 30 imagens por segundo. Uma imagem não comprimida, codificada digitalmente, é constituída de um conjunto de pixels, e cada pixel é codificado por um número de bits para representar luminância e cor. Há dois tipos de redundância em vídeo, e ambos podem ser explorados para compressão. Redundância espacial é a

redundância dentro de uma dada imagem. Por exemplo, uma imagem cuja maior parte é constituída de espaço em branco pode ser comprimida eficientemente. Redundância temporal reflete a repetição de imagens subsequentes. Se, por exemplo, uma imagem e a imagem subsequente forem exatamente as mesmas, não haverá razão para codificar novamente a imagem subsequente; será mais eficiente simplesmente indicar, durante a codificação, que a imagem subsequente é exatamente a mesma.

Os padrões de compressão MPEG estão entre as técnicas de compressão mais populares. Entre essas técnicas estão a **MPEG 1** para vídeo com qualidade de CD-ROM (1,5 Mbps), a **MPEG 2** para vídeo **DVD** de alta qualidade (3 a 6 Mbps) e a **MPEG 4** para compressão de vídeo orientada para objeto. O padrão MPEG tem muito do padrão JPEG para compressão de imagens. Os padrões **H.261** para compressão de vídeo são também muito populares na Internet. Além desses padrões, existem numerosos esquemas proprietários, entre eles o QuickTime da Apple e os codificadores da Real Networks. Leitores interessados em aprender mais sobre a codificação de áudio e vídeo podem consultar [Rao, 1996] e [Solari, 1997]. Um bom livro sobre rede multimídia em geral é [Crowcroft, 1999].



História

Fluxo contínuo armazenado de áudio e vídeo da RealNetworks ao YouTube

A RealNetworks, pioneira em fluxo contínuo de áudio e vídeo, foi a primeira empresa a trazer o áudio de Internet à corrente principal. Seu primeiro produto, o sistema RealAudio — lançado em 1995 — incluía um codificador de áudio, um servidor de áudio e um tocador de áudio. Permitindo que os usuários navegassem, selecionassem e interagissem com o fluxo de áudio, facilmente se tornou um sistema popular de distribuição para os provedores de entretenimento, educação e conteúdo jornalístico.

Hoje em dia, os fluxos de áudio e vídeo estão entre os serviços mais populares na Internet. Não somente uma abundância de empresas oferecem esse tipo de conteúdo, mas há também uma quantidade enorme de diferentes servidores, tocadores e tecnologias de protocolos sendo empregadas. Alguns exemplos interessantes (em 2009) são:

- **O Rhapsody da RealNetworks:** Fornece serviços de assinatura de fluxo contínuo e download aos usuários. Rhapsody usa seu próprio cliente proprietário, que resgata músicas de seu servidor proprietário através de HTTP. Ao passo que uma música chega através de HTTP, é tocada por um cliente Rhapsody. Acesso a conteúdo de download é restrito por um sistema de Gerenciamento de Direitos Digitais [*Digital Rights Management* — DRM].
- **MSN Vídeo:** Usuários podem ter acesso ao fluxo de uma variedade de conteúdos, incluindo notícias internacionais e videoclipes musicais. Vídeos podem ser tocados através do popular Windows Media Player (WMP), o qual está disponível na maioria dos hospedeiros Windows. A comunicação entre o WMP e os servidores da Microsoft é feito pelo protocolo proprietário MMS (Micromedia Media Server), que tipicamente tenta fazer o fluxo de conteúdo através de RTSP/RTP; se isso falha, devido aos firewalls, ele tenta recuperar conteúdo através de HTTP.
- **Muze:** Fornece serviço de amostras de áudio à varejistas, como o BestBuy e o Yahoo. Amostras de músicas selecionadas nos sites desses varejistas vêm do Muze e tem seu fluxo pelo WMP. O Muze, o Rhapsody, o YouTube e muitos outros provedores de conteúdo de fluxo usam as redes de distribuição de conteúdo (CDNs) para distribuírem seu conteúdo, como visto na Seção 7.3.
- **YouTube:** O imensamente popular serviço de compartilhamento de vídeo usa um cliente baseado em Flash (embutido em uma página Web). A comunicação entre o cliente e os servidores do YouTube é feita através de HTTP.

O que está guardado para o futuro? Hoje em dia a maior parte do conteúdo do fluxo de vídeo é de qualidade baixa, codificada em taxas de 500 kbps ou menos. A qualidade de vídeo certamente melhorou ao passo que a banda larga e o acesso à Internet via fibra em casa se tornaram acessíveis. É muito provável que nossos tocadores de música portáteis não armazenem mais músicas — em vez disso eles irão conseguir tudo por encomenda, por canais sem fio!

7.2 Áudio e vídeo de fluxo contínuo armazenados

Nos últimos anos, as aplicações de áudio e vídeo de fluxo contínuo tornaram-se populares e consumidoras significativas de largura de banda. Em áudio e vídeo de fluxo contínuo, clientes solicitam arquivos de áudio e vídeo comprimidos que residem em servidores. Como discutiremos nesta seção, esses servidores podem ser servidores Web comuns ou servidores de fluxo contínuo especiais configurados para a aplicação de áudio e vídeo de fluxo contínuo. A pedido do cliente, o servidor dirige a ele um arquivo de áudio/vídeo enviando o arquivo para uma porta. Apesar de ambos, TCP e UDP, poderem ser usados, atualmente a maior parte do tráfego de fluxo de áudio/vídeo é transportada via TCP. (Firewalls são muitas vezes configurados para bloquear o tráfego UDP. Além disso, ao usar o TCP, com sua entrega confiável, todo arquivo é transferido ao cliente sem perda de pacote, o que permite que o arquivo seja visto novamente de um cache local no futuro.) [Sripanidkulchai, 2004]. Uma vez que o arquivo requisitado de áudio/vídeo começa a chegar, o cliente começa a apresentar o arquivo (normalmente) em alguns segundos. Alguns sistemas também fornecem a interatividade ao usuário, por exemplo, as opções de pausa/retorno e pulos temporais dentro do arquivo de áudio/vídeo. O RTSP (*real-time streaming protocol* — protocolo de fluxo contínuo em tempo real), discutido no final desta seção, é um protocolo de domínio público que provê interatividade com o usuário.

Os usuários quase sempre requisitam reprodução de áudio e vídeo de fluxo contínuo por meio de um cliente Web (isto é, um browser), mas então, reproduzem e controlam o áudio/vídeo usando um **transdutor**, como o Windows Media Player ou um transdutor Flash. O transdutor desempenha diversas funções, entre as quais:

- *Descompressão*. Áudio e vídeo quase sempre são comprimidos para economizar espaço de armazenamento em disco e largura de banda de rede. Um transdutor deve descomprimir áudio e vídeo enquanto são reproduzidos.
- *Eliminação da variação de atraso*. A variação de atraso do pacote é a variabilidade dos atrasos origem-destino dos pacotes dentro da mesma corrente de pacotes. Uma vez que áudio e vídeo têm de ser reproduzidos à mesma taxa em que foram gravados, o receptor colocará os pacotes recebidos em um buffer durante um curto período de tempo para eliminar essa variação. Examinaremos esse tópico detalhadamente na Seção 7.3.

7.2.1 Acesso a áudio e vídeo por meio de um servidor Web

Áudio e vídeo armazenados podem residir no servidor Web que entrega áudio/video para o cliente sobre HTTP ou em um servidor de fluxo contínuo dedicado de áudio/vídeo usando um protocolo HTTP ou qualquer outro protocolo. Nesta subseção, examinaremos a entrega de áudio/vídeo a partir de um servidor Web; na subseção seguinte, abordaremos a entrega de áudio/vídeo a partir de um servidor de fluxo contínuo.

Considere, em primeiro lugar, o caso de áudio de fluxo contínuo. Quando um arquivo de áudio reside em um servidor Web, ele é um objeto comum dentro do sistema de arquivos do servidor, exatamente como os arquivos HTML e JPEG. Quando um usuário quer ouvir o arquivo de áudio, o hospedeiro do usuário estabelece uma conexão TCP com o servidor Web e envia uma requisição HTTP para o objeto. Ao receber a requisição, o servidor Web encapsula o arquivo de áudio em uma mensagem de resposta HTTP e devolve a mensagem de resposta para a conexão TCP. No caso do vídeo, essa operação pode ser um pouco mais complicada, porque a parte sonora e a parte visual do vídeo podem estar armazenadas em dois arquivos. Também é possível que o áudio e o vídeo estejam intercalados no mesmo arquivo, de modo que somente um objeto precise ser enviado ao cliente. Para simplificar a discussão, no caso de um vídeo propriamente dito, admitiremos que o áudio e o vídeo estejam contidos em um único arquivo.

Em muitas implementações de fluxo contínuo de áudio e vídeo através de HTTP, a funcionalidade do lado do cliente é separada em duas partes. A tarefa do navegador é requisitar um metarquivo que fornece informações (por exemplo, uma URL e o tipo de codificação, para que o transdutor adequado possa ser identificado) sobre o arquivo multimídia que deverá ser transmitido sobre HTTP. Este metarquivo é então enviado ao servidor HTTP e encaminhado do navegador ao transdutor, sendo sua tarefa contatar o servidor HTTP, que então envia o arquivo multimídia ao transdutor através de HTTP. Estes passos estão ilustrados na Figura 7.1:

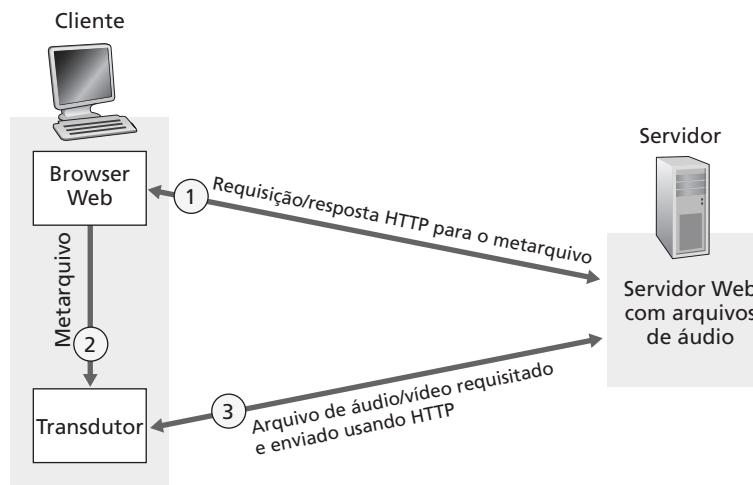


Figura 7.1 Servidor Web envia áudio/vídeo diretamente ao transdutor

1. O usuário clica sobre um hiperlink de um arquivo de áudio/vídeo. O hiperlink não aponta diretamente para o arquivo de áudio/vídeo, mas para um metarquivo. O metarquivo contém o URL do arquivo de áudio/vídeo. A mensagem de resposta HTTP que encapsula o metarquivo contém uma linha de cabeçalho de tipo de conteúdo que indica a aplicação de áudio/vídeo específica.
2. O browser cliente examina a linha de cabeçalho de tipo de conteúdo da mensagem de resposta, lança o transdutor associado e passa todo o corpo da mensagem de resposta (isto é, o metarquivo) para o transdutor.
3. O transdutor estabelece uma conexão TCP diretamente com o servidor HTTP e envia uma mensagem de requisição HTTP do arquivo de áudio/vídeo para a conexão TCP. O arquivo de áudio/vídeo é enviado ao transdutor dentro de uma mensagem de resposta HTTP. O transdutor exibe o arquivo de áudio/vídeo de fluxo contínuo.

A importância da etapa intermediária de aquisição do metarquivo é clara. Ao verificar o tipo de conteúdo do arquivo, o browser pode lançar o transdutor adequado e, dessa maneira, fazer com que o transdutor conte diretamente o servidor.

Acabamos de aprender como um metarquivo pode permitir que um transdutor dialogue diretamente com um servidor Web que armazena um arquivo de áudio/vídeo. Ainda assim, muitas empresas que vendem produtos que entregam áudio/vídeo de fluxo contínuo não recomendam a arquitetura que acabamos de descrever. Ao invés disso, elas recomendam o armazenamento de áudio/vídeo de servidores de fluxo especializados, que foram otimizados para o fluxo contínuo.

7.2.2 Envio de multimídia de um servidor de fluxo contínuo a uma aplicação auxiliar

Como servidor de fluxo contínuo pode ser proprietário, tal como os comercializados pela RealNetworks e pela Microsoft, ou pode ser de domínio público. Com um servidor de fluxo contínuo, áudio e vídeo podem ser enviados por UDP (e não por TCP) usando protocolos de camada de rede que podem ser mais bem configurados para áudio/vídeo de fluxo contínuo do que o HTTP.

Essa arquitetura requer dois servidores, como mostra a Figura 7.2. Um dos servidores, o servidor Web, serve páginas Web (incluindo metarquivos). O segundo servidor, o **servidor de fluxo contínuo**, serve os arquivos de áudio/vídeo. Os dois servidores podem rodar no mesmo sistema final ou em dois sistemas finais distintos. As etapas dessa arquitetura são semelhantes às descritas na subseção anterior. Contudo, nesse caso o transdutor requisita o arquivo de um servidor de fluxo contínuo, e não de um servidor Web, e o transdutor e o servidor de fluxo contínuo podem interagir usando seus próprios protocolos. Esses protocolos permitem uma interação mais rica do usuário com o fluxo de áudio/vídeo.

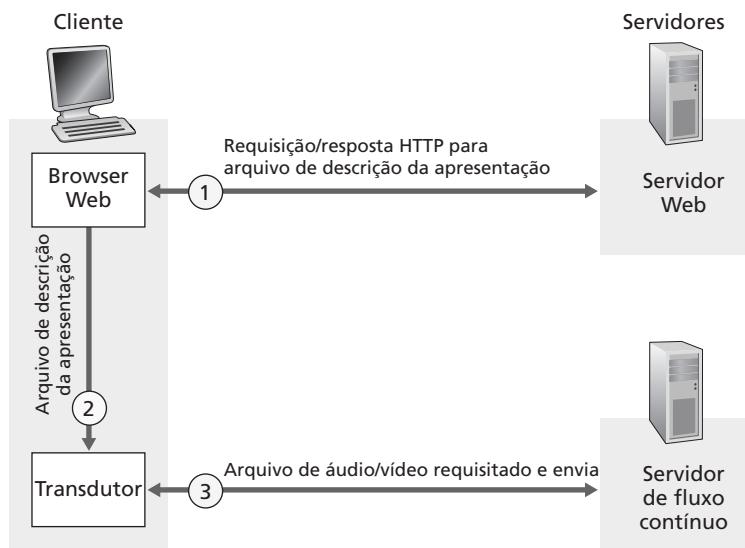


Figura 7.2 Envio de fluxo contínuo de um servidor de fluxo contínuo para um transdutor

Na arquitetura da Figura 7.2 há muitas opções para a entrega de áudio/vídeo do servidor de fluxo contínuo ao transdutor. Uma lista parcial de opções é dada a seguir:

1. O áudio/vídeo é enviado por UDP a uma taxa constante igual à taxa de reprodução do receptor (que é a taxa codificada para o áudio/vídeo). Por exemplo, se o áudio for comprimido usando GSM a uma taxa de 13 kbps, então o servidor transmitirá o arquivo comprimido de áudio a uma taxa de 13 kbps. Tão logo receba o áudio/vídeo comprimido da rede, o cliente o descomprime e o reproduz.
2. É igual à primeira opção, mas o transdutor atrasa a reprodução de 2 a 5 segundos para eliminar a variação de atraso induzida pela rede. O cliente realiza essa tarefa colocando a mídia comprimida que recebe da rede em um buffer cliente, como mostra a Figura 7.3. Tão logo tenha pré-capturado alguns poucos segundos de mídia, o cliente começa a descarregar o buffer. Para essa opção, e também para a anterior, a taxa de preenchimento $x(t)$ é igual à taxa de saída d , exceto quando há perda de pacotes, caso em que $x(t)$ é momentaneamente menor que d .
3. A mídia é enviada por TCP. O servidor lança o arquivo de mídia para dentro da porta TCP o mais rápido possível; o cliente (isto é, o transdutor) lê da porta TCP o mais rápido possível e coloca o vídeo comprimido no buffer do transdutor. Após um atraso inicial de 2 a 5 segundos, o transdutor lê de seu buffer a uma taxa d e repassa a mídia comprimida para descompressão e reprodução. Como o TCP retransmite pacotes perdidos, ele tem o potencial de fornecer melhor qualidade de som do que o UDP. Por outro lado, a taxa de preenchimento $x(t)$ varia com o tempo devido ao controle de congestionamento e ao controle de fluxo de janela do TCP. De fato, após perda de pacotes, o controle de congestionamento TCP pode

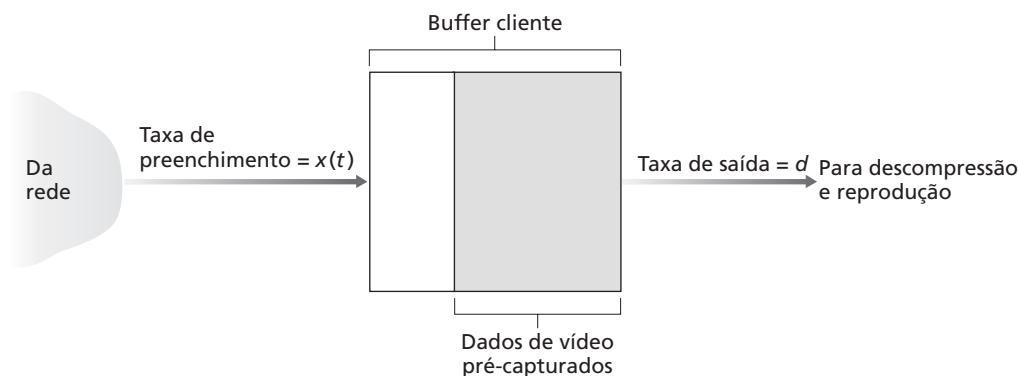


Figura 7.3 Buffer cliente preenchido a uma taxa $x(t)$ e esvaziado a uma taxa d

reduzir a taxa instantânea a valores menores do que d por longos períodos de tempo. Isso pode esvaziar o buffer cliente (um processo conhecido como **inanição**) e introduzir pausas indesejáveis na saída de uma corrente de áudio/vídeo no cliente. [Wang, 2004] mostra que quando a vazão média do TCP é aproximadamente 2 vezes maior que a taxa média de bit, o fluxo contínuo TCP resulta em inanição mínima e baixos níveis de atrasos de partida.

Para a terceira opção, o comportamento de $x(t)$ dependerá muitíssimo do tamanho do buffer cliente (que não deve ser confundido com o buffer de recepção TCP). Se o buffer for grande o suficiente para conter todo o arquivo de mídia (possivelmente por armazenagem em disco), então o TCP fará uso de toda a largura de banda instantânea disponível para a conexão, de modo que $x(t)$ pode se tornar muito maior do que d . Se $x(t)$ permanecer muito maior do que d por longos períodos de tempo, então uma grande porção da mídia será pré-capturada para dentro do cliente e uma subsequente inanição do cliente será pouco provável. Se, por outro lado, o buffer cliente for pequeno, então $x(t)$ flutuará ao redor da taxa de saída d . O risco de inanição do cliente é maior nesse caso.

7.2.3 RTSP (protocolo de fluxo contínuo em tempo real)

Muitos usuários de multimídia pela Internet (em especial aqueles que cresceram com um controle remoto de TV na mão) vão querer controlar a reprodução de mídia de taxa constante fazendo pausas na reprodução, reposicionando a reprodução em um ponto do tempo futuro ou passado, avançando ou atrasando a reprodução em modo rápido e assim por diante. Essa funcionalidade é semelhante à que um aparelho de DVD oferece ao usuário quando este assiste a um DVD ou à que oferece um aparelho de CD para ouvir a música gravada no CD. Para permitir que um usuário controle a reprodução, o transdutor e o servidor precisam de um protocolo para trocar informações de controle de reprodução. O RTSP, definido no RFC 2326, é esse protocolo.

Antes de entrar nos detalhes do RTSP, vamos apresentar o que o RTSP não faz:

- O RTSP não define esquemas de compressão para áudio e vídeo.
- O RTSP não define como áudio e vídeo são encapsulados em pacotes para transmissão por uma rede; o encapsulamento para mídia de fluxo contínuo pode ser fornecido por RTP ou por um protocolo proprietário. (O RTP é discutido na Seção 7.4.) Por exemplo, os servidores e os transdutores da RealNetworks usam o RTSP para enviar informações de controle reciprocamente, mas a corrente de mídia em si pode ser encapsulada em pacotes RTP ou em algum formato de dados proprietário.
- O RTSP não restringe o modo como a mídia de fluxo contínuo é transportada; ela pode ser transportada por UDP ou TCP.
- O RTSP não restringe o modo como o transdutor armazena o áudio/vídeo. O áudio/vídeo pode ser reproduzido logo que começa a chegar ao cliente, após um atraso de alguns segundos, ou pode ser descarregado integralmente antes de ser reproduzido.

Portanto, se o RTSP não executa nenhuma dessas funções, o que ele faz? O RTSP permite que um transdutor controle a transmissão de uma corrente de mídia. Como mencionamos há pouco, as ações de controle incluem pausa/reinício, reposicionamento da reprodução, avanço e retrocesso rápidos. O RTSP é um protocolo ‘fora da banda’. Em particular, as mensagens RTSP são enviadas fora da banda, ao passo que a corrente de mídia, cuja estrutura de pacote não é definida pelo RTSP, é considerada ‘dentro da banda’. Mensagens RTSP usam o número de porta 544; a corrente de mídia utiliza um número diferente. A especificação do RTSP [RFC 2326] permite que as mensagens RTSP sejam enviadas por TCP ou UDP.

Lembre-se de que o protocolo de transferência de arquivos (FTP — [Seção 2.3]) também usa a noção de fora da banda. Em particular, o FTP usa dois pares de portas cliente-servidor, cada par com seu próprio número de porta: um par de portas cliente-servidor suporta uma conexão TCP que transporta informações de controle; o outro par de portas cliente-servidor suporta uma conexão TCP que transporta o arquivo propriamente dito. O canal RTSP é semelhante, em muitos aspectos, ao canal de controle do FTP.

Vamos agora percorrer um exemplo simples de RTSP, que é ilustrado na Figura 7.4. O browser Web primeiramente solicita um arquivo de descrição da apresentação a um servidor Web. Esse arquivo pode conter refe-

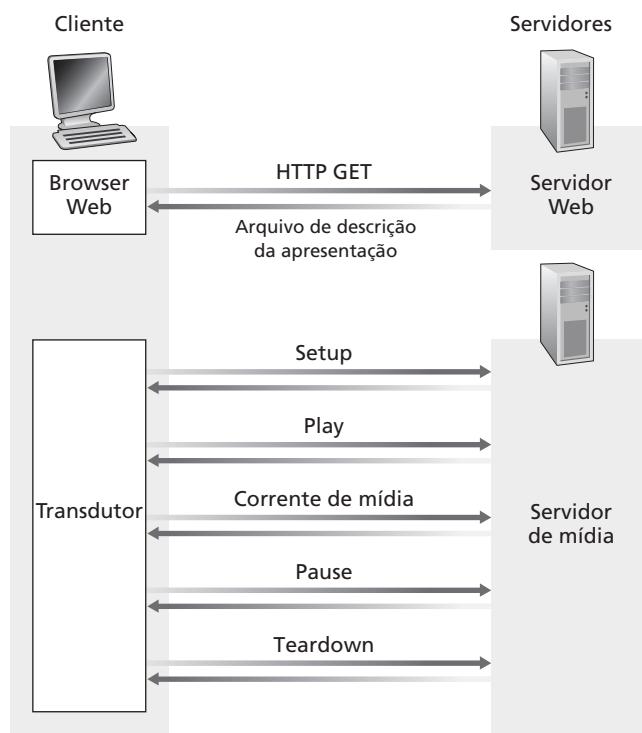


Figura 7.4 Intereração entre cliente e servidor usando RTSP

rências a diversos arquivos de mídia de taxa constante, bem como diretrizes para sincronização dos arquivos de mídia de taxa contínua. Cada referência a um arquivo de mídia de taxa contínua começa com o método URL, rtsp://. Fornecemos a seguir uma amostra de arquivo de apresentação que foi adaptado de [Schulzrinne, 1997]. Nessa apresentação, uma corrente de áudio e uma de vídeo são reproduzidas em paralelo e com voz sincronizada (como parte do mesmo grupo). Para a corrente de áudio, o transdutor pode escolher (comutar) entre duas gravações de áudio, uma de baixa fidelidade e outra de alta fidelidade. (O formato do arquivo é semelhante ao SMIL [SMIL, 2009],* que é usado por muitos produtos de taxa contínua para definir apresentações sincronizadas de multimídia.)

O servidor Web encapsula o arquivo de descrição da apresentação em uma mensagem de resposta HTTP e envia a mensagem ao browser. Quando recebe a mensagem de resposta HTTP, o browser invoca um transdutor (isto é, uma aplicação auxiliar) com base no campo de tipo de conteúdo da mensagem. O arquivo de descrição da apresentação contém referências às correntes de mídia, usando o método URL rtsp://, como mostrado no exemplo anterior. Como ilustra a Figura 7.4, o transdutor e o servidor então enviam um ao outro uma série de mensagens RTSP. O transdutor envia uma requisição RTSP SETUP e o servidor responde com uma mensagem RTSP OK. O transdutor envia uma requisição RTSP PLAY, digamos, para áudio de baixa fidelidade e o servidor responde com uma mensagem RTSP OK. Nesse ponto, o servidor de fluxo contínuo bombeia o áudio de baixa fidelidade para dentro de seu próprio canal ‘dentro da banda’. Mais tarde, o transdutor envia uma requisição RTSP PAUSE e o servidor responde com uma mensagem RTSP OK. Quando o usuário termina, o transdutor envia uma requisição RTSP TEARDOWN e o servidor confirma com uma resposta RTSP OK.

```
<title>Twister</title>
<session>
  <group language=en lipsync>
    <switch>
      <track type=audio>
```

```
e="PCMU/8000/1"
src = "rtsp://audio.example.com/twister/audio.en/lofi">
<track type=audio
e="DVI4/16000/2" pt="90 DVI4/8000/1"
src="rtsp://audio.example.com/twister/audio.en/hifi">
</switch>
<track type="video/jpeg"
src="rtsp://video.example.com/twister/video">
</group>
</session>
```

Agora vamos examinar rapidamente as mensagens RTSP propriamente ditas. Segue um exemplo simplificado de uma sessão RTSP entre um cliente (C:) e um remetente (S:).

```
C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
Cseq: 1
Transport: rtp/udp; compression; port=3056; mode=PLAY
S: RTSP/1.0 200 OK
Cseq: 1
Session: 4231
C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Range: npt=0-
Cseq: 2
Session: 4231
S: RTSP/1.0 200 OK
Cseq:2
Session:4231
C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Range: npt=37
Cseq: 3
Session: 4231
S: RTSP/1.0 200 OK
Cseq: 3
Session: 4231
C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Cseq: 4
Session: 4231
S: RTSP/1.0 200 OK
Cseq: 4
Session: 4231
```

É interessante notar as similaridades entre HTTP e RTSP. Todas as mensagens de requisição e resposta são em texto ASCII, o cliente emprega métodos padronizados (SETUP, PLAY, PAUSE e assim por diante) e o servidor responde com códigos padronizados de resposta. Uma diferença importante, contudo, é que o servidor RTSP monitora o estado do cliente para cada sessão RTSP em curso. Por exemplo, o servidor monitora se o cliente está em um estado de inicialização, um estado de reprodução (play) ou um estado de pausa (ver a tarefa de programação ao final deste capítulo). Os números de sessão e de sequência, que fazem parte de cada requisição e resposta RTSP, ajudam o servidor a monitorar o estado da sessão. O número da sessão é fixo durante toda a sessão; o cliente incrementa o número de sequência cada vez que envia uma nova mensagem; o servidor devolve um echo do número de sessão e do número de sequência corrente.

Como o exemplo demonstra, o cliente inicia a sessão com a requisição SETUP, fornecendo o URL do arquivo que deverá ser transmitido e a versão do RTSP. A mensagem de estabelecimento (SETUP) inclui o número de porta do cliente para o qual a mídia deve ser enviada. Essa mensagem também indica que a mídia deve ser enviada por UDP usando o protocolo de empacotamento RTP (que será discutido na Seção 7.4). Note que, nesse exemplo, o transdutor escolheu não reproduzir a apresentação completa, mas apenas a porção de baixa fidelidade.

Na verdade, o RTSP pode fazer muito mais do que o que descrevemos nessa breve introdução. Em particular, o RTSP tem facilidades que permitem que os clientes enviem um fluxo contínuo na direção do servidor (por exemplo, para gravar). Esse protocolo foi adotado pela RealNetworks, uma das líderes no setor de áudio/vídeo de fluxo constante. Henning Schulzrinne disponibiliza uma página Web sobre RTSP [Schulzrinne — RTSP, 2009].

Ao final deste capítulo, você encontrará uma tarefa de programação para criar um sistema de reprodução de vídeo em fluxo contínuo (para ambos, servidor e cliente) que utiliza o RTSP. Essa tarefa envolve escrever código que constrói e envia mensagens RTSP no cliente. A tarefa fornece o código do servidor RTSP, que analisa as mensagens RTSP e constrói respostas apropriadas. Incentivamos os leitores interessados em aprofundar seus conhecimentos sobre RTSP a executar essa interessante tarefa de programação.

7.3 Fazendo o melhor possível com o serviço de melhor esforço

O protocolo de camada de rede da Internet, IP, presta um serviço de melhor esforço. Isso quer dizer que o serviço faz seu melhor esforço para transportar cada datagrama da fonte ao destino o mais rápido possível. Contudo, o serviço de melhor esforço nada promete quanto à dimensão do atraso fim a fim para um pacote individual ou quanto à dimensão da variação do atraso ou da perda de pacotes dentro da corrente de pacotes. A falta de garantias quanto ao atraso e à variação do atraso propõe desafios significativos ao projeto de aplicações de multimídia em tempo real, como telefone por Internet e videoconferência em tempo real, que são muito sensíveis ao atraso, à variação do atraso e à perda de pacotes.

Nesta seção, examinaremos vários modos de melhorar o desempenho das aplicações multimídias através de redes de melhor esforço. Nossa foco será em técnicas de aplicação de camadas, por exemplo, abordagens que não requerem mudanças no núcleo de rede ou até na camada de aplicação até os hospedeiros de extremidades. Primeiro descreveremos o efeito da perda de pacotes, atrasos e a medida de variação de atrasos em aplicações multimídias. Então veremos técnicas para reparar tais falhas. Então descreveremos como redes de distribuição de conteúdo e super-provisionamento de recursos podem ser usadas para evitar tais falhas.

7.3.1 As limitações de um serviço de melhor esforço

Mencionamos que o serviço de melhor esforço pode acarretar perda de pacotes, excessivo atraso fim a fim e variação de atraso. Vamos examinar esses assuntos com maiores detalhes. Para deixarmos essa discussão mais real, discutiremos os mecanismos no contexto de uma aplicação de telefone para Internet, como descrito abaixo. Esse exemplo é similar para uma aplicação de conferência em vídeo de tempo real.

O alto-falante no nosso exemplo de telefone para Internet gera um sinal de áudio que é formado por períodos alternados de falas e silêncios. Para conservar a banda, nossa aplicação de telefone para Internet gera um pacote apenas durante os períodos de fala. Durante o período de fala, o transmissor gera bytes em uma velocidade de 8.000 bytes por segundo, e a cada 20 mseg o transmissor reúne os bytes em blocos. Assim, o número de bytes em um bloco é $(20 \text{ mseg}) \cdot (8.000 \text{ bytes/seg}) = 160 \text{ bytes}$. Um cabeçalho especial é anexado a cada bloco, sendo seu conteúdo discutido abaixo. O bloco e seu cabeçalho são encapsulados em um segmento UDP, através da interface socket. Dessa forma, durante um período de fala, um segmento UDP é enviado a cada 20 mseg.

Se cada pacote chega ao receptor e tem um pequeno atraso fim a fim constante, então pacotes chegam ao receptor periodicamente a cada 20 mseg durante um período de fala. Nessas condições ideais, o receptor pode reproduzir novamente cada bloco assim que chegarem. Mas, infelizmente, alguns pacotes podem ser perdidos e a

maioria dos pacotes não terá o mesmo atraso fim a fim, mesmo em uma Internet pouco congestionada. Por esta razão, o receptor precisa ter mais cuidado para determinar (1) quando for o momento de reproduzir um bloco, e (2) o que fazer com um bloco perdido.

Perda de pacotes

Considere um dos segmentos UDP gerados por nossa aplicação de telefone por Internet. O segmento UDP é encapsulado em um datagrama IP. Enquanto o datagrama vagueia pela rede, ele passa por buffers (isto é, por filas) nos roteadores, para poder alcançar os enlaces de saída. É possível que um ou mais dos buffers na rota entre o remetente e o destinatário estejam lotados e não possam aceitar o datagrama IP. Nesse caso, o datagrama IP será descartado e nunca chegará à aplicação receptora.

A perda pode ser eliminada enviando os pacotes por TCP, em vez de por UDP. Lembre-se de que o TCP retransmite pacotes que não chegam ao destino. Contudo, os mecanismos de retransmissão frequentemente são considerados inaceitáveis para aplicações interativas de áudio em tempo real, como o telefone por Internet, porque aumentam o atraso fim a fim [Bolot, 1996]. Além disso, devido ao controle de congestionamento do TCP, após a perda de pacote a taxa de transmissão no remetente pode ser reduzida a uma taxa mais baixa do que a taxa de reprodução no receptor, o que pode causar um forte impacto sobre a inteligibilidade da voz no receptor. Por essas razões, quase todas as aplicações de telefone por Internet existentes executam em UDP e não se preocupam em retransmitir pacotes perdidos. [Baset, 2006] relata que o UDP é usado pelo Skype a não ser que o usuário esteja atrás de um NAT ou de um firewall que bloqueia os segmentos UDP (nesse caso TCP é usado).

Mas perder pacotes não é necessariamente tão grave quanto se possa imaginar. Na verdade, taxas de perda de pacotes entre 1 e 20 por cento podem ser toleradas, dependendo de como a voz é codificada e transmitida e de como a perda é ocultada no receptor. Por exemplo, o mecanismo de correção de erros de repasse (*forward error correction* — FEC) pode ajudar a ocultar a perda de pacotes. Veremos mais adiante que, com a FEC, a informação redundante é transmitida junto com a informação original, de modo que parte dos dados originais perdidos pode ser recuperada da informação redundante. Não obstante, se um ou mais dos enlaces entre remetente e receptor estiverem fortemente congestionados e a perda de pacotes exceder 10–20 por cento (embora essas taxas raramente sejam observadas em redes bem provisionadas), então, na realidade, nada poderá ser feito para conseguir uma qualidade de som aceitável. Assim, fica claro que o serviço de melhor esforço tem suas limitações.

Atraso fim a fim

Atraso fim a fim é o acúmulo de atrasos de processamento, de transmissão e de formação de filas nos roteadores; atrasos de propagação nos enlaces e atrasos de processamento em sistemas finais. Para as aplicações de áudio altamente interativas, como o telefone por Internet, atrasos fim a fim menores do que 150 milissegundos não são percebidos pelo ouvido humano; atrasos entre 150 e 400 milissegundos podem ser aceitáveis, mas não são o ideal, e atrasos que excedem 400 milissegundos podem atrapalhar seriamente a interatividade em conversações por voz. O lado receptor de uma aplicação de telefone por Internet em geral desconsiderará quaisquer pacotes cujos atrasos ultrapassarem um determinado patamar, por exemplo, mais do que 400 milissegundos. Assim, os pacotes cujos atrasos ultrapassem o patamar são efetivamente perdidos.

Variação de atraso

Um componente crucial do atraso fim a fim são os atrasos aleatórios de fila nos roteadores. Por causa desses atrasos variáveis dentro da rede, o tempo decorrido entre o momento em que um pacote é gerado na fonte e o momento em que é recebido no destinatário pode variar de pacote para pacote. Esse fenômeno é denominado **variação de atraso**.

Como exemplo, considere dois pacotes consecutivos dentro de uma rajada de voz em nossa aplicação de telefone por Internet. O remetente envia o segundo pacote 20 milissegundos após ter enviado o primeiro. Mas, no receptor, o espaçamento entre esses pacotes pode ficar maior do que 20 milissegundos. Para entender, suponha

que o primeiro pacote chegue a uma fila de roteador quase vazia, mas que, exatamente antes de o segundo pacote chegar à fila, um grande número de pacotes vindos de outras fontes chegue à mesma fila. Como o primeiro pacote sofre um pequeno atraso de fila e o segundo pacote sofre um grande atraso de fila nesse roteador, o primeiro e o segundo pacotes ficam espaçados em mais de 20 milissegundos. O espaçamento entre pacotes consecutivos também pode ficar menor do que 20 milissegundos. Para entender, considere novamente dois pacotes consecutivos dentro de uma rajada de voz. Suponha que o primeiro pacote se junte ao final de uma fila com um grande número de pacotes e que o segundo pacote chegue à fila antes dos pacotes de outras fontes. Nesse caso, nossos dois pacotes se encontram um exatamente atrás do outro na fila. Se o tempo que leva para transmitir um pacote no enlace de saída do roteador for menor do que 20 milissegundos, então o primeiro e o segundo pacotes ficarão espaçados por menos de 20 milissegundos.

A situação é análoga a dirigir carros em rodovias. Suponha que você e um amigo seu, cada um dirigindo seu próprio carro, estejam viajando de San Diego a Phoenix, tenham um estilo semelhante de dirigir e mantenham a velocidade de 100 km/h, se o tráfego existente permitir. Por fim, suponha que seu amigo parte uma hora antes de você. Então, dependendo do tráfego, você pode chegar a Phoenix mais ou menos uma hora depois de seu amigo.

Se o receptor ignorar a presença de variação de atraso e reproduzir as porções assim que elas chegam, então a qualidade de áudio resultante poderá facilmente se tornar ininteligível no receptor. Felizmente, a variação de atraso frequentemente pode ser eliminada com a utilização de **números de sequência, marcas de tempo e atraso de reprodução**, como discutido a seguir.

7.3.2 Eliminação da variação de atraso no receptor para áudio

Para uma aplicação de voz como telefone por Internet ou áudio sob demanda, o receptor deve tentar prover reprodução sincronizada de porções de voz na presença de variação de atraso aleatório; aplicações de vídeo geralmente têm exigências semelhantes. Isto normalmente é feito combinando os três mecanismos seguintes:

Preceder cada porção com um número de sequência. O remetente aumenta em um o número de sequência para cada um dos pacotes que gera.

Preceder cada porção com uma marca de tempo. O remetente marca cada porção com o instante em que ela foi gerada.

Atrasar a reprodução de porções no receptor. O atraso da reprodução das porções de áudio recebidas deve ser suficientemente longo para que a maioria dos pacotes seja recebida antes de seus tempos de reprodução programados. O atraso da reprodução pode ser fixado para todo o período de duração da sessão de áudio ou pode variar adaptativamente durante o período útil da sessão. Os pacotes que não chegarem antes de seu tempo de reprodução programado serão considerados perdidos e esquecidos; como observado antes, o receptor pode usar algum tipo de interpolação de voz para tentar disfarçar a perda.

Discutiremos agora como esses três mecanismos, quando combinados, podem atenuar ou até eliminar os efeitos da variação de atraso. Examinaremos duas estratégias de reprodução: atraso de reprodução fixo e atraso de reprodução adaptativo.

Atraso de reprodução fixo

Com a estratégia do atraso fixo, o receptor tenta reproduzir cada porção exatamente q milissegundos após a porção ter sido gerada. Assim, se a marca de tempo de uma porção for t , o receptor reproduz a porção no tempo $t + q$, admitindo-se que a porção já tenha chegado naquele instante. Os pacotes que chegam após seus tempos de reprodução programados são descartados e considerados perdidos.

Qual é uma boa escolha para q ? Telefone por Internet pode suportar atrasos de até cerca de 400 milissegundos, embora com valores menores que q a experiência interativa resultante seja mais satisfatória. Por outro lado, caso se escolha um q muito menor do que 400 milissegundos, muitos pacotes podem perder seus tempos de reprodução programados devido à variação de atraso induzida pela rede. Em termos gerais, se grandes varia-

ções no atraso fim a fim forem características, será preferível usar um q grande; por outro lado, se o atraso for pequeno e as variações no atraso também forem pequenas, será preferível usar um q pequeno, talvez menor do que 150 milissegundos.

O compromisso entre o atraso de reprodução e a perda de pacotes é ilustrado na Figura 7.5. A figura mostra os tempos em que os pacotes são gerados e reproduzidos para uma única rajada de voz. Dois atrasos de reprodução iniciais distintos são considerados. Como mostram os degraus mais à esquerda do gráfico, o remetente gera pacotes a intervalos regulares — digamos, a cada 20 milissegundos. O primeiro pacote dessa rajada de voz é recebido no tempo r . Como mostra a figura, as chegadas dos pacotes subsequentes não são espaçadas regularmente devido à variação de atraso da rede.

Para o primeiro esquema de reprodução, o atraso inicial de reprodução está estabelecido em $p - r$. Com esse esquema, o quarto pacote não chega dentro de seu atraso de reprodução programado e o receptor o considera perdido. Pelo segundo esquema de reprodução, o atraso inicial de reprodução fixo está estabelecido em $p' - r$. Por esse esquema, todos os pacotes chegam antes de seu tempo de reprodução e, portanto, não há perda.

Atraso de reprodução adaptativo

O exemplo anterior demonstra um importante compromisso entre atraso e perda que surge ao se projetar uma estratégia de reprodução com atrasos de reprodução fixos. Ao se decidir por um atraso inicial de reprodução grande, a maioria dos pacotes cumprirá suas programações e, portanto, a perda será insignificante; contudo, para serviços interativos como telefone por Internet, atrasos longos podem se tornar incômodos, se não intoleráveis. Idealmente, gostaríamos que o atraso de reprodução fosse minimizado, mas que mantivesse a limitação de perda abaixo de uns poucos por cento.

A maneira natural de tratar esse compromisso é estimar o atraso de rede e a variância de atraso de rede e ajustar o atraso de reprodução de acordo com o resultado, no início de cada rajada de voz. Esse ajuste adaptativo de atrasos de reprodução no início das rajadas de voz fará com que os períodos de silêncio no receptor sejam comprimidos e alongados; contudo, compressão e alongamento de silêncio em pequenas quantidades não são percebidos durante a fala.

Tomando [Ramjee, 1994] como base, descrevemos agora um algoritmo genérico que o receptor pode usar para ajustar seus atrasos de reprodução adaptativamente. Para tal, consideremos:

t_i = marca de tempo do i -ésimo pacote = o instante em que o pacote foi gerado pelo remetente

r_i = o instante em que o pacote i é recebido pelo receptor

p_i = o instante em que o pacote i é reproduzido no receptor

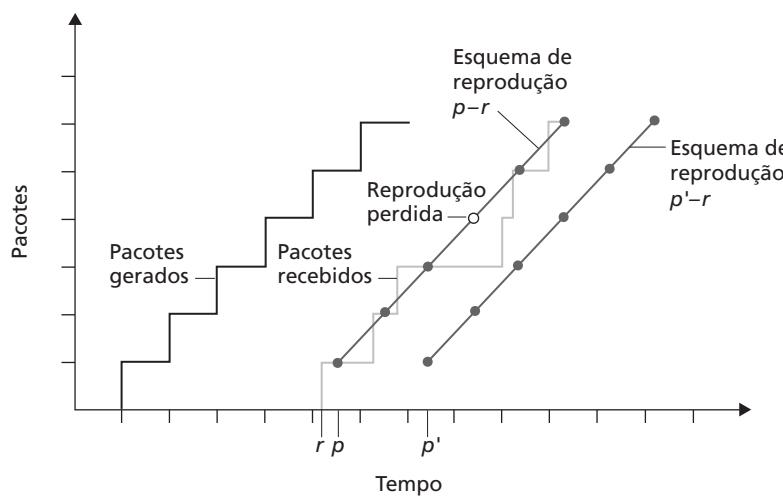


Figura 7.5 Perda de pacotes para diferentes atrasos de reprodução fixos

O atraso de rede fim a fim do i -ésimo pacote é $r_i - t_i$. Devido à variação de atraso da rede, esse atraso variará de pacote a pacote. Seja d_i a estimativa do atraso de rede médio para a recepção do i -ésimo pacote. Essa estimativa é obtida das marcas de tempo como segue:

$$d_i = (1 - u) d_{i-1} + u(r_i - t_i),$$

onde u é uma constante fixa (por exemplo, $u = 0,01$). Assim, d_i é uma média ajustada dos atrasos de rede observados $r_1 - t_1, \dots, r_i - t_i$. A estimativa atribui maior peso aos atrasos de rede verificados mais recentemente do que aos atrasos de rede ocorridos no passado distante. Esse modelo de estimativa não deve ser tão fora do comum; uma ideia semelhante é usada para estimar os tempos de viagem de ida e volta no TCP, como discutido no Capítulo 3. Seja v_i uma estimativa do desvio médio do atraso em relação ao atraso médio estimado. A estimativa também é obtida com base nas marcas de tempo:

$$v_i = (1 - u) v_{i-1} + u |r_i - t_i - d_i|.$$

As estimativas d_i e v_i são calculadas para todos os pacotes recebidos, embora somente sejam usadas para determinar o ponto de reprodução para o primeiro pacote em qualquer rajada de voz.

Uma vez calculadas essas estimativas, o receptor emprega o seguinte algoritmo para a reprodução de pacotes. Se o pacote i for o primeiro pacote de uma rajada de voz, seu tempo de reprodução p_i será computado como:

$$p_i = t_i + d_i + Kv_i,$$

onde K é uma constante positiva (por exemplo, $K = 4$). A finalidade do termo Kv_i é estabelecer o tempo de reprodução em um futuro suficientemente longínquo, de modo que apenas uma pequena fração dos pacotes que estão chegando na rajada de voz seja perdida devido às chegadas atrasadas. O ponto de reprodução para qualquer pacote subsequente em uma rajada de voz é computado como um deslocamento em relação ao ponto no tempo em que o primeiro pacote da rajada de voz foi reproduzido. Em particular, seja

$$q_i = p_i - t_i$$

a duração do tempo decorrido desde a geração do primeiro pacote da rajada de voz até sua reprodução. Se o pacote j também pertencer a essa rajada de voz, ele será reproduzido no tempo

$$p_j = t_j + q_i.$$

O algoritmo que acabamos de descrever tem perfeito sentido admitindo-se que o receptor possa dizer se um pacote é o primeiro pacote na rajada de voz. Se não houver perda de pacotes, então o receptor poderá determinar se o pacote i é o primeiro pacote da rajada de voz comparando a marca de tempo do i -ésimo pacote com a marca de tempo do pacote de ordem ($i - 1$). Realmente, se $t_i - t_{i-1} > 20$ milissegundos, então o receptor sabe que o i -ésimo pacote inicia uma nova rajada de voz. Mas suponha agora que haja uma perda de pacote ocasional. Nesse caso, dois pacotes sucessivos recebidos no destino podem ter marcas de tempo que diferem de mais de 20 milissegundos quando os dois pacotes pertencem à mesma rajada de voz. E é nesse caso que os números de sequência são particularmente úteis. O receptor pode usar os números de sequência para determinar se uma diferença de mais de 20 milissegundos nas marcas de tempo é devida a uma nova rajada de voz ou a pacotes perdidos.

Áudio e vídeo de fluxo contínuo armazenados

Vamos concluir esta seção falando um pouco sobre fluxo contínuo de áudio e vídeo armazenados. Aplicações de fluxo contínuo de áudio/vídeo armazenado também usam, tipicamente, números de sequência, marcas de tempo e atraso de reprodução para aliviar ou até eliminar os efeitos da variação de atraso da rede. Contudo, há uma importante diferença entre áudio/vídeo interativo em tempo real e fluxo contínuo de áudio/vídeo armazenado. Especificamente, fluxo contínuo de áudio/vídeo pode tolerar atrasos significativamente maiores. De fato, quando um usuário requisita um clipe de áudio/vídeo, é provável que ele ache aceitável esperar 5 segundos ou mais antes que a reprodução se inicie. E a maioria dos usuários pode tolerar atrasos semelhantes após ações interativas, como um salto temporal dentro de uma corrente de mídia. Essa tolerância maior ao atraso dá ao desenvolvedor de aplicações maior flexibilidade ao projetar aplicações de mídias armazenadas.

7.3.3 Recuperação de perda de pacotes

Já discutimos com algum detalhe como uma aplicação de telefone por Internet pode lidar com a variação de atraso de pacote. Agora, descreveremos brevemente diversos esquemas que tentam preservar uma qualidade aceitável de áudio na presença da perda de pacotes. Esses esquemas são denominados **esquemas de recuperação de perdas**. Definimos aqui a perda de pacotes em um sentido amplo: um pacote será considerado perdido se nunca chegar ao receptor ou se chegar após o tempo de reprodução programado. Nossa exemplo do telefone por Internet servirá outra vez de contexto para a descrição de esquemas de recuperação de perdas.

Como mencionamos no início desta seção, em geral a retransmissão de pacotes perdidos não é apropriada para aplicações interativas em tempo real como telefone por Internet. Na verdade, a retransmissão de um pacote que perdeu seu prazo de reprodução não serve para absolutamente nada. E a retransmissão de um pacote que transbordou de uma fila de roteador normalmente não pode ser feita com a necessária rapidez. Devido a essas considerações, aplicações de telefone por Internet frequentemente usam algum tipo de esquema de prevenção de perda. Dois desses esquemas são a **FEC** e a **intercalação**.

FEC (correção de erros de repasse)

A ideia básica da FEC é adicionar informações redundantes à corrente de pacotes original. Ao custo de aumentar marginalmente a taxa de transmissão do áudio da corrente, a informação redundante pode ser usada para reconstruir aproximações ou versões exatas de alguns pacotes perdidos. Esboçamos, agora, dois mecanismos de FEC segundo [Bolot, 1996] e [Perkins, 1998]. O primeiro mecanismo envia uma porção redundante codificada após cada n porções. A porção redundante é obtida por XOR das n porções originais [Shacham, 1990]. Desse modo, se qualquer pacote do grupo de $n + 1$ pacotes for perdido, o receptor poderá reconstruir integralmente o pacote perdido. Mas, se dois ou mais pacotes de um grupo forem perdidos, o receptor não poderá reconstruir os pacotes perdidos. Mantendo $n + 1$ (o tamanho do grupo) pequeno, uma grande fração dos pacotes perdidos pode ser recuperada, quando a perda não for excessiva. Contudo, quanto menor o tamanho do grupo, maior será o aumento relativo da taxa de transmissão da corrente de áudio. Em particular, a taxa de transmissão aumenta de um fator de $1/n$; por exemplo, se $n = 3$, então a taxa de transmissão aumenta 33 por cento. Além disso, esse esquema simples aumenta o atraso de reprodução, pois o receptor tem de esperar o término da recepção do grupo de pacotes inteiro antes de poder começar a reproduzir. Se o leitor quiser mais detalhes práticos sobre como a FEC funciona para transporte de multimídia, consulte [RFC 2733].

O segundo mecanismo de FEC é enviar uma corrente de áudio de resolução mais baixa como informação redundante. Por exemplo, o remetente pode criar uma corrente de áudio nominal e uma corrente de áudio correspondente de baixa resolução e baixa taxa de bits. (A corrente nominal poderia ser uma codificação PCM de 64 kbps e a corrente de qualidade mais baixa, uma codificação GSM de 13 kbps.) A corrente de baixa taxa de bits é denominada corrente redundante. Como mostra a Figura 7.6, o remetente constrói o enésimo pacote tomando a enésima porção da corrente nominal e anexando a ela a porção de ordem $(n - 1)$ da corrente redundante. Dessa maneira, sempre que houver perda de pacote não consecutiva, o receptor pode ocultar a perda reproduzindo a porção codificada de baixa taxa de bits que chegar com o pacote subsequente. É evidente que as porções de baixa taxa de bits oferecem qualidade mais baixa do que as porções nominais. Contudo, uma corrente na qual a maioria das porções é de alta qualidade, as porções de baixa qualidade são ocasionais e nenhuma porção é perdida oferece boa qualidade geral de áudio. Note que, nesse esquema, o receptor tem de receber somente dois pacotes antes da reprodução, de modo que o aumento no atraso de reprodução é pequeno. Além disso, se a codificação de baixa taxa de bits for muito menor do que a codificação nominal, o aumento marginal da taxa de transmissão será pequeno.

Para enfrentar perdas consecutivas, podemos utilizar uma variação simples. Em vez de anexar apenas a porção de baixa taxa de bits de ordem $(n - 1)$ à enésima porção nominal, o remetente pode anexar as porções de baixa taxa de bits de ordens $(n - 1)$ e $(n - 2)$ ou anexar as porções de baixa taxa de bits de ordens $(n - 1)$ e $(n - 3)$ e assim por diante. Anexando mais porções de baixa taxa de bits a cada porção nominal, a qualidade do áudio no receptor fica aceitável para uma variedade mais ampla de ambientes de melhor esforço adversos. Por outro lado, as porções adicionais aumentam a largura de banda de transmissão e o atraso de reprodução.

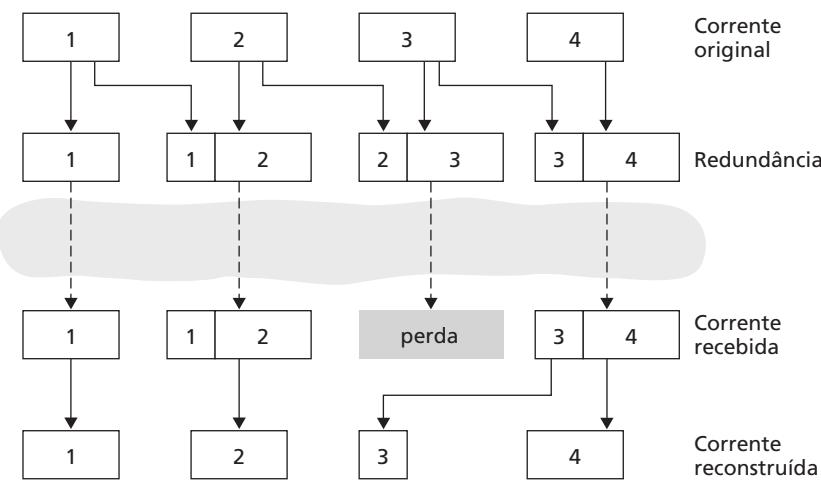


Figura 7.6 Dando carona à informação redundante de qualidade mais baixa

RAT [RAT, 2009] é uma aplicação de telefone por Internet bem documentada que usa FEC. Ela pode transmitir correntes de áudio de qualidade mais baixa juntamente com a corrente de áudio nominal, como descrito antes. Veja também [Rosenberg, 2000].

Intercalação

Como uma alternativa à transmissão redundante, uma aplicação de telefone por Internet pode enviar áudio intercalado. Como mostra a Figura 7.7, o remetente rearranja a sequência das unidades de dados de áudio antes da transmissão, de modo que unidades originalmente adjacentes fiquem separadas por uma certa distância na corrente transmitida. A intercalação pode atenuar o efeito da perda de pacotes. Se, por exemplo, as unidades têm comprimento de 5 milissegundos e as porções são de 20 milissegundos (isto é, 4 unidades por porção), então a primeira porção pode conter unidades 1, 5, 9, 13; a segunda porção pode conter unidades 2, 6, 10, 14, e assim por diante. A Figura 7.7 mostra que a perda de um único pacote de uma corrente intercalada resulta em várias

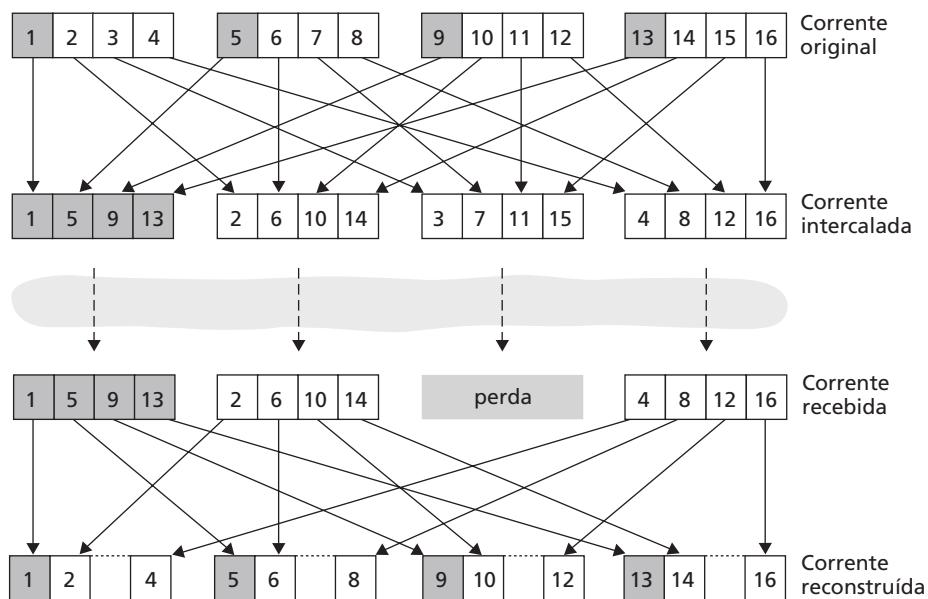


Figura 7.7 Envio de áudio intercalado

pequenas lacunas na corrente reconstruída, em vez de em uma única lacuna grande que ocorreria com um sistema não intercalado.

A intercalação pode melhorar significativamente a qualidade percebida de uma corrente de áudio [Perkins, 1998]. Além disso, a sobrecarga é baixa. A desvantagem óbvia da intercalação é que ela aumenta a latência, o que limita seu uso em aplicações interativas como o telefone por Internet, embora possa funcionar bem para fluxo contínuo de áudio armazenado. Uma importante vantagem da intercalação é que ela não aumenta as exigências de largura de banda de uma corrente.

Reparação de correntes de áudio danificadas realizada no receptor

Esquemas de recuperação baseados no receptor tentam produzir um substituto para um pacote perdido semelhante ao original. Como discutido em [Perkins, 1998], isso é possível desde que os sinais de áudio, em especial os de voz, exibam grandes índices de semelhança entre si dentro de períodos de tempo reduzidos. Assim, essas técnicas funcionam para taxas de perda relativamente pequenas (menos de 15 por cento) e para pacotes pequenos (4–40 milissegundos). Quando o comprimento da perda se aproxima do comprimento de um fonema (5–100 milissegundos), essas técnicas causam pane, já que fonemas inteiros podem ser perdidos pelo ouvinte.

Talvez o modo mais simples de recuperação baseada no receptor seja a repetição de pacotes. A repetição de pacotes substitui pacotes perdidos por cópias de pacotes que chegaram imediatamente antes da perda. É de baixa complexidade computacional e funciona razoavelmente bem. Outro modo de recuperação baseada no receptor é a interpolação, que usa áudio antes e após a perda para interpolar um pacote adequado para cobrir a perda. Ela funciona um pouco melhor do que a repetição de pacotes, mas é significativamente mais trabalhosa para processamento [Perkins, 1998].

7.3.4 Distribuição de multimídia: redes de distribuição de conteúdo

Com taxas de vídeo em fluxo contínuo na faixa de centenas de kbps para vídeos de baixa resolução até vários Mbps para vídeo em DVD, a tarefa de transmitir um vídeo armazenado em fluxo contínuo, sob demanda, a um grande número de usuários distribuídos geograficamente é um desafio e tanto! A abordagem mais simples seria armazenar o vídeo em um único servidor e simplesmente transmiti-lo de um servidor de vídeo (ou de uma unidade lógica de processamento corporativo — *server farm*) a um cliente, para cada requisição de cliente, como foi discutido na Seção 7.2. Mas há dois problemas óbvios nessa solução. Em primeiro lugar, como um cliente pode estar muito distante do servidor, os pacotes de servidor para o cliente podem passar por muitos ISPs, aumentando a probabilidade de atrasos e perdas significativos. Em segundo lugar, se o vídeo for muito popular, provavelmente será enviado muitas vezes através dos mesmos ISPs (e pelos mesmos enlaces de comunicação), consumindo, por conseguinte, largura de banda significativa. No Capítulo 2 discutimos como o cache pode atenuar esses problemas. Embora tenhamos discutido cache em termos do conteúdo tradicional da Web, deve ficar claro que ele também é apropriado para conteúdo de multimídia, como áudio e vídeo armazenados. Nesta seção, discutiremos **redes de distribuição de conteúdo** (*content distribution networks* — CDNs), que oferecem uma abordagem alternativa à distribuição de conteúdo de multimídia armazenado (bem como à distribuição de conteúdo Web tradicional).

A filosofia básica das CDNs é que, se o cliente não pode vir até o conteúdo (porque o caminho de melhor esforço do cliente ao servidor não pode suportar vídeo de fluxo contínuo), então o conteúdo deve ser levado até o cliente. Assim, CDNs usam um modelo diferente do cache na Web. Para uma CDN, os clientes pagantes não são mais as ISPs, mas as provedoras de conteúdo. Uma provedora de conteúdo que tenha um vídeo para distribuir (tal como a CNN) paga a uma empresa CDN (como a Akamai) para levar seu vídeo até os usuários requisitantes com os menores atrasos possíveis.

Uma empresa CDN tipicamente provê seu serviço de distribuição de conteúdo da seguinte maneira:

1. A empresa de CDN instala centenas de **servidores CDN** por toda a Internet. Em geral a empresa instala seus servidores CDN em uma **central de dados**, que muitas vezes estão em ISPs de nível mais baixo, próximas às redes de acesso ISP e aos clientes.

2. A CDN duplica o conteúdo de seu cliente nos servidores CDN. Sempre que seu cliente atualiza seu conteúdo, a CDN redistribui o conteúdo renovado aos servidores CDN.
3. A empresa CDN provê um mecanismo que entrega, a um cliente Web que tenha requisitado, conteúdo pelo servidor CDN que melhor possa atendê-lo. Esse servidor pode ser o que estiver mais próximo do cliente (talvez no mesmo ISP) ou pode ser um servidor CDN cujo caminho até o cliente está livre de congestionamento.

A Figura 7.8 mostra a interação entre a provedora de conteúdo e a empresa CDN. A provedora de conteúdo primeiro determina qual de seus objetos (por exemplo, vídeos) ela quer que a CDN distribua. (A provedora de conteúdo distribui os objetos remanescentes sem a intervenção da CDN.) A provedora de conteúdo rotula o conteúdo e então o envia para um nó CDN, que, por sua vez, duplica esse conteúdo e o retransmite para todos os seus servidores CDN. A empresa CDN pode ser proprietária de uma rede privada que levará o conteúdo do nó CDN até os servidores CDN. Sempre que a provedora de conteúdo modifica um objeto distribuído pela CDN, ela envia a versão atualizada ao nó CDN, que, mais uma vez imediatamente duplica e distribui o objeto aos servidores CDN. É importante ter em mente que cada servidor CDN normalmente contém objetos de muitas provedoras de conteúdo.

Agora surge a pergunta interessante. Quando um browser no hospedeiro de um usuário recebe instruções para extrair um objeto específico (identificado por um URL), como o browser determina se deve extrair o objeto do servidor de origem ou de um dos servidores CDN? Normalmente as CDNs utilizam redirecionamento DNS para guiar os browsers até o servidor correto [Kangasharju, 2000].

Suponha, por exemplo, que o nome de hospedeiro da servidora de conteúdo seja `www.foo.com`. Suponha que o nome da empresa CDN seja `cdn.com`. Suponha ainda que a provedora de conteúdo quer que somente seus arquivos de vídeo mpeg sejam distribuídos pela CDN; todos os outros objetos, incluindo as páginas de base HTML, são distribuídos diretamente pela provedora de conteúdo. Para conseguir isso, a provedora de conteúdo modifica todos os objetos HTML no servidor de origem de modo que os URLs dos arquivos de vídeo tenham o prefixo `http://www.cdn.com`. Assim, se um arquivo HTML na provedora de conteúdo tinha, originalmente, uma referência a `http://www.foo.com/sports/highlights.mpg`, a provedora de conteúdo rotularia esse objeto substituindo a referência no arquivo HTML por `http://www.cdn.com/www.foo.com/sports/highlights.mpg`.

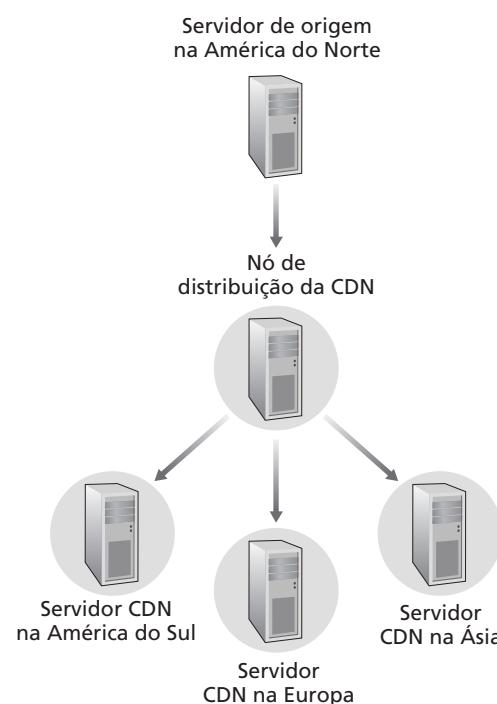


Figura 7.8 A CDN envia a seus servidores CDN objetos rotulados da provedora de conteúdo

Quando um browser requisita uma página Web que contém a imagem `ruth.mpg`, ocorrem as seguintes ações:

1. O browser envia sua requisição para o objeto base HTML ao servidor de origem `www.foo.com`, que envia o objeto HTML requisitado ao browser. O browser analisa o arquivo HTML e encontra a referência a `http://www.cdn.com/www.foo.com/sports/highlights.mpg`.
2. O browser então faz uma consulta DNS em `www.cdn.com`, que é o nome de hospedeiro para o URL referenciado. O DNS é configurado de modo que todas as consultas sobre `www.cdn.com` que chegarem a um servidor DNS raiz sejam enviadas a um servidor DNS com autoridade para `www.cdn.com`. Quando o servidor DNS com autoridade recebe a consulta, extrai o endereço IP do browser requisitante. Usando um mapa interno da rede que construiu para toda a Internet, o servidor DNS da CDN retorna o endereço IP do servidor CDN que parece ser o melhor para o browser requisitante (muitas vezes é o servidor CDN mais próximo do browser).
3. O DNS no cliente requisitante recebe uma resposta DNS com o endereço IP. O browser então envia sua requisição HTTP ao servidor CDN que tem aquele endereço IP. O browser obtém `ruth.mpg` de seu servidor CDN. Para requisições subsequentes de `www.cdn.com`, o cliente continua a usar o mesmo servidor CDN, já que o endereço IP para `www.cdn.com` está no cache DNS (no hospedeiro cliente ou no servidor local de nomes DNS).

Em suma, como ilustrado na Figura 7.9, o hospedeiro requisitante primeiro vai ao servidor Web de origem para obter o objeto base HTML, em seguida vai até o servidor DNS com autoridade pertencente à CDN para obter o endereço IP do melhor servidor CDN e, finalmente, vai até aquele servidor CDN para obter o vídeo. Note que, para implementar esse esquema de distribuição, não é preciso fazer nenhuma mudança no HTTP, no DNS nem no browser.

Ainda resta explicar como uma empresa CDN determina o ‘melhor’ servidor CDN para o hospedeiro requisitante. Embora cada empresa CDN tenha seu modo proprietário exclusivo para fazer isso, não é difícil ter uma ideia aproximada do que elas fazem. A empresa CDN monitora o melhor servidor CDN para cada acesso ISP na Internet que contenha clientes requisitantes potenciais. Então a CDN determina o melhor servidor CDN com base em seu conhecimento das tabelas de roteamento da Internet (especificamente, as tabelas BGP, que discutimos no Capítulo 4), em estimativas de tempo de viagem de ida e volta e em outros dados de medição que possui, de seus vários servidores a várias redes de acesso.

Se o leitor quiser uma discussão mais detalhada, pode consultar [Verma, 2001]. Dessa maneira, a CDN estima qual servidor CDN provê o melhor serviço de melhor esforço para o ISP. A CDN faz isso para um grande número de acessos ISPs na Internet e utiliza essa informação para configurar o servidor DNS com autoridade. Para uma discussão sobre o fluco contínuo de mídia do ponto de vista de uma grande CDN operacional (Akamai), veja [Sripanidkulchpai, 2004]. Para desenvolvimentos recentes em pesquisas sobre CDN, veja [Krishnamurthy, 2001; Mao, 2002; Saroiu, 2002; Freedman, 2004; Huang, 2008].

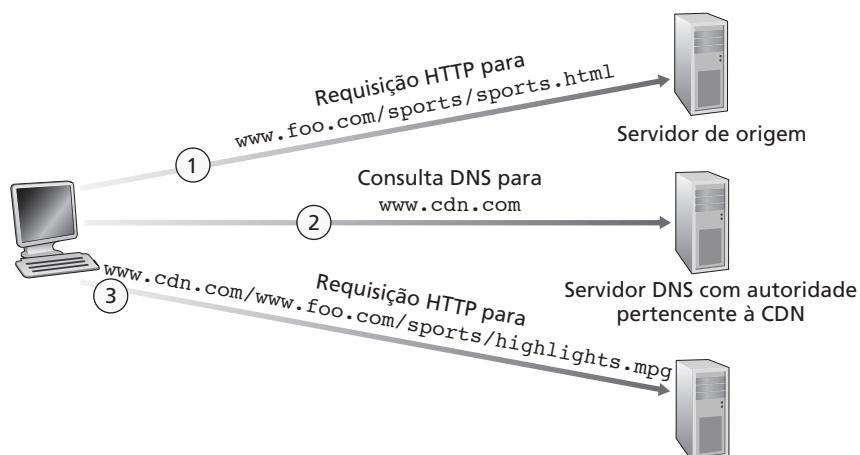


Figura 7.9 A CDN usa DNS para direcionar requisições a um servidor CDN próximo

7.3.5 Dimensionando redes de melhor desempenho para fornecer um serviço de qualidade

Em seções anteriores, vimos como técnicas de nível de aplicação como reprodução de pacotes, FEC, pacotes intercalados nos hospedeiros, e uma estrutura CDN implementada através de rede, podem melhorar a qualidade de aplicações multimídias na Internet de melhor esforço hoje em dia. Fundamentalmente, a dificuldade de suportar aplicações multimídia surgem de suas restritas exigências de desempenho — baixo atraso de pacote fim a fim, variação de atraso e perda — e pelo fato que atraso fim a fim, variação de atraso e perda ocorrem sempre que uma rede se torna congestionada. Uma abordagem final para melhorar a qualidade das aplicações multimídia — uma abordagem que pode ser usada frequentemente para resolver qualquer problema onde os recursos são restritos — é simplesmente “jogar dinheiro no problema” e assim simplesmente evitar a contenção de recursos. No caso de multimídia em rede, isso significa fornecer capacidade de enlace suficiente através da rede, para que o congestionamento de rede, e suas consequentes perdas e atrasos de pacote, nunca (ou raramente) ocorram. Com capacidade de enlace suficiente, pacotes poderiam ser compactados através da Internet atual sem perda ou atrasos. Por muitas perspectivas esta é uma situação ideal — aplicações multimídia funcionaria perfeitamente, usuários estariam felizes, e isto poderia ser alcançado sem mudanças na arquitetura de melhor esforço da Internet. A pergunta, claro, é quanta capacidade é “suficiente” para atingir este nirvana, e se os custos de fornecimento de banda larga “suficiente” são convenientes do ponto de vista comercial aos ISPs.

A questão de quanta capacidade deve-se fornecer aos enlaces da rede em uma determinada topologia para atingir um determinado desempenho fim a fim é frequentemente conhecido como provisionamento de banda. O problema mais complicado de como se projetar uma topologia de rede (no momento em que colocar roteadores, como interconectar roteadores com enlaces, e que capacidade designar aos enlaces) para obter um determinado desempenho fim a fim é um problema de projeto de rede, normalmente conhecido como dimensionamento de rede. Ambos provisionamento de banda e dimensionamento de rede são tópicos complexos, muito além da finalidade desse livro-texto. Enfatizaremos aqui, no entanto, que os seguintes assuntos devem ser abordados para prever o desempenho de nível de aplicação entre dois terminais de rede, assim providenciando capacidade suficiente para adequar-se às exigências de desempenho da aplicação.

Modelos de demanda de tráfego entre terminais de rede. Modelos precisarão ser especificados tanto no nível de chamada (por exemplo, usuários “chegando” à rede e iniciando suas aplicações fim-fim), quanto no nível de pacote (por exemplo, pacotes sendo gerados em aplicações em andamento). Observe que a carga de trabalho pode mudar através dos tempos.

Exigências de desempenho definidas. Por exemplo, a exigência de desempenho para suportar tráfego sensível ao atraso, como uma aplicação interativa de áudio/vídeo, pode ser que a probabilidade de atraso fim-fim sofrido pelas mensagens da aplicação seja maior que o limite máximo tolerável de atraso, seja menor que um valor pequeno, e [Fraleigh, 2003].

Modelos para prever desempenho fim-fim de uma modelo de uma determinada carga de trabalho, e técnicas para descobrir um custo mínimo de alocação de banda larga que resultará em todas exigências do usuário garantidas. Aqui, pesquisadores estão ocupados desenvolvendo modelos de listas de tarefas (Veja a Seção 1.4) que podem quantificar o desempenho de uma determinada carga de trabalho, e melhorar as técnicas para descobrir o custo mínimo de alocações de banda larga a fim de satisfazer as exigências do desempenho.

Visto que hoje em dia a Internet de melhor esforço poderia (do ponto de vista tecnológico) dar suporte ao tráfego multimídia em um desempenho apropriado se fosse direcionada a isto, a pergunta objetiva seria por que que a Internet não faz isso? As respostas são primeiramente econômicas e organizacionais. Do ponto de vista econômico, será que os usuários estariam dispostos a pagar a seus ISPs, para que estes instalassem a banda larga que suportaria as aplicações multimídias através de uma Internet de melhor esforço? As questões organizacionais talvez sejam mais desencorajadoras. Observe que um caminho fim-fim entre dois terminais multimídia passaria através de redes de múltiplos ISPs. Do ponto de vista organizacional, esses ISPs concordariam em ajudar (talvez com a repartição dos ganhos) na certificação de que o caminho fim-fim é propriamente dimensionado para dar suporte à aplicações multimídia? Para uma perspectiva dessas questões econômicas e organizacionais, veja



[Davies, 2005]. Para uma perspectiva sobre o fornecimento de redes backbone para suportar o tráfego de atraso sensível, veja [Fraleigh, 2003].

7.4 Protocolos para aplicações interativas em tempo real

Aplicações interativas em tempo real, incluindo telefone por Internet e videoconferência, prometem liderar grande parte do crescimento futuro da Internet. Portanto, não é surpresa que organismos padronizadores, tais como a IETF e o ITU, tenham se ocupado durante tantos anos (e ainda se ocupem!) com a produção de padrões para essa classe de aplicações. Tendo à mão padrões apropriados para aplicações interativas em tempo real, empresas independentes poderão criar novos produtos atraentes que interagem uns com os outros. Nesta seção estudamos RTP, SIP e H.323 para aplicações interativas em tempo real. Todos os três conjuntos de padrões estão sendo implementados amplamente em produtos do setor.

7.4.1 Protocolo de tempo real (RTP)

Na seção anterior, aprendemos que o lado remetente de uma aplicação de multimídia anexa campos de cabeçalho às porções de áudio/vídeo antes de passá-las à camada de transporte. Esses campos de cabeçalhos contêm números de sequência e marcas de tempo. Já que a maioria das aplicações de rede multimídia pode fazer uso de números de sequência e de marcas de tempo, é conveniente ter uma estrutura de pacote padronizada que inclua campos para dados de áudio/vídeo, números de sequência e marcas de tempo, bem como outros campos potencialmente úteis. O RTP, definido no RFC 3550, é um padrão desse tipo. Ele pode ser usado para transportar formatos comuns como PCM, GSM e MP3 para som e MPEG e H.263 para vídeo. O protocolo também pode ser usado para transportar formatos proprietários de som e de vídeo. Hoje, o RTP é amplamente implementado em centenas de protótipos de produtos e de pesquisa. Também é complementar a outros importantes protocolos interativos de tempo real, entre eles SIP e H.323.

Nesta seção, faremos uma introdução ao RTP e a seu protocolo companheiro, o RTCP. Também aconselhamos o leitor a visitar o site de Henning Schulzrinne [Schulzrinne, RTP 2009], que trata do RTP e traz uma profusão de informações sobre o assunto. O leitor também poderá visitar o site da RAT [RAT, 2009], que documenta uma aplicação de telefone por Internet que usa RTP.

O básico do RTP

O RTP comumente roda sobre UDP. O lado remetente encapsula uma porção de mídia dentro de um pacote RTP, em seguida encapsula o pacote em um segmento UDP, e então passa o segmento para o IP. O lado receptor extraí o pacote RTP do segmento UDP, em seguida extraí a porção de mídia do pacote RTP e então passa a porção para o transdutor para decodificação e apresentação.

Como exemplo, considere a utilização do RTP para transportar voz. Suponha que a fonte de voz esteja codificada (isto é, amostrada, quantizada e digitalizada) em PCM a 64 kbps. Suponha também que a aplicação colete os dados codificados em porções de 20 milissegundos, isto é, 160 bytes por porção. O lado remetente precede cada porção dos dados de áudio com um **cabeçalho RTP** que contém o tipo de codificação de áudio, um número de sequência e uma marca de tempo. O tamanho do cabeçalho RTP é normalmente 12 bytes. A porção de áudio, juntamente com o cabeçalho RTP, forma o **pacote RTP**. O pacote RTP é, então, enviado para dentro do socket de interface UDP. No lado receptor, a aplicação recebe o pacote RTP da interface do seu socket. A aplicação extraí a porção de áudio do pacote RTP e usa os campos de cabeçalho do pacote RTP para decodificar e reproduzir adequadamente a porção de áudio.

Se uma aplicação incorporar o RTP — em vez de um esquema proprietário para fornecer tipo de carga útil, número de sequência ou marcas de tempo —, ela interagirá mais facilmente com as outras aplicações de rede multimídia. Por exemplo, se duas empresas diferentes desenvolvem um software de telefone por Internet e ambas incorporam o RTP a seu produto, pode haver alguma chance de um usuário que estiver usando um dos produtos de telefone por Internet conseguir se comunicar com um usuário que estiver usando o outro produto de telefone por Internet. Na Seção 7.4.3, veremos que o RTP é muito utilizado em conjunto com os padrões de telefonia por Internet.

Devemos enfatizar que o RTP não fornece nenhum mecanismo que assegure a entrega de dados a tempo nem fornece outras garantias de qualidade de serviço (QoS); ele não garante nem mesmo a entrega dos pacotes nem impede a entrega de pacotes fora de ordem. Na verdade, o encapsulamento realizado pelo RTP é visto somente nos sistemas finais. Os roteadores não distinguem datagramas IP que carregam pacotes RTP de datagramas IP que não os carregam.

O RTP permite que seja atribuída a cada fonte (por exemplo, uma câmera ou um microfone) sua própria corrente independente de pacotes RTP. Por exemplo, para uma videoconferência entre dois participantes, quatro correntes RTP podem ser abertas — duas correntes para transmitir o áudio (uma em cada direção) e duas correntes para transmitir o vídeo (uma em cada direção). Contudo, muitas técnicas de codificação populares — incluindo MPEG1 e o MPEG2 — conjugam áudio e vídeo em uma única corrente durante o processo de codificação. Quando áudio e vídeo são conjugados pelo codificador, somente uma corrente RTP é gerada em cada direção.

Pacotes RTP não são limitados às aplicações *unicast*. Eles podem também ser enviados por árvores *multicast* um-para-muitos e muitos-para-muitos. Para uma sessão *multicast* muitos-para-muitos, todos os remetentes e fontes da sessão em geral usam o mesmo grupo *multicast* para enviar suas correntes RTP. Correntes *multicast* RTP que formam um conjunto, como correntes de áudio e vídeo que emanam de vários remetentes em uma aplicação de videoconferência, pertencem a uma **sessão RTP**.

Campos de cabeçalho do pacote RTP

Como mostra a Figura 7.10, os quatro principais campos de cabeçalho do pacote RTP são os campos de tipo da carga útil, número de sequência, marca de tempo e os campos identificadores da fonte.

O campo de tipo de carga útil do pacote RTP tem 7 bits de comprimento. Para uma corrente de áudio, o campo de tipo de carga útil serve para indicar o tipo de codificação de áudio (por exemplo, PCM, modulação delta adaptativa, codificação por previsão linear) que está sendo usado. Se um remetente decidir mudar a codificação no meio de uma sessão, ele poderá informar a mudança ao receptor por meio desse campo de tipo de carga útil. O remetente pode querer mudar o código para aumentar a qualidade do áudio ou para reduzir a taxa de bits da corrente RTP. A Tabela 7.2 apresenta alguns dos tipos de carga útil de áudio correntemente suportados pelo RTP.

Para uma corrente de vídeo, o tipo de carga útil é usado para indicar o tipo de codificação de vídeo (por exemplo, JPEG com movimento, MPEG1, MPEG2, H.261). Novamente, o remetente pode mudar a codificação de vídeo durante a sessão. A Tabela 7.3 apresenta alguns tipos de carga útil de vídeo atualmente suportados pelo RTP. Os outros campos importantes são:

Campo do número de sequência. O campo do número de sequência tem comprimento de 16 bits. O número de sequência é incrementado de uma unidade a cada pacote RTP enviado e pode ser usado pelo receptor para detectar perda de pacotes e restaurar a sequência de pacotes. Por exemplo, se o lado receptor da aplicação receber uma corrente de pacotes RTP com uma lacuna entre os números de sequência 86 e 89, então o receptor saberá que os pacotes 87 e 88 estão faltando. O receptor poderá, então, tentar ocultar os dados perdidos.

Campo de marca de tempo. O campo de marca de tempo tem 32 bits de comprimento. Ele reflete o instante da amostragem do primeiro byte no pacote RTP. Como vimos na seção anterior, o receptor pode usar marcas de tempo para eliminar a variação de atraso dos pacotes introduzida na rede e fornecer recepção sincronizada no receptor. A marca de tempo é derivada de um relógio de amostragem no remetente. Como exemplo, para áudio, o relógio de marca de tempo é incrementado de uma unidade para cada período de amostragem (por exemplo, a cada 125 microssegundos para um relógio de amostragem de 8 kHz); se a aplicação de áudio gerar porções consistindo em 160 amostras codificadas, então a marca de tempo aumentará em 160 para cada pacote RTP enquanto a fonte estiver ativa. O relógio de marca de tempo continua a aumentar a uma taxa constante, mesmo que a fonte esteja inativa.

Tipo de carga útil	Número de sequência	Marca de tempo	Identificador de sincronização da fonte	Campos variados
--------------------	---------------------	----------------	---	-----------------

Figura 7.10 Campos de cabeçalho do RTP

Número do tipo de carga útil	Formato de áudio	Taxa de amostragem	Vazão
0	PCM lei μ	8 kHz	64 kbps
1	1016	8 kHz	4,8 kbps
3	GSM	8 kHz	13 kbps
7	LPC	8 kHz	2,4 kbps
9	G.722	16 kHz	48–64 kbps
14	Áudio MPEG	90 kHz	—
15	G.728	8 kHz	16 kbps

Tabela 7.2 Tipos de carga útil de áudio suportados pelo RTP

SSRC Identificador de sincronização da fonte (*synchronization source identifier* — SSRC). O campo SSRC tem 32 bits de comprimento. Ele identifica a fonte da corrente RTP. Em geral, cada fonte de uma sessão RTP tem um SSRC distinto. O SSRC não é o endereço IP do remetente, mas um número atribuído aleatoriamente pela fonte quando uma nova corrente é iniciada. A probabilidade de que seja atribuído o mesmo SSRC a duas correntes é muito pequena. Se isso acontecer, as duas fontes escolherão novos valores SSRC.

Desenvolvimento de aplicações de software com RTP

Há duas abordagens para desenvolver uma aplicação de rede baseada em RTP. A primeira abordagem é o desenvolvedor incorporar o RTP à mão, isto é, escrever, ele mesmo, o código que executa o encapsulamento RTP no lado remetente e decodifica o RTP no lado do receptor. A segunda abordagem é o desenvolvedor da aplicação utilizar as bibliotecas de RTP (para programadores que usam linguagem C) e as classes Java (para programadores em Java) existentes, que executam o encapsulamento e a decodificação para a aplicação. Como você pode estar ansioso para escrever sua primeira aplicação de rede multimídia usando RTP, vamos nos aprofundar um pouco nessas duas abordagens. (A tarefa de programação ao final deste capítulo servirá de guia para a criação de uma aplicação RTP.) Faremos isso no contexto da comunicação *unicast* (em vez de *multicast*).

Lembre-se de que, no Capítulo 2, mostramos que a API UDP requer que, para cada segmento UDP que envia, o processo remetente determine o endereço IP de destino e o número da porta de destino antes de enviar o pacote para dentro do socket UDP. Então, o segmento UDP vagueará pela Internet e (se não for perdido, por exemplo, por causa de transbordamento de buffer de roteador), a qualquer momento, chegará ao socket do processo receptor para a aplicação. Esse socket terá um endereço completo, ou seja, o endereço IP de destino e o número da porta de destino. Na verdade, qualquer datagrama IP que contenha esse endereço IP de destino e número de porta de destino será direcionado ao socket UDP do processo receptor. (A API UDP também permite que o desenvolvedor da aplicação estabeleça o número UDP de porta da fonte; contudo, esse valor não tem nenhum efeito sobre o processo para o qual o segmento é enviado.) É importante notar que o RTP não impõe um número de porta específico. Quando o desenvolvedor de aplicações cria uma aplicação RTP, ele especificará os números de porta para os dois lados da aplicação.

Número do tipo de carga útil	Formato de vídeo
26	Motion JPEG
31	H.261
32	Vídeo MPEG1
33	Vídeo MPEG2

Tabela 7.3 Alguns tipos de carga útil de vídeo suportados pelo RTP

Como parte da tarefa de programação deste capítulo, você escreverá um servidor RTP que encapsula quadros de vídeo armazenado dentro de pacotes RTP. Você fará isso à mão; isto é, sua aplicação pegará um quadro de vídeo, adicionará os cabeçalhos RTP ao quadro para criar um pacote RTP e então passará o quadro RTP para a porta UDP. Para fazer isso, você precisará criar campos de reserva para os vários cabeçalhos RTP, incluindo um campo de número de sequência e um campo de marca de tempo. E, para cada pacote RTP que for criado, você terá de determinar o número de sequência e a marca de tempo apropriados. Você codificará explicitamente todas essas operações RTP dentro do lado remetente de sua aplicação. Como mostra a Figura 7.11, sua API com a rede será o socket padrão UDP.

Uma abordagem alternativa (que não é desenvolvida na tarefa de programação) é usar uma classe Java RTP (ou uma biblioteca RTP em C para programadores que utilizam essa linguagem) para implementar as operações RTP. Como mostra a Figura 7.12, com essa abordagem o desenvolvedor terá a impressão de que o RTP é parte da camada de transporte. Sem entrar nos aspectos fundamentais (pois eles dependem da classe/biblioteca), quando envia uma porção de mídia para dentro da API, o lado remetente da aplicação precisa fornecer à interface a porção de mídia em si, um número de tipo de carga útil, um SSRC e uma marca de tempo, juntamente com um número de porta de destino e um endereço IP de destino. Mencionamos aqui que a Java Media Framework (JMF) inclui uma implementação completa de RTP.

7.4.2 Protocolo de controle RTP (RTCP)

O RFC 3550 também especifica o RTCP, um protocolo que uma aplicação de rede multimídia pode usar juntamente com o RTP. Como mostrado no cenário *multicast* da Figura 7.13, pacotes RTCP são transmitidos por cada participante de uma sessão RTP para todos os outros participantes da sessão usando IP *multicast*. Para uma sessão RTP há, tipicamente, um único endereço *multicast* e todos os pacotes RTP e RTCP pertencentes à sessão usam o endereço *multicast*. Pacotes RTP e RTCP se distinguem uns dos outros pela utilização de números de porta distintos. (O número de porta RTCP é configurado para ser igual ao número da porta RTP mais uma unidade.)

Pacotes RTCP não encapsulam porções de áudio ou de vídeo. Em vez disso, eles são enviados periodicamente e contêm relatórios de remetente e/ou receptor com dados estatísticos que podem ser úteis para a aplicação. Esses dados estatísticos contêm número de pacotes enviados, número de pacotes perdidos e variação de atraso entre chegadas. A especificação RTP [RFC 3550] não determina o que a aplicação deve fazer com essas informações de retorno; isso depende do desenvolvedor da aplicação. Remetentes podem usar as informações, por exemplo, para modificar suas taxas de transmissão. Essas informações também podem ser utilizadas para finalidades de diagnóstico; por exemplo, receptores podem determinar se os problemas são locais, regionais ou globais.

Tipos de pacotes RTCP

Para cada corrente RTP que um receptor recebe como parte de uma sessão, ele gera um relatório de recepção. O receptor agrupa seus relatórios de recepção em um único pacote RTCP. O pacote é então enviado para a árvore *multicast* que conecta todos os participantes da sessão. O relatório de recepção contém diversos campos; os mais importantes são relacionados a seguir:

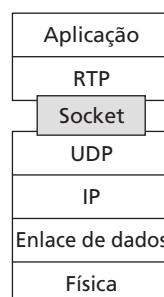


Figura 7.11 RTP é parte da aplicação e está acima do socket UDP

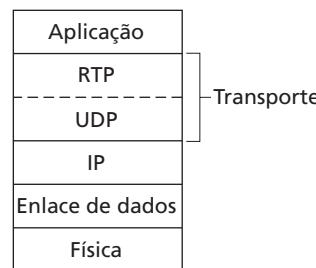


Figura 7.12 RTP pode ser visto como uma subcamada da camada de transporte

- O SSRC da corrente RTP para a qual o relatório de recepção está sendo gerado.
- A fração de pacotes perdida dentro da corrente RTP. Cada receptor calcula o número de pacotes RTP perdidos dividido pelo número de pacotes RTP enviados como parte da corrente. Se um remetente receber relatórios de recepção indicando que os receptores estão recebendo apenas uma pequena fração dos pacotes transmitidos por ele, ele poderá passar para uma taxa de codificação mais baixa, com o objetivo de reduzir o congestionamento da rede e melhorar a taxa de recepção.
- O último número de sequência recebido na corrente de pacotes RTP.
- A variação de atraso entre chegadas, que é uma estimativa amortecida da variação do intervalo de tempo entre chegadas de pacotes sucessivos na corrente RTP.

Para cada corrente RTP que um remetente estiver transmitindo, ele criará e transmitirá pacotes de relatório de remetente RTCP. Esses pacotes contêm informações sobre a corrente RTP, incluindo:

- O SSRC da corrente RTP.
- A marca de tempo e o tempo do relógio físico (tempo real) do pacote da corrente RTP gerado mais recentemente.
- O número de pacotes enviados na corrente.
- O número de bytes enviados na corrente.

Relatórios de remetente podem ser usados para sincronizar diferentes correntes de mídia dentro de uma sessão RTP. Por exemplo, considere uma aplicação de videoconferência para a qual cada remetente gera duas

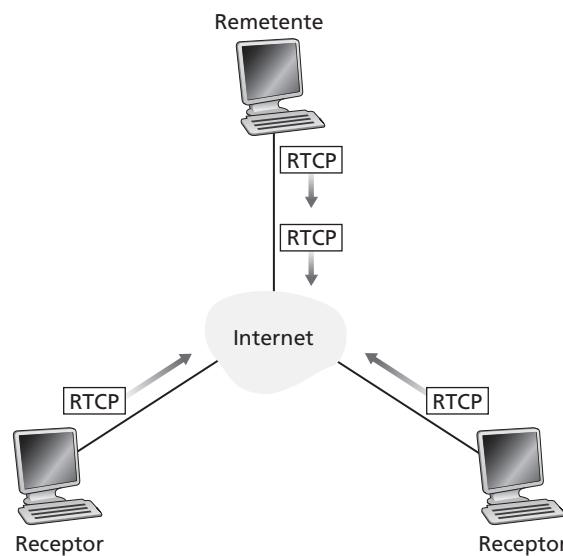


Figura 7.13 Ambos, remetentes e receptores, enviam mensagens RTCP

correntes RTP independentes, uma para vídeo e uma para áudio. As marcas de tempo desses pacotes RTP estão vinculadas aos relógios de amostragem de vídeo e áudio, e não ao tempo do relógio físico (isto é, ao tempo real). Cada relatório de remetente RTCP contém, para o pacote gerado mais recentemente na corrente RTP associada, a marca de tempo do pacote RTP e o tempo do relógio físico referente à criação do pacote. Assim, os pacotes de relatório de remetente RTCP associam o relógio de amostragem ao relógio físico, de tempo real. Os receptores podem usar essa associação presente nos relatórios de remetente para sincronizar a reprodução de áudio e vídeo.

Para cada corrente RTP que está transmitindo, o remetente também cria e transmite pacotes de descrição da fonte. Esses pacotes contêm informações sobre a fonte, como o endereço de e-mail do remetente, o nome do remetente e a aplicação que gera a corrente RTP. Contém também o SSRC da corrente RTP associada e fornecem um mapeamento entre o identificador da fonte (isto é, o SSRC) e o nome do usuário/hospedeiro.

Pacotes RTCP podem ser empilhados, isto é, relatórios de recepção do receptor, relatórios de remetentes e descrições da fonte podem ser concatenados em um único pacote. O pacote resultante é, então, encapsulado dentro de um segmento UDP e repassado para a árvore *multicast*.

Escalabilidade da largura de banda do RTCP

Você já deve ter observado que o RTCP apresenta um problema potencial de escalabilidade. Considere, por exemplo, uma sessão RTP que consista em um remetente e um grande número de receptores. Se cada um dos receptores gerar periodicamente pacotes RTCP, então a taxa de transmissão agregada de pacotes RTCP poderá exceder em muito a taxa de pacotes RTP enviados pelo remetente. Observe que o total de tráfego RTP enviado à árvore *multicast* não muda à medida que o número de receptores aumenta, ao passo que o total de tráfego RTCP aumenta linearmente com o número de receptores. Para resolver esse problema de escalabilidade, o RTCP muda a taxa com a qual um participante envia pacotes RTCP para dentro da árvore *multicast* como uma função do número de participantes da sessão. Além disso, como cada participante envia pacotes de controle para todos os outros, cada um deles pode estimar o número total de participantes da sessão [Friedman, 1999].

O RTCP tenta limitar seu tráfego a 5 por cento da largura de banda da sessão. Por exemplo, suponha que haja um remetente que está enviando vídeo a uma taxa de 2 Mbps. Então, o RTCP tenta limitar seu tráfego a 5 por cento de 2 Mbps, ou a 100 kbps, como a seguir. O protocolo dá 75 por cento dessa taxa, ou seja, 75 kbps, aos receptores; dá os restantes 25 por cento da taxa, ou 25 kbps, ao remetente. Os 75 kbps dedicados aos receptores são compartilhados igualmente entre eles. Assim, se houver R receptores, cada receptor conseguirá enviar tráfego RTCP a uma taxa de $75/R$ kbps e o remetente conseguirá enviar tráfego RTCP a uma taxa de 25 kbps. Um participante (remetente ou receptor) determina o período de transmissão do pacote RTCP calculando dinamicamente o tamanho médio do pacote RTCP (para toda a sessão) e dividindo o tamanho médio do pacote RTCP pela taxa alocada a ele, participante. Em suma, para um remetente, o período para transmitir pacotes RTCP é:

$$T = \frac{\text{número de remetentes}}{0,25 \cdot 0,05 \cdot \text{largura de banda da sessão}} \quad (\text{tamanho médio do pacote RTCP})$$

E, para um receptor, o período para transmitir pacotes RTCP é:

$$T = \frac{\text{número de receptores}}{0,75 \cdot 0,05 \cdot \text{largura de banda da sessão}} \quad (\text{tamanho médio do pacote RTCP})$$

7.4.3 SIP

Imagine um mundo no qual, enquanto você está trabalhando em seu computador, ele recebe suas chamadas telefônicas pela Internet. E, quando você sai para descansar um pouco, as novas chamadas telefônicas que chegam são automaticamente roteadas para seu PDA. E, quando você está dirigindo, novas chamadas telefônicas são automaticamente roteadas para algum equipamento conectado à Internet instalado no seu carro. Nesse mesmo mundo, durante uma conferência em rede você pode acessar uma agenda de endereços e convidar outras pessoas

para participar da reunião virtual. Esses outros participantes poderão estar trabalhando em seus computadores ou em trânsito com seus PDAs à mão ou em seus carros — não importa onde estejam, seu convite será roteado para eles de modo transparente. Ainda nesse mesmo mundo, quando você visitar alguma home page pessoal, encontrará um link ‘Ligue para mim’; ao clicar sobre esse link, será estabelecida uma sessão de telefone Internet entre seu computador e o proprietário da home page (onde quer que ele esteja).

Nesse mundo não existirá mais uma rede de telefonia por comutação de circuitos. Em vez disso, todas as chamadas telefônicas passarão pela Internet — fim a fim. Ainda nesse mesmo mundo, empresas não utilizarão mais centrais privadas de comutação telefônica (*private automatic branch exchange* — PABX), isto é, mesas locais de comutação de circuitos, para manipular chamadas telefônicas internas. Em vez disso, o tráfego telefônico interno fluirá pela LAN de alta velocidade da empresa.

Tudo isso pode parecer ficção científica. E é claro que as redes de comutação de circuitos e centrais de PABX de hoje não vão desaparecer completamente no futuro próximo (Jiang, 2001). Não obstante, já existem protocolos e produtos para transformar essa visão em realidade. Entre os protocolos mais promissores para essa finalidade está o Protocolo de Inicialização de Sessão (*Session Initiation Protocol* — SIP), definido no RFC 3261; RFC 5411. O SIP é um protocolo simples que faz o seguinte:

- Provê mecanismos para estabelecer chamadas entre dois interlocutores por uma rede IP. Permite que quem chama avise ao que é chamado que quer iniciar uma chamada. Permite que os participantes concordem com a codificação da mídia. E também permite que encerrem as chamadas.
- Provê mecanismos que permitem a quem chama determinar o endereço IP corrente de quem é chamado. Os usuários não têm um endereço IP único, fixo, porque podem receber endereços dinamicamente (usando DHCP) e porque podem ter vários equipamentos IP, cada um com um endereço IP diferente.
- Provê mecanismos para gerenciamento de chamadas, tais como adicionar novas correntes de mídia, mudar a codificação, convidar outros participantes, tudo durante a chamada, e ainda transferir e seguir chamadas.

Estabelecendo uma chamada para um endereço IP conhecido

Para entender a essência do SIP, é melhor examinar um exemplo concreto. Nesse exemplo, Alice está trabalhando em seu computador e quer chamar Bob, que também está trabalhando em seu próprio computador. Ambos os PCs estão equipados com software baseado em SIP para fazer e receber chamadas telefônicas. Nesse exemplo inicial, admitiremos que Alice conhece o endereço IP do PC de Bob. A Figura 7.14 ilustra o processo de estabelecimento de chamada do SIP.

Na Figura 7.14, vemos que uma sessão SIP começa quando Alice envia a Bob uma mensagem INVITE, que é parecida com uma mensagem de requisição HTTP. Essa mensagem INVITE é enviada por UDP à porta bem conhecida 5060, que é a porta do SIP. (Mensagens SIP também podem ser enviadas por TCP). A mensagem INVITE inclui um identificador para Bob (bob@193.64.210.89), uma indicação do endereço IP corrente de Alice, uma indicação de que Alice deseja receber áudio, o qual deve ser codificado em formato AVP 0 (PCM codificado com lei μ) e encapsulado em RTP e uma indicação de que ela quer receber os pacotes RTP na porta 38060. Após receber a mensagem INVITE de Alice, Bob envia uma mensagem de resposta SIP, que é parecida com uma mensagem de resposta HTTP. Essa mensagem de resposta SIP também é enviada à porta SIP 5060. A resposta de Bob inclui um 200 OK, bem como uma indicação de seu endereço IP, o código e o empacotamento que deseja para recepção e seu número de porta para a qual os pacotes de áudio devem ser enviados. Note que, nesse exemplo, Alice e Bob vão usar mecanismos diferentes de codificação de áudio: Alice deve codificar seu áudio com GSM, ao passo que Bob deve codificar seu áudio com a lei μ do PCM. Após receber a resposta de Bob, Alice lhe envia uma mensagem SIP de reconhecimento (ACK). Após essa transação SIP, Bob e Alice podem conversar. (Para melhor visualização, a Figura 7.14 mostra Alice falando depois de Bob, mas, na verdade, normalmente eles falariam ao mesmo tempo.) Bob codificará e empacotará o áudio como solicitado e enviará os pacotes de áudio para a porta número 38060 no endereço IP 167.180.112.24. Alice também codificará e empacotará o áudio como solicitado e enviará os pacotes de áudio para a porta número 48753 no endereço IP 193.64.210.89.

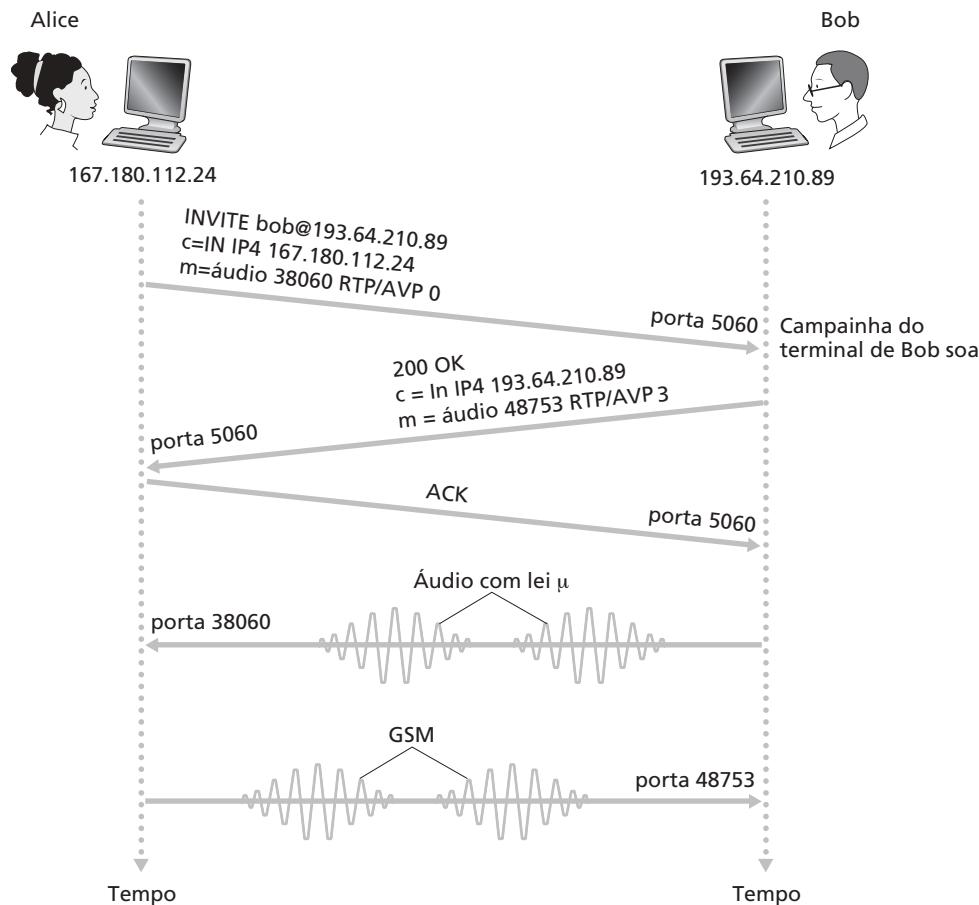


Figura 7.14 Estabelecimento de chamada SIP quando Alice conhece o endereço IP de Bob

Com esse exemplo simples, aprendemos várias características fundamentais do SIP. Em primeiro lugar, o SIP é um protocolo ‘fora da banda’: as mensagens SIP são enviadas e recebidas em portas diferentes das utilizadas para enviar e receber os dados da mídia. Em segundo lugar, as próprias mensagens SIP podem ser lidas em ASCII e são parecidas com mensagens HTTP. Em terceiro lugar, o SIP requer que todas as mensagens sejam reconhecidas, portanto, ele pode executar sobre UDP ou TCP.

Nesse exemplo, vamos considerar o que aconteceria se Bob não tivesse um codec PCM lei μ para codificar áudio. Nesse caso, em vez de responder com 200 OK, Bob responderia com um 600 Not Acceptable e apresentaria na mensagem uma lista com todos os codecs que ele pode usar. Então, Alice escolheria um dos codecs da lista e enviaria outra mensagem INVITE, desta vez anunciando o codec escolhido. Bob também poderia simplesmente rejeitar a chamada enviando um dos muitos códigos de rejeição de resposta possíveis. (Há muitos desses códigos, entre eles: ‘ocupado’ (busy), ‘encerrado’ (gone), ‘pagamento solicitado’ (payment required) e ‘proibido’ (forbidden)).

Endereços SIP

No exemplo anterior, o endereço SIP de Bob é `sip:bob@193.64.210.89`. Contudo, esperamos que muitos endereços SIP — se não a maioria — sejam parecidos com endereços de e-mail. Por exemplo, o endereço de Bob poderia ser `sip:bob@domain.com`. Quando o dispositivo SIP de Alice enviasse uma mensagem INVITE, a mensagem incluiria esse endereço semelhante a um endereço de e-mail; a infraestrutura SIP então rotearia a mensagem ao dispositivo IP que Bob está usando no momento em questão (como discutiremos mais adiante). Outras formas possíveis para o endereço SIP poderiam ser o número de telefone legado de Bob ou simplesmente seu nome próprio, ou seu primeiro ou segundo sobrenome (admitindo-se que sejam únicos).



Uma característica interessante de endereços SIP é que eles podem ser incluídos em páginas Web, exatamente como endereços de e-mail são incluídos nessas páginas juntamente com o URL de envio. Por exemplo, suponha que Bob tem uma home page pessoal e quer fornecer um meio para que os visitantes da página entrem em contato com ele. Ele poderia, então, simplesmente incluir o URL ‘sip:bob@domain.com’. Quando o visitante clicar sobre o URL, a aplicação SIP no dispositivo do visitante será lançada e uma mensagem INVITE será enviada a Bob.

Mensagens SIP

Nessa curta introdução ao SIP, não abordaremos todos os tipos e cabeçalhos de mensagens SIP. Em lugar disso, examinaremos brevemente a mensagem SIP INVITE, juntamente com algumas linhas de cabeçalho comuns. Mais uma vez, vamos supor que Alice quer iniciar uma chamada telefônica IP com Bob e, desta vez, ela conhece somente o endereço SIP de Bob, bob@domain.com, e não conhece o endereço IP do dispositivo que Bob está correntemente usando. Então, sua mensagem poderia ser algo parecido com:

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

A linha INVITE inclui a versão do SIP, assim como uma mensagem de requisição HTTP. Sempre que uma mensagem SIP passa por um dispositivo SIP (incluindo o dispositivo que origina a mensagem), ele anexa um cabeçalho Via, que indica o endereço IP do dispositivo. (Veremos, em breve, que uma mensagem INVITE típica passa por muitos dispositivos SIP antes de chegar à aplicação SIP do usuário que foi chamado.) Semelhante a uma mensagem de e-mail, a mensagem SIP inclui uma linha de cabeçalho From e uma linha de cabeçalho To. Inclui também um Call-ID (identificador de chamadas), que identifica a chamada exclusivamente (semelhante à mensagem-ID no e-mail). Inclui uma linha de cabeçalho Content-Type (Tipo de Conteúdo), que define o formato usado para descrever o conteúdo da mensagem SIP. Inclui também uma linha de cabeçalho Content-Length (Tamanho de Conteúdo), que indica o comprimento em bytes do conteúdo na mensagem. Finalmente, após um carriage return e um line feed, a mensagem contém o corpo de informação. Nesse caso, o conteúdo fornece informações sobre o endereço IP de Alice e como Alice quer receber o áudio.

Tradução de nome e localização de usuário

No exemplo da Figura 7.14, admitimos que o dispositivo SIP de Alice conhecia o endereço IP onde Bob podia ser contatado. Mas essa premissa é pouco realista, não somente porque muitas vezes os endereços IP são atribuídos dinamicamente com DHCP, mas também porque Bob pode ter vários dispositivos IP (por exemplo, dispositivos diferentes em casa, no trabalho e no carro). Portanto, agora vamos supor que Alice conhece somente o endereço de e-mail de Bob, bob@domain.com, e que esse mesmo endereço é usado para chamadas SIP. Nesse caso, Alice precisa obter o endereço IP do dispositivo que o usuário bob@domain.com está usando naquele momento. Para descobrir esse endereço, Alice cria uma mensagem INVITE que começa com INVITE bob@domain.com SIP/2.0, e envia essa mensagem a um proxy SIP. O proxy responderá com uma resposta SIP que poderá incluir o endereço IP do dispositivo que bob@domain.com está usando naquele momento. Alternativamente, a resposta poderia incluir o endereço IP da caixa postal de voz de Bob, ou poderia incluir um URL de uma página Web (que diz “Bob está dormindo. Não perturbe!”). Além disso, o resultado devolvido pelo proxy poderia depender de quem chama: se a chamada vier da sogra de Bob, ele poderia responder com o URL que aponta para a página Web do “estou dormindo”!

Agora, você provavelmente está imaginando como o servidor proxy pode determinar o endereço IP correto para bob@domain.com. Para responder a essa pergunta, é preciso dizer algumas palavras sobre um outro dispositi-

tivo SIP, a entidade registradora SIP. Cada usuário SIP tem uma entidade registradora associada. Sempre que um usuário lança uma aplicação SIP em um dispositivo, a aplicação envia uma mensagem de registro SIP à entidade registradora, informando seu endereço IP corrente. Por exemplo, quando Bob lançasse sua aplicação SIP em seu PDA, a aplicação enviaria uma mensagem semelhante a essa:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

A entidade registradora monitora o endereço IP corrente de Bob. Sempre que ele mudar para um novo dispositivo SIP, esse novo dispositivo enviará uma nova mensagem de registro, indicando o novo endereço IP. Além disso, se Bob permanecer no mesmo dispositivo durante um longo período de tempo, o dispositivo enviará mensagens de confirmação de registro, indicando que o endereço IP mais recentemente enviado ainda é válido. (No exemplo acima, é preciso enviar mensagens de confirmação a cada 3.600 segundos para manter o endereço no servidor da entidade registradora.) O fato de a entidade registradora ser análoga a um servidor de nomes DNS com autoridade não é digno de nota; o servidor DNS traduz nomes de hospedeiros fixos para endereços IP fixos; a entidade registradora SIP traduz identificadores fixos de pessoas (por exemplo, bob@domain.com) para endereços IP dinâmicos. Muitas vezes, as entidades registradoras SIP e os proxies SIP são executadas no mesmo hospedeiro.

Agora vamos examinar como o servidor proxy SIP de Alice obtém o endereço IP corrente de Bob. Vimos, na discussão precedente, que o servidor proxy precisa simplesmente transmitir a mensagem INVITE de Alice à entidade registradora/proxy de Bob. Então, a registradora/proxy poderia transmitir a mensagem ao dispositivo SIP corrente de Bob. Finalmente, como agora Bob já recebeu a mensagem INVITE de Alice, ele pode enviar a ela uma resposta SIP.

Como exemplo, considere a Figura 7.15, na qual jim@umass.edu, que está correntemente trabalhando em 217.123.56.89, quer iniciar uma sessão de voz sobre IP com keith@upenn.edu, correntemente trabalhando em 197.87.54.21. São seguidas as seguintes etapas: (1) Jim envia uma mensagem INVITE ao proxy SIP de umass.

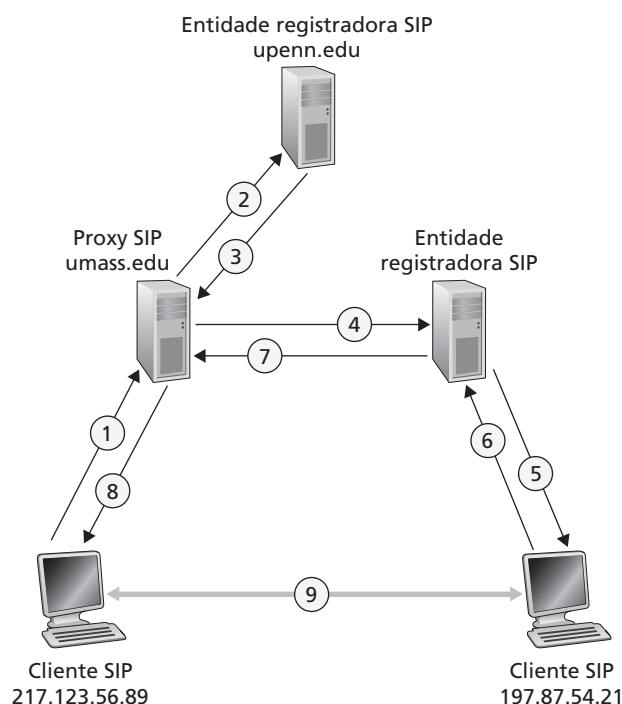


Figura 7.15 Inicialização de sessão envolvendo proxies e entidades registradoras SIP

(2) O proxy faz uma consulta DNS para a entidade registradora SIP de upenn.edu (que não é mostrada no diagrama) e então envia a mensagem ao servidor da registradora. (3) Como keith@upenn.edu não está mais registrado na entidade registradora de upenn, esta envia uma resposta de redirecionamento, indicando que é preciso tentar keith@eurecom.fr. (4) O proxy de umass envia uma mensagem INVITE à registradora SIP de eurecom. (5) A registradora de eurecom conhece o endereço IP de keith@eurecom.fr e repassa a mensagem INVITE ao hospedeiro 197.87.54.21, que está executando o cliente SIP de Keith. (6–8) Uma resposta SIP é devolvida por meio de registradoras/proxies ao cliente SIP em 217.123.56.89. (9) Então a mídia é enviada diretamente entre os dois clientes. (Também há uma mensagem de reconhecimento SIP, que não é mostrada.)

Nossa discussão sobre o SIP focalizou a inicialização de chamadas para chamadas de voz. Como o SIP é um protocolo de sinalização para inicializar e encerrar chamadas em geral, ele pode ser usado para chamadas de videoconferência, bem como para sessões de texto. Na verdade, o SIP tornou-se um componente fundamental em muitas aplicações de mensagem instantânea. Os leitores que quiserem saber mais sobre esse protocolo devem visitar o site Web de Henning Schulzrinne [Schulzrinne-SIP, 2009]. Em particular, nesse site você encontrará software de código-fonte aberto para clientes e servidores SIP [SIP software, 2009].

7.4.4 H.323

Como uma alternativa ao SIP, o H.323 é um padrão popular para audioconferência e videoconferência entre sistemas finais na Internet. Como ilustrado na Figura 7.16, o padrão também abrange a maneira como sistemas finais ligados à Internet se comunicam com telefones ligados às redes normais de telefonia de comutação de circuitos. (O SIP também faz isso, embora não tenhamos discutido esse assunto.) O gatekeeper H.323 é um dispositivo semelhante a uma entidade registradora SIP.

O padrão H.323 é uma especificação guarda-chuva que inclui as seguintes especificações:

- Uma especificação para o modo como os terminais negociam codificações comuns de áudio/vídeo. Como o H.323 suporta uma variedade de padrões de codificação de áudio e vídeo, é preciso um protocolo para permitir que os terminais comunicantes cheguem a um acordo quanto a uma codificação comum.
- Uma especificação para o modo como porções de áudio e vídeo são encapsuladas e enviadas à rede. Em particular, para essa finalidade o H.323 impõe o RTP.
- Uma especificação para o modo como terminais se comunicam com seus respectivos gatekeepers.

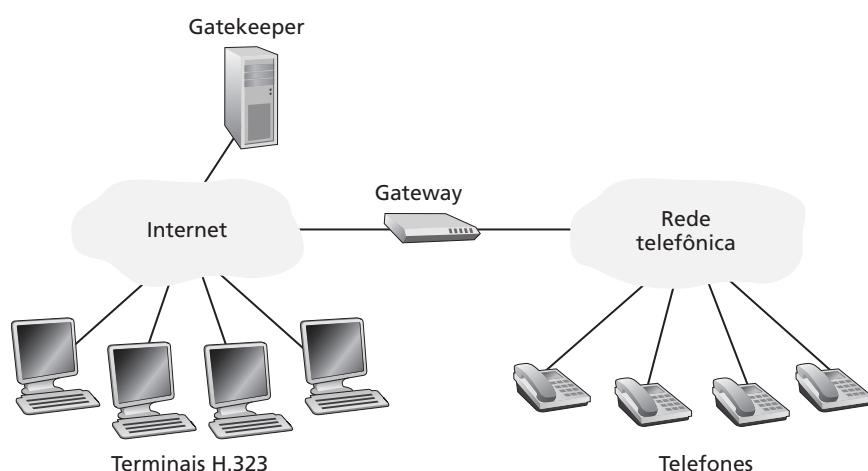


Figura 7.16 Sistemas finais H.323 ligados à Internet podem se comunicar com telefones ligados a uma rede telefônica de comutação de circuitos

Uma especificação para o modo como telefones por Internet se comunicam por meio do gateway com os telefones comuns na PSTN.

No mínimo, cada terminal H.323 tem de suportar o padrão de compressão de voz G.711. O G.711 usa PCM para gerar voz digitalizada a 56 kbps ou 64 kbps. Embora o H.323 exija que cada terminal seja habilitado para voz (por meio do G.711), capacidades de vídeo são opcionais. Como o suporte de vídeo é opcional, os fabricantes de terminais podem vender tanto terminais de voz mais simples como terminais mais complexos que suportam áudio e vídeo. Capacidades de vídeo para um terminal H.323 são opcionais. Todavia, se um terminal suportar vídeo, então ele deverá (no mínimo) suportar o padrão de vídeo QCIF H.261 (176 x 144 pixels).

O H.323 é um padrão guarda-chuva abrangente que, além dos padrões e protocolos que acabamos de descrever, impõe um protocolo de controle H.245, um canal de sinalização Q.931 e um protocolo RAS para registro no gatekeeper.

Concluímos esta seção destacando algumas das diferenças mais importantes entre H.323 e SIP.

O H.323 é um conjunto de protocolos completo, integrado verticalmente, para conferência multimídia: sinalização, registro, controle de admissão, transporte e codecs.

O SIP, por outro lado, aborda apenas inicialização e gerenciamento de sessão e é um componente separado. O SIP trabalha com RTP, mas não o impõe. Trabalha com codecs de voz G.711 e codecs de vídeo QCIF H.261, mas não os impõe. Pode ser combinado com outros protocolos e serviços.

O H.323 vem da ITU (telefonia), enquanto o SIP vem da IETF e toma emprestado muitos conceitos da Web, do DNS e do e-mail da Internet.

O H.323, como é um padrão guarda-chuva, é grande e complexo. O SIP usa o princípio KISS: mantenha a simplicidade, seu ignorante (keep it simple, stupid)!

Há uma excelente discussão sobre protocolos H.323, SIP e de voz sobre IP em geral, em [Hersent, 2000].

7.5 Fornecendo múltiplos classes de serviço

Nas seções anteriores, aprendemos como números de sequência, marcas de tempo, FEC, RTP e H.323 podem ser usados por aplicações de multimídia na Internet de hoje. CDNs representam uma solução para distribuição de conteúdo de multimídia no âmbito do sistema inteiro. Mas será que somente essas técnicas são suficientes para suportar aplicações de multimídia confiáveis e robustas, como um serviço de telefonia IP equivalente ao serviço atualmente oferecido pela rede telefônica? Antes de responder a essa pergunta, vamos lembrar novamente que a Internet de hoje oferece um serviço de melhor esforço a todas as suas aplicações, isto é, ela nada promete quanto à qualidade de serviço (QoS) que uma aplicação receberá. Uma aplicação receberá o nível de desempenho (por exemplo, atraso fim a fim e perda de pacotes) que a rede estiver capacitada a dar em determinado momento. Lembre-se também de que a Internet pública de hoje não permite que aplicações de multimídia sensíveis ao atraso requisitem algum tratamento especial. Como todos os pacotes recebem igual tratamento dos roteadores, incluindo os pacotes de áudio e vídeo sensíveis ao atraso, para arruinar a qualidade de uma chamada telefônica IP em andamento basta haver suficiente tráfego interferente (isto é, congestionamento na rede) que aumente de modo significativo o atraso e a perda percebidos por uma chamada telefônica IP.

Mas se o objetivo é fornecer um modelo de serviço que provê mais que um serviço “um tamanho serve todos” de melhor esforço da Internet atual, qual tipo de serviço exato deve ser fornecido? Um modelo simples de serviço aprimorado é dividir o tráfego em classes e prover diferentes níveis de serviço a estas classes diferentes de tráfego. Por exemplo, um ISP pode querer fornecer uma classe de serviço mais alta para uma voz de atraso sensível através de IP ou para tráfego de teleconferência (e cobrar a mais por esse serviço!) em vez de atender o tráfego como FTP ou HTTP. Estamos todos familiarizados com as diferentes classes de serviço do nosso dia a dia — passageiros de primeira classe recebem um serviço melhor do que os passageiros de classe executiva, que por sua vez recebem um serviço melhor do que os passageiros da classe econômica; VIPs imediatamente recebem entradas para eventos, enquanto todos os outros esperam na fila; idosos são reverenciados em alguns países e recebem lugares de honra e a melhor comida em um evento.

É importante observar que esses serviços diferenciados são fornecidos entre conglomerados de tráfego, por exemplo, entre classes de tráfego, não entre conexões individuais. Por exemplo, todos os passageiros da primeira classe são tratados da mesma maneira (sem que qualquer passageiro da primeira classe receba um tratamento diferencial), assim como os pacotes VoIP receberiam o mesmo tratamento dentro de uma rede, independentemente da conexão fim-fim a qual eles pertencem. Como veremos, ao lidar com um número pequeno de conglomerados de tráfego, ao invés de um grande número de conexões individuais, os mecanismos de rede que necessitam fornecer o melhor serviço, podem ser mantidos relativamente simples.

Os primeiros projetistas da Internet claramente tinham essa noção de múltiplas classes de serviço em mente. Relembre do campo tipo de serviço (ToS) no cabeçalho IPv4 na Figura 4.13. IEN123 [ISI 1979] descreve que o campo ToS, também presente em um ancestral do datagrama IPv4, como segue: “O tipo de serviço [campo] fornece uma indicação dos parâmetros abstratos da qualidade de serviço desejada. Esses parâmetros devem ser usados para guiar a escolha de parâmetros através de serviço quando um datagrama estiver sendo transmitido através de uma rede específica. Muitas redes oferecem serviços prioritários, os quais de alguma forma reservam mais atenção ao tráfego de prioridade alta do que aos outros.” Até mesmo há três décadas, a visão de fornecer diferentes níveis de serviço a diferentes níveis de tráfego estava evidente. No entanto, levou um longo período para que pudéssemos perceber essa visão.

Começaremos nosso estudo na Seção 7.5 considerando vários cenários que motivarão a necessidade de mecanismos específicos para o suporte de múltiplas classes de serviço. Veremos então dois tópicos importantes — escalonamento de camada de enlace e **classificação/regulação** de pacotes na Seção 7.5.2. Na Seção 7.5.3, veremos Diffserv — o padrão atual da Internet de fornecimento de serviços diferenciados.

7.5.1 Cenários motivadores

A Figura 7.17 mostra um cenário simples de rede que usaremos para ilustrar os componentes arquitetônicos mais importantes que têm sido propostos com a finalidade de fornecer suporte explícito para as necessidades de QoS de aplicações de multimídia. Suponha que dois fluxos de pacotes de aplicação se originem nos hospedeiros H1 e H2 em uma LAN e sejam destinados aos hospedeiros H3 e H4 em outra LAN. Os roteadores nas duas LANs são ligados por um enlace de 1,5 Mbps. Vamos admitir que as taxas das LANs sejam significativamente mais altas do que 1,5 Mbps e vamos focalizar a fila de saída do roteador R1; é aqui que ocorrerão atraso e perda de pacotes se a taxa agregada de envio de H1 e H2 exceder 1,5 Mbps. Vamos agora considerar diversos cenários; cada um deles nos dará importantes percepções a necessidade de mecanismos específicos para o suporte de classes múltiplas de serviço.

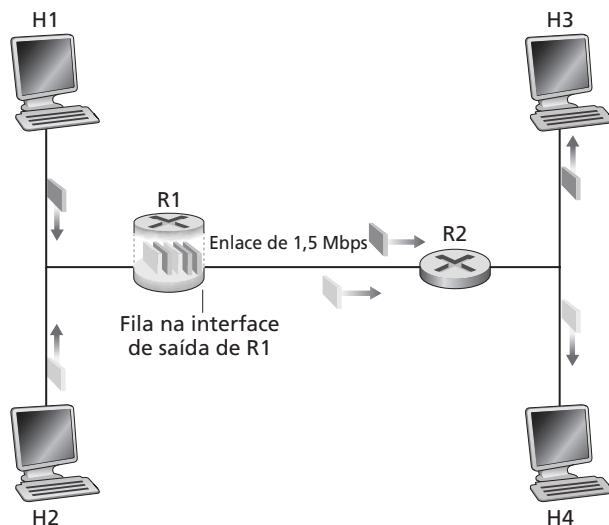


Figura 7.17 Uma rede simples com duas aplicações

Cenário 1: uma aplicação de áudio de 1 Mbps e uma transferência FTP

O cenário 1 está ilustrado na Figura 7.18. Nessa figura, uma aplicação de áudio de 1 Mbps (por exemplo, uma chamada de áudio com qualidade de CD) compartilha o enlace de 1,5 Mbps entre R1 e R2 com uma aplicação FTP que está transferindo um arquivo de H2 para H4. Na Internet de melhor esforço, os pacotes de áudio e FTP são misturados na fila de saída de R1 e (caracteristicamente) transmitidos na ordem primeiro a entrar/primeiro a sair (*first-in-first-out* — FIFO). Nesse cenário, uma rajada de pacotes da fonte FTP poderia potencialmente lotar a fila, fazendo com que pacotes IP de áudio sofressem atraso excessivo ou perda, devido ao transbordamento do buffer de R1. Como resolveríamos esse problema potencial? Dado que a aplicação FTP não tem limitação de tempo, poderíamos, por intuição, dar prioridade estrita aos pacotes de áudio em R1. Sob uma disciplina restritiva de escalonamento por prioridade, um pacote IP de áudio no buffer de saída de R1 seria sempre transmitido antes de qualquer pacote FTP nesse mesmo buffer de saída de R1. O enlace de R1 a R2 pareceria um enlace dedicado de 1,5 Mbps para o tráfego de áudio e o tráfego FTP usaria o enlace R1–R2 somente quando não houvesse nenhum tráfego de áudio na fila.

Para R1 distinguir os pacotes de áudio dos pacotes FTP em sua fila, cada pacote deve ser marcado como pertencente a uma dessas duas classes de tráfego. Lembre-se de que esse era o objetivo final do campo de tipo de serviço (*Type-of-service* — ToS) do IPv4 (veja a Seção 4.4.1). Por mais óbvio que pareça, esse é, então, nosso primeiro vislumbre dos mecanismos necessários para o fornecimento de classes múltiplas de tráfego:

Princípio 1: a marcação de pacotes permite que um roteador faça a distinção de pacotes pertencentes a diferentes classes de tráfego.

Cenário 2: uma aplicação de áudio de 1 Mbps e uma transferência FTP de alta prioridade

Nosso segundo cenário é apenas ligeiramente diferente do cenário 1. Suponha agora que o usuário FTP comprou de seu ISP um serviço de acesso à Internet de alto preço, ao passo que o usuário de áudio comprou um serviço de Internet barato, cujo preço é uma minúscula fração do outro serviço. Nesse caso, os pacotes de áudio do usuário do serviço barato deveriam ter prioridade em relação aos pacotes FTP? Pode-se afirmar que não. Aqui, o mais razoável seria distinguir pacotes com base no endereço IP do remetente. De modo mais geral, vemos que é necessário que um roteador classifique pacotes segundo algum critério, o que exige uma leve modificação no princípio 1:

Princípio 1 (modificado): a classificação de pacotes permite que um roteador faça a distinção de pacotes pertencentes a diferentes classes de tráfego.

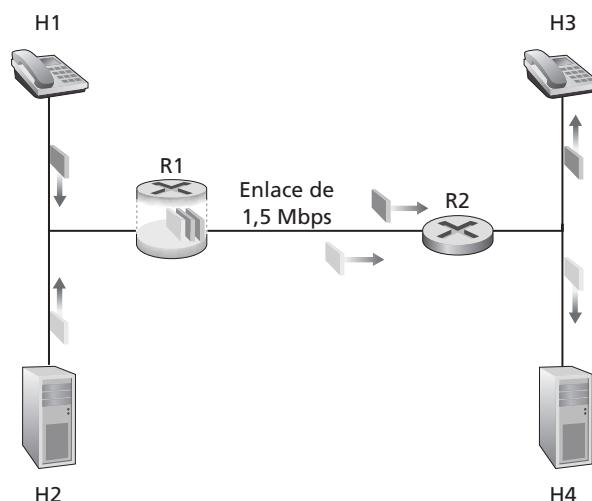


Figura 7.18 Aplicações concorrentes de áudio e FTP

A marcação explícita de pacotes é uma maneira pela qual os pacotes podem ser distinguidos. Contudo, a marca carregada por um pacote não determina por si só que o pacote receberá uma dada qualidade de serviço. A marcação é apenas um dos mecanismos para distinguir pacotes. O modo como um roteador distingue pacotes tratando-os de maneira diferenciada é uma decisão administrativa.

Cenário 3: uma aplicação de áudio malcomportada e uma transferência FTP

Suponha agora que, de alguma maneira (pela utilização de mecanismos que estudaremos nas seções subsequentes), o roteador saiba que deve dar prioridade a pacotes da aplicação de áudio de 1 Mbps. Como a taxa de saída do enlace é de 1,5 Mbps, mesmo que os pacotes FTP recebam prioridade mais baixa, ainda assim receberão, na média, um serviço de transmissão de 0,5 Mbps. Mas o que acontece se a aplicação de áudio começar a enviar pacotes a uma taxa de 1,5 Mbps ou mais alta (seja com má intenção, seja devido a um erro de aplicação)? Nesse caso, os pacotes FTP morrerão por inanição, isto é, não receberão nenhum serviço no enlace R1–R2. Problemas semelhantes ocorreriam se várias aplicações (por exemplo, várias chamadas de áudio), todas com a mesma prioridade, estivessem compartilhando a largura de banda de um enlace; um fluxo malcomportado poderia degradar e arruinar o desempenho dos outros fluxos. Idealmente, o que se quer é um certo grau de isolamento entre fluxos, para proteger um fluxo do mau comportamento de outro fluxo. Essa noção de proteger fluxos individuais uns dos outros dentro de uma classe de tráfego contradiz nossas observações anteriores de que pacotes de todos os fluxos dentro de uma classe devem ser tratados da mesma maneira. Na prática, pacotes dentro de uma classe são de fato tratados da mesma forma em roteadores dentro do núcleo da rede. No entanto, na borda de uma rede, os pacotes dentro de um determinado fluxo podem ser monitorados para se certificar que a taxa de conglomeração de um fluxo individual não excede um valor determinado.

Estas considerações originam nossa segundo o princípio:

Princípio 2: é desejável fornecer um certo grau de isolamento entre os fluxos de tráfego, de modo que um fluxo não seja afetado adversamente por outro fluxo de comportamento inadequado.

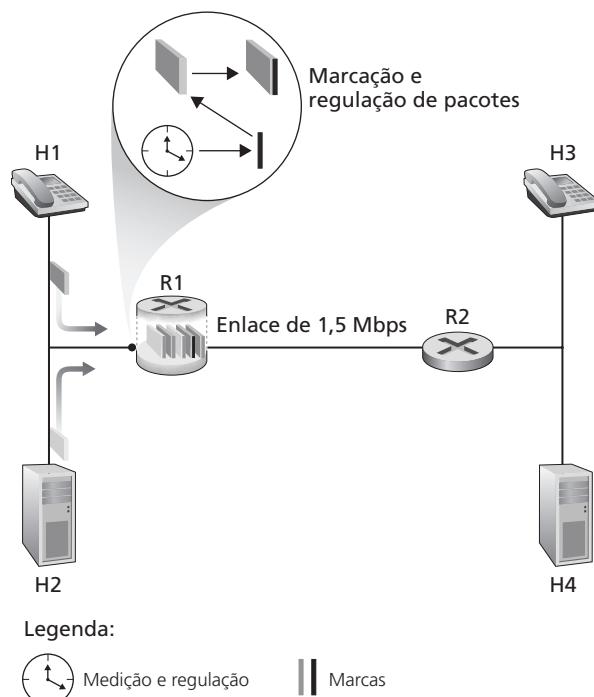


Figura 7.19 Regulação (e marcação) de fluxos de áudio e FTP

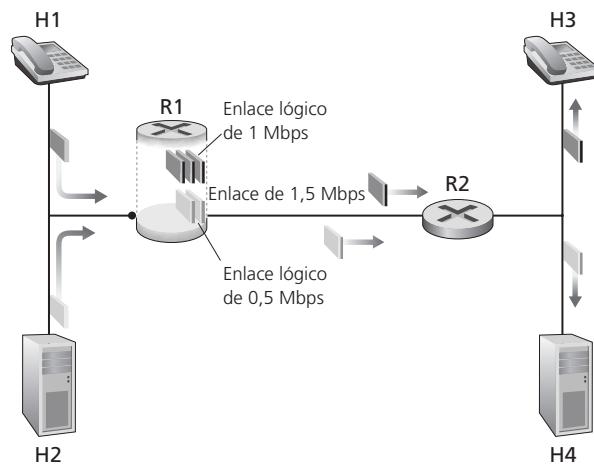


Figura 7.20 Isolamento lógico de fluxos de aplicações de áudio e de FTP

Na próxima seção, examinaremos diversos mecanismos específicos que fornecem isolamento entre fluxos. Observamos aqui que podem ser adotadas duas abordagens amplas. Com a primeira, é possível regular fluxos de tráfego, como mostra a Figura 7.19. Se um fluxo de tráfego deve se ajustar a certos critérios (por exemplo, o fluxo de áudio não deve exceder uma taxa de pico de 1 Mbps), então um mecanismo de regulação pode ser introduzido para assegurar que esse critério seja, de fato, observado. Se a aplicação regulada se comportar mal, o mecanismo de regulação executará alguma ação (por exemplo, descartará ou atrasará pacotes que estão violando o critério), de modo que o tráfego que está entrando na rede obedeça aos critérios. O mecanismo do tipo leaky bucket ('balde furado'), que examinaremos na seção seguinte, talvez seja o mecanismo de regulação mais amplamente usado. Na Figura 7.19, o mecanismo de classificação e marcação de pacotes (Princípio 1) e o mecanismo de regulação (Princípio 2) estão localizados conjuntamente na borda da rede, seja no sistema final, seja em um roteador de borda.

Uma abordagem alternativa para prover isolamento entre os fluxos de tráfego é deixar para o mecanismo de programação de pacotes na camada de enlace a tarefa de alocar explicitamente uma porção fixa da largura de banda de enlace a cada fluxo de aplicação. Por exemplo, o fluxo de áudio poderia ser alocado em R1 a 1 Mbps, e o fluxo de FTP 0,5 Mbps. Nesse caso, os fluxos de áudio e FTP perceberiam um enlace lógico com capacidade de 1 Mbps e 0,5 Mbps, respectivamente, como mostra a Figura 7.20.

Com imposição estrita de alocação de largura de banda na camada de enlace, um fluxo pode usar apenas a quantidade de largura de banda que lhe foi alocada; em particular, ele não pode utilizar largura de banda que não está sendo usada correntemente pelas outras aplicações. Por exemplo, se o fluxo de áudio silenciasse (se o interlocutor fizesse uma pausa e não gerasse nenhum pacote de áudio, por exemplo), ainda assim o fluxo FTP não conseguiria transmitir mais do que 0,5 Mbps pelo enlace R1–R2, mesmo que a alocação de largura de banda de 1 Mbps para o fluxo de áudio não estivesse sendo utilizada naquele momento. Por conseguinte, é desejável usar a largura de banda da maneira mais eficiente possível, permitindo que uma classe ou um fluxo utilize a largura de banda não usada por outro fluxo a qualquer instante. Essa consideração origina nosso terceiro princípio:

Princípio 3: ao fornecer isolamento de fluxos, é desejável que se usem os recursos (por exemplo, largura de banda de enlace e buffers) da maneira mais eficiente possível.

7.5.2 Mecanismos de escalonamento e regulação

Agora que estabelecemos princípios sobre os mecanismos necessários para o fornecimento de diferentes classes de serviço, vamos considerar dois dos mais importantes mecanismos — escalonamento e regulação — com mais detalhes.

Mecanismos de escalonamento

Lembre-se de que dissemos na Seção 1.3 e na Seção 4.3 que pacotes pertencentes a vários fluxos de rede são multiplexados e enfileirados para transmissão nos buffers de saída associados a um enlace. O modo como os pacotes enfileirados são selecionados para transmissão pelo enlace é conhecido como **disciplina de escalonamento do enlace**. Vamos agora examinar mais detalhadamente algumas das mais importantes disciplinas de escalonamento.

Primeiro a entrar/primeiro a sair (FIFO)

A Figura 7.21 mostra a representação do modelo de enfileiramento para a disciplina de escalonamento de enlace primeiro a entrar/primeiro a sair. Pacotes que chegam à fila de saída do enlace esperam pela transmissão se, naquele momento, o enlace estiver ocupado com a transmissão de outro pacote. Se não houver espaço de buffer suficiente para guardar o pacote que chega, a **política de descarte de pacotes** da fila então determinará se o pacote será descartado (perdido) ou se outros pacotes serão retirados da fila para dar espaço ao pacote que está chegando. Em nossa discussão a seguir, vamos ignorar o descarte de pacotes. Quando um pacote é transmitido integralmente pelo enlace de saída (isto é, recebe serviço), ele é retirado da fila.

A disciplina de escalonamento FIFO — também conhecida como FCFS (*first-come-first-served*) — primeiro a chegar/primeiro a ser atendido — seleciona pacotes para transmissão pelo enlace na mesma ordem em que eles chegaram à fila de saída do enlace. Todos estamos familiarizados com as filas FIFO em pontos de ônibus (particularmente na Inglaterra, onde parece que as filas atingiram a perfeição) ou em outras centrais de serviços, onde os clientes que chegam se juntam ao final da fila de espera, permanecem na ordem e então são atendidos quando chegam ao início da fila.

A Figura 7.22 mostra a fila FIFO em operação. As chegadas de pacotes são indicadas por setas numeradas acima da linha temporal superior; os números indicam a ordem em que os pacotes chegaram. As saídas de pacotes individuais são mostradas abaixo da linha temporal inferior. O tempo que um pacote passa no atendimento (sendo transmitido) é indicado pelo retângulo sombreado entre as duas linhas de tempo. Por causa da disciplina FIFO, os pacotes saem na mesma ordem em que chegaram. Note que, após a saída do pacote 4, o enlace permanece ocioso (uma vez que os pacotes 1 a 4 já foram transmitidos e retirados da fila) até a chegada do pacote 5.

Enfileiramento prioritário

Sob a **regra do enfileiramento prioritário**, pacotes que chegam ao enlace de saída são classificados em classes de prioridade na fila de saída, como mostra a Figura 7.23. Como discutimos na seção anterior, a classe de prioridade de um pacote pode depender de uma marca explícita que ele carrega em seu cabeçalho de pacote (por exemplo, o valor dos bits de ToS em um pacote IPv4), do endereço IP de sua origem ou destino, do número de porta de seu destino ou de outro critério. Cada classe de prioridade tipicamente tem sua própria fila. Ao escolher um pacote para transmitir, a disciplina de enfileiramento prioritário transmitirá um pacote da classe de prioridade mais alta cuja fila não esteja vazia (isto é, tenha pacotes esperando transmissão). A escolha entre pacotes da mesma classe de prioridade é feita, tipicamente, pelo método FIFO.

A Figura 7.24 ilustra a operação de uma fila prioritária com duas classes de prioridade. Os pacotes 1, 3 e 4 pertencem à classe de alta prioridade e os pacotes 2 e 5, à classe de baixa prioridade. O pacote 1 chega

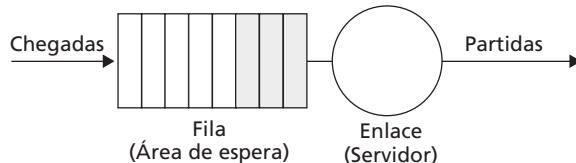


Figura 7.21 Representação do enfileiramento FIFO

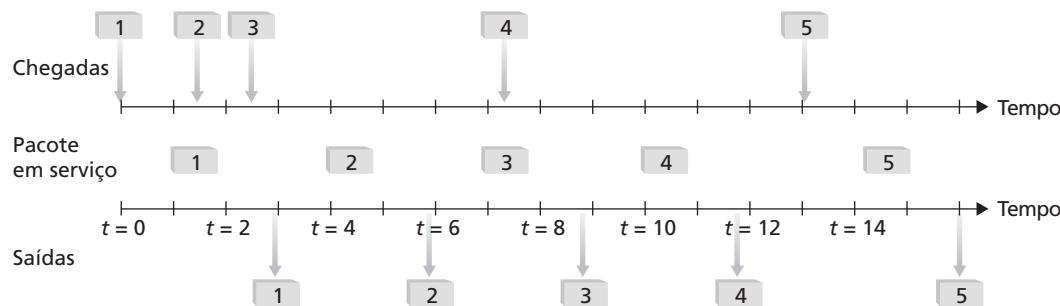


Figura 7.22 A fila FIFO em operação

e, encontrando o enlace vazio, inicia a transmissão. Durante a transmissão do pacote 1, os pacotes 2 e 3 chegam e são colocados nas filas de baixa prioridade e alta prioridade, respectivamente. Após a transmissão do pacote 1, o pacote 3 (um pacote de alta prioridade) é selecionado para transmissão, passando à frente do pacote 2 (que, mesmo tendo chegado mais cedo, é um pacote de baixa prioridade). Ao término da transmissão do pacote 3, começa a transmissão do pacote 2 (um pacote de baixa prioridade). O pacote 4 (um pacote de alta prioridade) chega durante a transmissão do pacote 2 (um pacote de baixa prioridade). Sob uma disciplina de enfileiramento prioritário não preemptiva, a transmissão de um pacote não será interrompida se já tiver começado. Nesse caso, o pacote 4 entra na fila para transmissão e começa a ser transmitido após a conclusão da transmissão do pacote 2.

Varredura cíclica e WQF (enfileiramento justo ponderado)

Sob a **disciplina de enfileiramento por varredura cíclica**, pacotes são classificados do mesmo modo que no enfileiramento prioritário. Contudo, em vez de haver uma prioridade estrita de serviço entre as classes, um escalonador de varredura cíclica alterna serviços entre as classes. Na forma mais simples de escalonamento por varredura cíclica, um pacote de classe 1 é transmitido, seguido por um pacote de classe 2, seguido por um pacote de classe 1, seguido por um pacote de classe 2 e assim por diante. Uma disciplina de enfileiramento de conservação de trabalho nunca permitirá que o enlace fique ocioso enquanto houver pacotes (de qualquer classe) enfileirados para transmissão. Uma **disciplina de varredura cíclica de conservação de trabalho** que procura um pacote de uma dada classe, mas não encontra nenhum, verifica imediatamente a classe seguinte na sequência da varredura cíclica.

A Figura 7.25 ilustra a operação de uma fila de duas classes por varredura cíclica. Nesse exemplo, os pacotes 1, 2 e 4 pertencem à classe 1 e os pacotes 3 e 5, à classe 2. O pacote 1 inicia a transmissão imediatamente após sua chegada à fila de saída. Os pacotes 2 e 3 chegam durante a transmissão do pacote 1 e, assim, entram na fila para transmissão. Após a transmissão do pacote 1, o escalonador de enlace procura um pacote

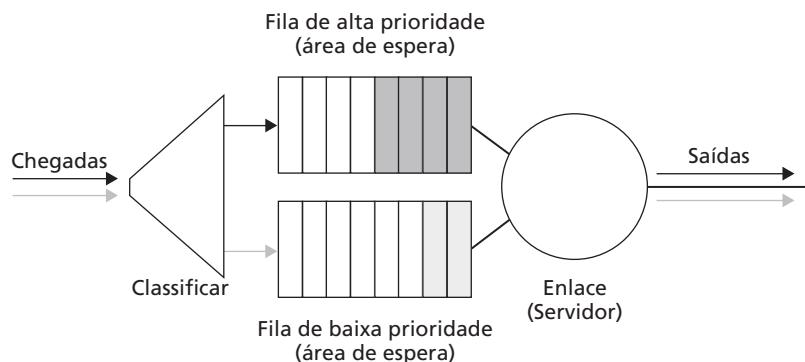


Figura 7.23 Modelo de enfileiramento prioritário

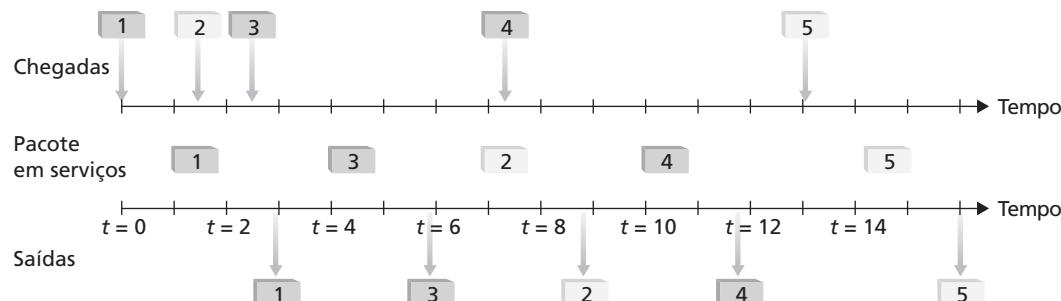


Figura 7.24 Operação do enfileiramento prioritário

de classe 2 e, então, transmite o pacote 3. Após a transmissão do pacote 3, o escalonador procura um pacote de classe 1 e, então, transmite o pacote 2. Após a transmissão do pacote 2, o pacote 4 é o único na fila; assim, ele é transmitido imediatamente após o pacote 2.

Uma abstração generalizada do enfileiramento por varredura cíclica que encontrou considerável utilização nas arquiteturas com QoS é a denominada disciplina de **enfileiramento justo ponderado** (*weighted fair queuing* — WFQ) [Demers, 1990; Parekh, 1993]. A WFQ é ilustrada na Figura 7.26. Os pacotes que chegam são classificados e enfileirados por classe em suas áreas de espera apropriadas. Como acontece no escalonamento por varredura cíclica, um programador WFQ atende às classes de modo cíclico — atende primeiramente à classe 1, depois à classe 2 e, em seguida, à classe 3; e então (admitindo que haja três classes) repete o esquema de serviço. A WFQ também é uma disciplina de enfileiramento de conservação de trabalho; assim, ao encontrar uma classe de fila vazia, ela imediatamente passa para a classe seguinte na sequência de atendimento.

A WFQ é diferente da varredura cíclica, pois cada classe pode receber uma quantidade de serviço diferenciado a qualquer intervalo de tempo. Em particular, a cada classe i é atribuído um peso w_i . A WFQ garante que, em qualquer intervalo de tempo durante o qual houver pacotes da classe i para transmitir, a classe i receberá uma fração de serviço igual a $w_i/(\sum w_j)$, onde o denominador é a soma de todas as classes que também têm pacotes enfileirados para transmissão. No pior caso, mesmo que todas as classes tenham pacotes na fila, a classe i ainda terá garantido o recebimento de uma fração $w_i/(\sum w_j)$ da largura de banda. Assim, para um enlace com taxa de transmissão R , a classe i sempre conseguirá uma vazão de, no mínimo, $R \cdot w_i/(\sum w_j)$. Descrevemos a WFQ em condições ideais, pois não consideraremos o fato de que os pacotes são unidades discretas de dados e que a transmissão de um pacote não será interrompida para dar início à transmissão de outro; [Demers, 1990] e [Parekh, 1993] discutem essa questão do empacotamento. Como veremos nas seções seguintes, a WFQ tem um papel fundamental nas arquiteturas com QoS. Ela também está disponível nos roteadores fabricados atualmente [Cisco QoS, 2009].

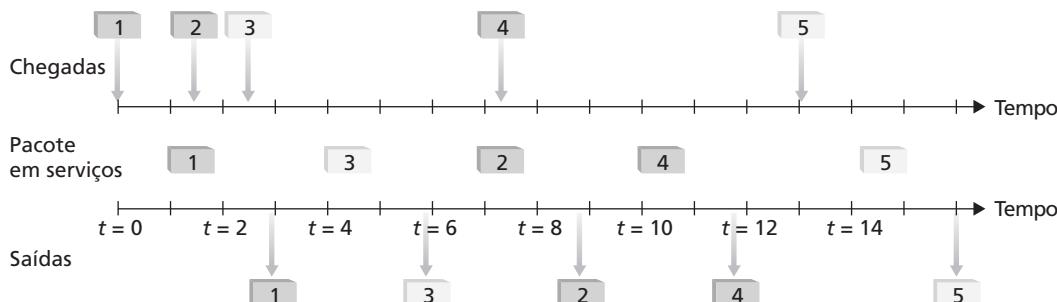


Figura 7.25 Operação de enfileiramento de duas classes por varredura cíclica

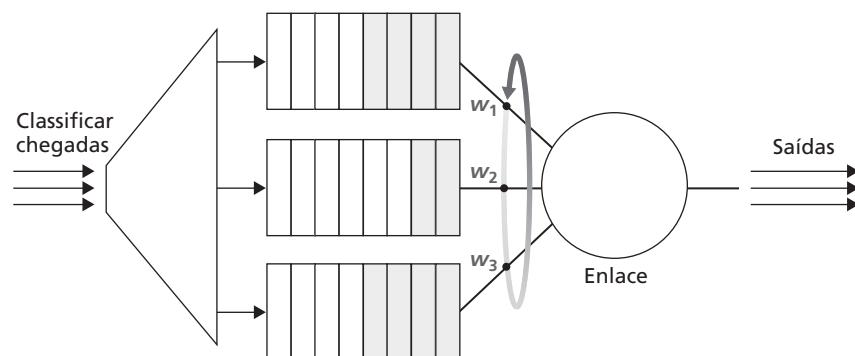


Figura 7.26 Enfileiramento justo ponderado (WFQ)

Regulação: o leaky bucket

Um dos nossos princípios na Seção 7.5.1, foi que o escalonamento, a regulação, o ajuste da taxa com a qual é permitido que um fluxo (suporemos que uma unidade de regulação é um fluxo na nossa discussão abaixo) injete pacotes na rede, são pedras fundamentais de qualquer arquitetura com QoS. Mas quais características da taxa de pacotes de um fluxo devem ser reguladas? Podemos identificar três critérios importantes de regulação diferentes entre si pela escala de tempo durante a qual o pacote é regulado:

Taxa média. A rede pode desejar limitar a taxa média durante um período de tempo (pacotes por intervalo de tempo) com a qual os pacotes de um fluxo podem ser enviados para dentro da rede. Uma questão crucial aqui é o intervalo de tempo durante o qual a taxa média será regulada. Um fluxo cuja taxa média está limitada a 100 pacotes por segundo é mais restringido do que uma fonte limitada a 6.000 pacotes por minuto, mesmo que ambos tenham a mesma taxa média durante um intervalo de tempo suficientemente longo. Por exemplo, a última limitação permitiria que um fluxo enviasse 1.000 pacotes em um dado intervalo de tempo de um segundo de duração, enquanto a limitação anterior desautorizaria esse comportamento de envio.

Taxa de pico. Enquanto a limitação da taxa média restringe a quantidade de tráfego que pode ser enviada para dentro da rede durante um período de tempo relativamente longo, uma limitação de taxa de pico restringe o número máximo de pacotes que podem ser enviados durante um período de tempo mais curto. Usando o mesmo exemplo anterior, a rede pode regular um fluxo a uma taxa média de 6.000 pacotes por minuto e, ao mesmo tempo, limitar a taxa de pico do fluxo a 1.500 pacotes por segundo.

Tamanho da rajada. A rede também pode limitar o número máximo de pacotes (a ‘rajada’ de pacotes) que podem ser enviados para dentro dela durante um intervalo de tempo extremamente curto. No limite, à medida que o comprimento do intervalo se aproxima de zero, o tamanho da rajada limita o número de pacotes que podem ser enviados instantaneamente para dentro da rede. Mesmo que seja impossível em termos físicos enviar vários pacotes para dentro da rede instantaneamente (afinal, cada enlace tem uma taxa de transmissão que não pode ser ultrapassada!), a abstração de um tamanho máximo de rajada é útil.

O mecanismo leaky bucket é uma abstração que pode ser usada para caracterizar esses limites da regulação. Como mostra a Figura 7.27, um leaky bucket é um balde que pode conter até b permissões. As permissões são adicionadas ao balde como segue. Novas permissões, que potencialmente podem ser adicionadas ao balde, estão sempre sendo geradas a uma taxa de r permissões por segundo. (Para facilitar, nesse caso admitimos que a unidade de tempo seja 1 segundo.) Se o balde estiver cheio com menos de b permissões quando é gerada uma permissão, a permissão recém-gerada será adicionada ao balde; caso contrário, ela será ignorada e o balde permanecerá cheio com as b permissões.

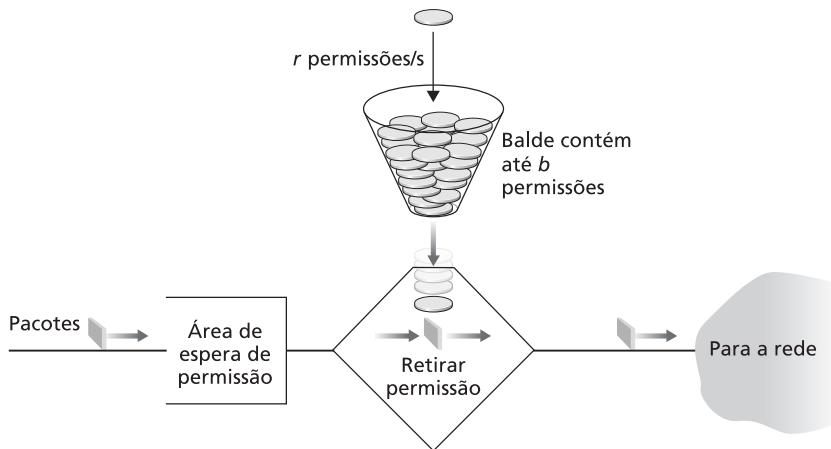


Figura 7.27 Regulação do leaky bucket

Vamos considerar agora como o leaky bucket pode ser usado para regular um fluxo de pacotes. Suponha que, antes de um pacote ser transmitido para dentro da rede, ele tenha de retirar uma permissão de dentro do balde de permissões. Se o balde de permissões estiver vazio, o pacote terá de esperar por uma permissão. (Uma alternativa é o pacote ser descartado, mas não vamos considerar essa opção aqui.) Vamos considerar agora como esse comportamento regula o fluxo de tráfego. Como pode haver no máximo b permissões no balde, o tamanho máximo da rajada para um fluxo regulado pela técnica do leaky bucket é b pacotes. Além disso, como a taxa de geração de permissões é r , o número máximo de pacotes que pode entrar na rede para qualquer intervalo de tempo t é $rt + b$. Assim, a taxa de geração de permissões, r , serve para limitar a taxa média de longo período com a qual pacotes podem entrar na rede. Também é possível usar leaky buckets (especificamente dois baldes em série) para regular a taxa de pico de um fluxo e a taxa média por um período de tempo; veja os exercícios ao final deste capítulo.

Leaky bucket e enfileiramento justo ponderado fornecem máximo atraso provável em uma fila

Logo examinaremos as abordagens denominadas Intserv e Diffserv para fornecimento de qualidade de serviço na Internet. Veremos que tanto a regulação do leaky bucket quanto o escalonamento WFQ podem desempenhar um papel importante. Vamos fechar esta seção considerando um enlace de saída de roteador que multiplexa n fluxos, cada um regulado por um leaky bucket com parâmetros b_i e r_i , $i = 1, \dots, n$, usando escalonamento WFQ. Usamos aqui o termo ‘fluxo’ de um modo um pouco impreciso para denominar conjuntos de pacotes que não são distinguidos uns dos outros pelo escalonador. Na prática, um fluxo pode conter o tráfego de uma única conexão fim a fim ou um conjunto de muitas conexões desse tipo; veja a Figura 7.28.

Lembre-se de que mencionamos, em nossa discussão sobre a WFQ, que cada fluxo i tem a garantia de receber uma parcela da largura de banda do enlace igual a, no mínimo, $R \cdot w_i / (\sum w_j)$, onde R é a taxa de transmissão do enlace em pacotes por segundo. Qual é, então, o atraso máximo que sofrerá um pacote enquanto espera ser atendido na WFQ (isto é, após ter passado pelo balde)? Vamos considerar o fluxo 1. Suponha que o balde de permissões do fluxo 1 esteja inicialmente cheio. Uma rajada de b_1 pacotes então chega ao regulador de leaky bucket para o fluxo 1. Esses pacotes retiram todas as permissões do balde (sem esperar) e, em seguida, juntam-se à área de espera WFQ para o fluxo 1. Como esses b_1 pacotes são atendidos a uma taxa de, no mínimo, $R \cdot w_i / (\sum w_j)$ pacotes por segundo, o último desses pacotes sofrerá um atraso máximo, d_{\max} , até que sua transmissão seja concluída, onde

$$d_{\max} = \frac{b_1}{R \cdot w_1 / \sum w_j}$$

A razão dessa fórmula é que, se houver b_1 pacotes na fila e os pacotes estiverem sendo atendidos (retirados) na fila a uma taxa de, no mínimo, $R \cdot w_i / (\sum w_j)$ por segundo, então a quantidade de tempo até que o

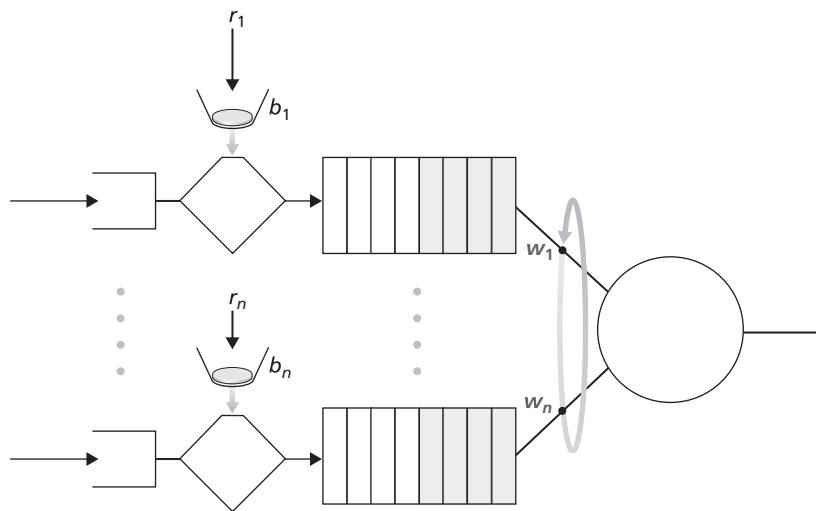


Figura 7.28 n fluxos multiplexados com leaky bucket com escalonamento WFQ

último bit do último pacote seja transmitido não poderá ser maior do que $b_i/(R \cdot w_i / (\sum w_j))$. Um exercício ao final deste capítulo solicita que você demonstre que, se $r_1 < R \cdot w_i / (\sum w_j)$, então d_{\max} é o atraso máximo que qualquer pacote do fluxo 1 pode sofrer na fila WFQ.

7.5.3 Diffserv

A arquitetura Diffserv [RFC 2475; Kilkki 1999] tem como objetivo fornecer um serviço diferenciado — isto é, a habilidade de lidar com as diferentes “classes” de tráfego de diferentes modos na Internet — e fazê-lo de modo escalável e flexível. A necessidade da escabilidade vem do fato que centenas de milhares de fluxos simultâneos de tráfego fonte-destino estão presentes em um roteador backbone da Internet. Veremos brevemente que essa necessidade é satisfeita ao colocarmos uma funcionalidade dentro do núcleo da rede, com uma operação mais completa de controle sendo implementada na borda da rede. A necessidade de flexibilidade surge pelo fato que novas classes podem aparecer e classes de serviços antigas podem se tornar obsoletas. A arquitetura Diffserv é flexível pois não define serviços específicos ou classes de serviço. Pelo contrário, Diffserv fornece componentes funcionais, ou seja, peças de uma arquitetura de rede, com as quais tais serviços podem ser feitos. Vamos agora examinar tais componentes mais detalhadamente.

Serviços diferenciados: um cenário simples

Com a finalidade de estabelecer uma estrutura para definir os componentes arquitetônicos do modelo de serviço diferenciado, vamos começar com a rede simples mostrada na Figura 7.29. Nesta seção, descreveremos um possível uso dos componentes da arquitetura Diffserv. Muitas outras variações são possíveis, como descreve o RFC 2475. Nosso objetivo aqui é apresentar uma introdução aos aspectos fundamentais do Diffserv, e não descrever o modelo arquitetônico com detalhes exaustivos. O leitor interessado em aprender mais sobre Diffserv pode consultar o livro muito abrangente de [Kilkki, 1999].

A arquitetura Diffserv consiste em dois conjuntos de elementos funcionais:

Funções de borda: classificação de pacotes e condicionamento de tráfego. Na borda de entrada da rede (isto é, ou em um hospedeiro habilitado a Diffserv que gera o tráfego, ou no primeiro roteador habilitado a Diffserv pelo qual o tráfego passa), os pacotes que chegam são marcados. Mais especificamente, o campo Differentiated Service (DS) do cabeçalho do pacote é configurado para algum valor. Por exemplo, na Figura 7.29, os pacotes que estão sendo enviados de H1 para H3 poderiam ser marcados em R1, ao passo que os pacotes enviados de H2 para H4 poderiam ser marcados em R2. A marca que um pacote

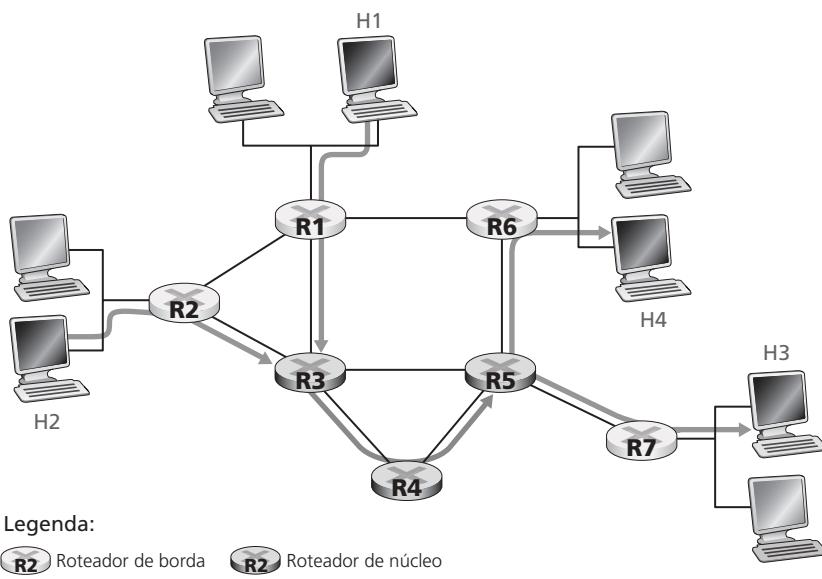


Figura 7.29 Um exemplo simples de rede Diffserv

recebe identifica a classe de tráfego à qual ele pertence. Assim, diferentes classes de tráfego receberão serviços diferenciados dentro do núcleo da rede.

Função central: envio. Quando um pacote marcado com DS chega a um roteador habilitado para Diffserv, ele é repassado até seu próximo salto de acordo com o **comportamento por salto** associado à classe do pacote. O comportamento por salto influencia a maneira pela qual os buffers e a largura de banda de um roteador são compartilhados entre as classes de tráfego concorrentes. Um dogma crucial da arquitetura Diffserv é que o comportamento por salto do roteador se baseará somente nas marcas dos pacotes, isto é, na classe de tráfego a que o pacote pertence. Assim, se os pacotes que estão sendo enviados de H1 para H3 na Figura 7.29 receberem a mesma marca dos que estão sendo enviados de H2 para H4, os roteadores da rede tratarão esses pacotes como um agregado, sem distinguir se eles se originam de H1 ou H2. Por exemplo, R3 não faria nenhuma distinção entre pacotes de H1 e H2 ao transmiti-los a R4. Portanto, a arquitetura de serviço diferenciado evita a necessidade de manter o estado do roteador para pares fonte–destino individuais — uma consideração importante para atender aos requisitos de escalabilidade discutidos no início desta seção.

Uma analogia poderia ser útil nesse ponto. Em muitos eventos sociais de grande escala (por exemplo, uma recepção com muitos convidados, uma boate ou discoteca, um concerto ou uma partida de futebol), as pessoas que participarão do evento adquirem algum tipo de ingresso. Há ingressos VIP para pessoas muito importantes; há ingressos para maiores de 18 anos (por exemplo, para eventos em que serão servidas bebidas alcoólicas), há credenciais que dão direito a visitar o camarim dos artistas; há ingressos especiais para repórteres e imprensa em geral; há até mesmo um ingresso comum para o cidadão comum. Esses ingressos são, em geral, adquiridos em bilheterias, isto é, na borda do evento. E é aqui, na borda, que são realizadas as operações que requerem intensa atividade computacional, como pagar pelo ingresso, verificar o tipo adequado de ingresso, comparar o ingresso com um documento de identidade. Além disso, pode haver um limite para o número de pessoas de um dado tipo que poderão entrar no evento. Se houver esse limite, as pessoas poderão ter de esperar antes de entrar. Uma vez lá dentro, o tipo de ingresso determina o tipo de serviço diferenciado recebido nos diversos locais do evento — um VIP recebe bebidas gratuitas, uma mesa melhor, refeição gratuita, dispõe de salas exclusivas e recebe serviço especial, ao passo que uma pessoa comum não pode entrar em certas áreas, paga sua bebida e recebe apenas o serviço básico. Em ambos os casos, o serviço recebido no evento depende exclusivamente do tipo de ingresso da pessoa. Além disso, todas as pessoas pertencentes à mesma classe recebem tratamento igual.

Classificação e condicionamento de tráfego do Diffserv

A Figura 7.30 apresenta uma visão lógica da função de classificação e marcação no roteador de borda. Os pacotes que chegam ao roteador de borda são primeiramente classificados. O classificador seleciona os pacotes com base nos valores de um ou mais campos de cabeçalho de pacote (por exemplo, endereço da fonte, endereço de destino, porta da fonte, porta de destino e protocolo ID) e dirige o pacote à função de marcação apropriada. A marca de um pacote é carregada dentro do campo DS [RFC 3260] no cabeçalho de pacote IPv4 ou IPv6. A intenção é que a definição do campo DS substitua as definições anteriores do campo de tipo de serviço do IPv4 e dos campos de classe de tráfego do IPv6 que discutimos no Capítulo 4.

Em alguns casos, um usuário final pode ter concordado em limitar sua taxa de envio de pacotes conforme um **perfil de tráfego** declarado. O perfil de tráfego poderia conter um limite à taxa de pico, bem como às rajadas do fluxo de pacotes, como vimos na anteriormente quando tratamos do mecanismo leaky bucket. Enquanto o usuário enviar pacotes para dentro da rede de um modo que esteja de acordo com o perfil de tráfego negociado, os pacotes receberão sua marca de prioridade e serão transmitidos ao longo de sua rota até o destino. Por outro lado, se o perfil de tráfego for violado, pacotes que estão fora do perfil poderão ser marcados de modo diferente, ajustados (por exemplo, atrasados de modo que seja observada a limitação da taxa máxima) ou descartados na borda da rede. O papel da **função de medição** mostrada na Figura 7.30 é comparar o fluxo de entrada de pacotes com o perfil de tráfego negociado. A decisão de imediatamente remarcar, transmitir, atrasar ou descartar um pacote é uma questão de política determinada pelo administrador da rede e não está especificada na arquitetura Diffserv.

Comportamentos por salto

Até aqui, examinamos as funções de borda da arquitetura Diffserv. O segundo componente fundamental dessa arquitetura envolve o comportamento por salto (PHB — *per-hop behavior*) realizado pelos roteadores habilitados para Diffserv. O PHB é definido de maneira cuidadosa, se bem que um tanto enigmática, como “uma descrição do comportamento de envio de um nó Diffserv que possa ser observado externamente aplicado a um comportamento agregado Diffserv em particular” [RFC 2475]. Explorando essa definição mais a fundo, podemos ver diversas considerações importantes nela embutidas:

- Um PHB pode resultar no recebimento de diferentes desempenhos por diferentes classes de tráfego (isto é, comportamentos de envio diferentes que possam ser observados externamente).
- Conquanto um PHB defina diferenças de desempenho (comportamento) entre classes, ele não determina nenhum mecanismo específico para conseguir esses comportamentos. Desde que os critérios de desempenho observados externamente sejam cumpridos, quaisquer mecanismos de implementação e quaisquer políticas de alocação de buffer/largura de banda podem ser usados. Por exemplo, um PHB não exigiria que fosse utilizada uma disciplina de enfileiramento de pacotes em particular (por exemplo, um enfileiramento prioritário *versus* um enfileiramento WFQ *versus* um enfileiramento FCFS)

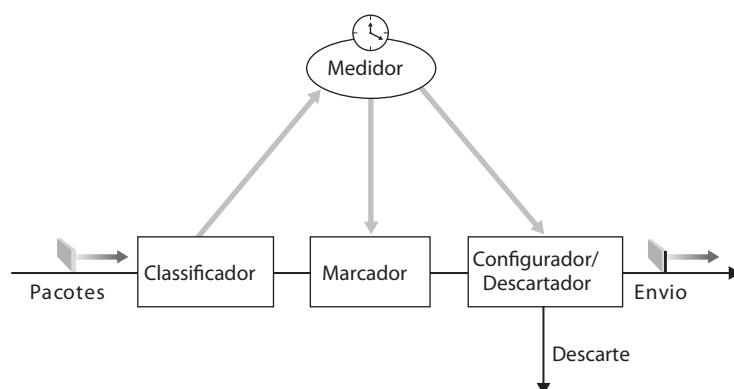


Figura 7.30 Visão lógica da classificação de pacotes e do condicionamento de tráfego no roteador de borda

para atingir um determinado comportamento. O PHB é o fim para o qual os mecanismos de alocação e implementação de recursos são os meios.

As diferenças de comportamento devem ser observáveis e, por conseguinte, mensuráveis.

Até agora foram definidos dois PHBs: um PHB de repasse acelerado (*expedited forwarding* — EF) [RFC 3246] e um PHB de repasse assegurado (*assured forwarding* — AF) [RFC 2597]:

O PHB de **repasse acelerado** especifica que a taxa de partida de uma classe de tráfego de um roteador deve ser igual ou maior do que uma taxa configurada. Isto é, durante qualquer intervalo de tempo, fica garantido que a classe de tráfego receba largura de banda suficiente de modo que a taxa de saída do tráfego seja igual ou maior do que essa taxa mínima configurada. Note que o comportamento por salto EF implica algum tipo de isolamento entre as classes de tráfego, pois essa garantia vigora independentemente da intensidade do tráfego de quaisquer outras classes que estejam chegando a um roteador. Assim, mesmo que as outras classes de tráfego estejam sobrecarregando os recursos do roteador e do enlace, recursos suficientes deverão ser mantidos disponíveis para a classe, para assegurar que ela receba sua taxa mínima garantida. Portanto, o EF fornece a uma classe uma abstração simples de um enlace com largura de banda de enlace mínima garantida.

O PHB de **envio assegurado** é mais complexo. O AF divide o tráfego em quatro classes e garante, a cada classe AF, o fornecimento de alguma quantidade mínima de largura de banda e de buffer. Dentro de cada classe, os pacotes ainda são repartidos em uma de três categorias de descarte preferencial. Quando ocorre um congestionamento dentro de uma classe AF, um roteador pode então descartar pacotes com base em seus valores de descarte preferencial. Para mais detalhes, consulte o RFC 2597. Variando a quantidade de recursos alocados a cada classe, um ISP pode fornecer diferentes níveis de desempenho às diversas classes de tráfego AF.

Retrospectiva do Diffserv

Durante os últimos 20 anos houve numerosas tentativas (a maior parte malsucedida) de introduzir QoS em redes de comutação de pacotes. As várias tentativas falharam, até agora, mais por razões econômicas e de compatibilidade do que por razões técnicas. Entre elas estão redes ATM fim a fim e redes TCP/IP. Vamos examinar algumas das questões envolvidas no contexto Diffserv (o qual estudaremos brevemente no próxima Seção).

Até aqui, admitimos que Diffserv é disponibilizados dentro de um único domínio administrativo. O caso mais típico é o do serviço fim a fim, que tem de ser configurado para várias ISPs localizadas entre os sistemas finais comunicantes. Para prover serviço Diffserv fim a fim, todas as ISPs entre o sistema final devem não apenas prover esse serviço, mas, acima de tudo, cooperar e fazer acordos para oferecer aos clientes finais um serviço verdadeiramente fim a fim. Sem esse tipo de cooperação, ISPs que vendem serviço Diffserv diretamente a clientes terão sempre de se justificar perante eles dizendo: "Sim, sabemos que você pagou um extra, mas não temos um acordo de serviço com suas ISPs de nível mais alto. Pedimos desculpas pelas muitas lacunas na sua chamada VoIP!"

Mesmo dentro de um domínio administrativo único, só a Diffserv não é suficiente para fornecer a qualidade da garantia de serviços a uma classe de serviço específico. Diffserv permite que diferentes classes de tráfego recebam diferentes níveis de desempenho. Se uma rede é gravemente subdimensionada, até uma classe de prioridade alta de tráfego pode receber uma performance ruim. Sendo assim, para ser eficaz, Diffserv precisa ser completada com o dimensionamento certo de rede (veja a Seção 7.3.5). Diffserv pode, no entanto, fazer com que um investimento ISP em uma capacidade de rede vá mais longe. Ao tornar os recursos disponíveis às classes de prioridade alta de tráfego sempre que necessário (ao custo das classes mais baixas de tráfego), o ISP pode emitir um desempenho de alta performance para essas classes de prioridade alta. Quando os recursos não são necessitados pelas classes de prioridade alta, eles podem ser usados pelas classes de tráfego de prioridade baixa (as quais provavelmente pagaram menos para esta classe de serviço).

Uma outra preocupação com esses serviços avançados é que eles precisam monitorar e possivelmente condicionar o tráfego, o que pode se revelar uma operação complexa e cara. Também é preciso cobrar pelo serviço de modo diferenciado, mas provavelmente por volume, e não por uma taxa mensal fixa, como fazem atual-

mente a maioria dos ISPs — um outro requisito caro para o ISP. Finalmente, se o Diffserv estivessem realmente disponíveis e a rede executasse somente sob carga moderada, na maior parte do tempo não se perceberia nenhuma diferença entre um serviço de melhor esforço e um serviço Diffserv. Na verdade, hoje, atraso é usualmente dominado pelas taxas de acesso e saltos de roteadores e não por atrasos de fila em roteadores. Imagine o infeliz cliente do Diffserv que pagou a mais por um serviço diferenciado, mas descobre que o serviço de melhor esforço oferecido por outros quase sempre apresenta o mesmo desempenho de um serviço diferenciado.

7.6 Fornecendo garantias de qualidade de serviços

Na seção anterior, vimos que a marcação de pacotes e regulação, isolamento de tráfego, e escalonamento a nível de enlace podem prover uma classe de serviço com melhor desempenho do que outra. Sob certas disciplinas de escalonamento, como escalonamento de prioridade, as classes inferiores de tráfego são basicamente “invisíveis” à classe de tráfego de prioridade mais alta. Com dimensionamento adequado da rede, a classe mais alta de serviço pode, de fato, atingir perda de pacote extremamente baixa e pequeno atraso — basicamente um desempenho semelhante ao circuito. Mas a rede pode *garantir* que um fluxo contínuo em uma classe de tráfego de alta prioridade continuará a receber tal serviço no decorrer da duração do fluxo usando apenas os mecanismos que descrevemos até agora? Não pode. Nesta seção, veremos por que os mecanismos adicionais da rede e protocolos são ainda necessários para prover qualidade com *garantias* de serviço.

7.6.1 Um exemplo motivador

Vamos voltar ao cenário da Seção 7.5.1 e considerar duas aplicações de áudio de 1 Mbps que transmitem seus pacotes através do enlace de 1,5 Mbps, conforme ilustrado na Figura 7.31. A taxa de dados combinados dos dois fluxos (2 Mbps) excede a capacidade do enlace. Mesmo com a classificação e marcação, isolamento de fluxos e o compartilhamento de largura de banda não utilizada, (a qual não existe nenhuma), esta é obviamente um caso perdido. Simplesmente não existe largura de banda suficiente para acomodar as necessidades de ambas as aplicações ao mesmo tempo. Se as duas aplicações compartilharem igualmente a largura de banda, cada uma receberia somente 0,75 Mbps. Visto de outra maneira, cada aplicação perderia 25 por cento de seus pacotes transmitidos. Essa QoS é tão inaceitavelmente baixa que ambas as aplicações de áudio são completamente inutilizadas: não há necessidade de transmitir quaisquer pacotes de áudio, em primeiro lugar.

Sabendo que essas duas aplicações, na Figura 7.31, não podem ser atendidas simultaneamente, o que a rede deve fazer? Permitir que as duas prossigam com uma QoS baixa desperdiça os recursos da rede em fluxos de aplicação que, no fim, não proveem nenhuma utilidade ao usuário final. A resposta, felizmente,

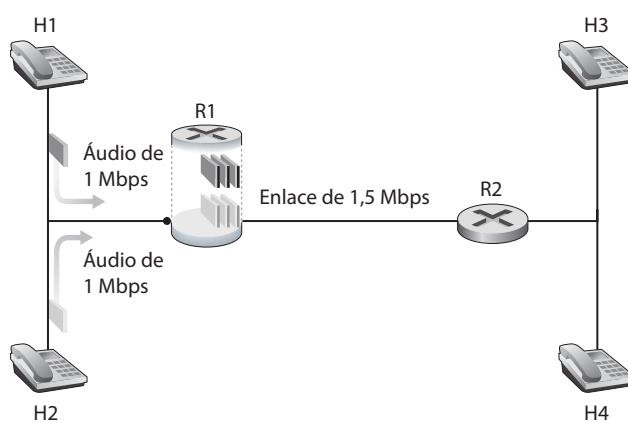


Figura 7.31 – Duas aplicações de áudio concorrentes sobrecarregando e enlace R1-R2

está evidente — um dos fluxos de aplicação deve ser bloqueado (ou seja, acesso negado à rede), enquanto o outro deve prosseguir, usando o 1 Mbps inteiro necessário pela aplicação. A rede telefônica é um exemplo de uma rede que realiza tal bloqueio de chamada — se as fontes necessárias (um circuito fim a fim no caso de uma rede telefônica) não podem ser alocadas para a chamada, ela é então bloqueada (impedida de entrar na rede) e o usuário recebe um sinal de ocupado. Em nosso exemplo, não há ganho em permitir que um fluxo entre na rede se ele não vai receber uma QoS suficiente para ser considerada utilizável. De fato, há um *custo* em admitir um fluxo que não recebe a QoS necessária, pois os recursos da rede estão sendo usados para suportar um fluxo que não oferece utilidade ao usuário final.

Admitindo ou bloqueando explicitamente os fluxos com base em suas necessidades de recursos e nas necessidades dos fluxos já admitidos, a rede pode garantir que os fluxos admitidos serão capazes de receber sua QoS solicitada. Implícita com a necessidade de prover uma QoS garantida a um fluxo está a necessidade do fluxo de declarar suas necessidades de QoS. O processo de um fluxo declarar sua exigência de QoS e a rede aceitar o fluxo (na QoS solicitada) ou bloqueá-lo é denominado processo de admissão de chamada. Isso é o nosso quarto princípio (além das três anteriores da Seção 7.5.1) em relação aos mecanismos necessários para prover a QoS.

Princípio 4: Se recursos suficientes nem sempre estiverem disponíveis e a QoS deve ser *garantida*, é necessário um processo de admissão de chamada no qual os fluxos declaram suas necessidades de QoS e, então, são admitidos à rede (na QoS solicitada) ou bloqueados da rede (se a QoS solicitada não pode ser fornecida pela rede).

7.6.2 Reserva de recurso, admissão de chamada e configuração de chamada

Nosso exemplo motivador enfatiza a necessidade de diversos novos mecanismos da rede e protocolos. Se uma chamada (um fluxo fim a fim) deve ser garantida, de modo a receber sempre uma QoS solicitada:

Reserva de recurso. A única maneira de *garantir* que uma chamada tenha os recursos (largura de banda de enlace, buffers) necessários para satisfazer a QoS desejada é alocar explicitamente esses recursos à chamada — um processo conhecido na linguagem de redes como reserva de recursos. Uma vez que os recursos são reservados, a chamada possui acesso sob demanda a esses recursos no decorrer de sua duração, independentemente das demandas de outras chamadas. Se uma chamada reserva e recebe uma garantia de x Mbps de largura de banda de enlace, e nunca transmite a uma taxa maior do que x , a chamada terá um desempenho sem perda e sem atraso.

Admissão de chamada. Se os recursos forem reservados, então a rede deve ter um mecanismo de chamadas para solicitar e reservar recursos — um processo conhecido como admissão de chamada. Visto que os recursos não são infinitos, uma chamada que faz uma solicitação de admissão de chamada será sua admissão negada, ou seja, bloqueada, se os recursos solicitados não estiverem disponíveis. Tal admissão de chamada é realizada pela rede telefônica — solicitamos recursos e discutimos um número. Se os circuitos (slots TDMA) necessários para completar a chamada estiverem disponíveis, os circuitos são alocados e a chamada é concluída. Se os circuitos não estiverem disponíveis, então a chamada é bloqueada e recebemos um sinal de ocupado. Uma chamada ocupada pode tentar novamente ganhar admissão à rede, mas não é permitido enviar tráfego à rede até que ela tenha completado com sucesso o processo de admissão de chamada.

Naturalmente, da mesma forma que o gerente do restaurante na Seção 1.3.1 não deve aceitar reservas para mais mesas do que o restaurante possui, um roteador que aloca uma largura de banda de enlace não deve alocar mais do que está disponível no enlace. Normalmente, uma chamada pode reservar somente uma fração da largura de banda do enlace, e um roteador pode alocar sua largura de banda do enlace para mais de uma chamada. Entretanto, a soma de largura de banda alocada a todas as chamadas deve ser menor do que a capacidade do enlace.

1. *Sinalização da configuração de chamada.* O processo de admissão de chamada descrito acima exige que uma chamada seja capaz de reservar recursos suficiente em cada e todo roteador da rede em seu caminho.

nho entre os remetentes e os receptores para garantir que sua exigência da QoS fim a fim seja satisfeita. Cada roteador deve determinar os recursos locais necessários pela sessão, considerar as quantidades de seus recursos que já estão comprometidos com outras sessões em andamento e determinar se ele possui recursos suficientes para satisfazer a exigência da QoS por salto da sessão naquele roteador sem violar as garantias de QoS feitas a uma sessão que já foi admitida. Um protocolo de sinalização é necessário para coordenar essas diversas atividades — a alocação por salto dos recursos locais, bem como a decisão fim a fim generalizada de se a ligação foi capaz ou não de reservar recursos necessários em cada e todo roteador no caminho fim a fim. Esse é o trabalho do protocolo de configuração de chamada.

A Figura 7.32 descreve o processo de configuração de chamada. Consideremos agora, em detalhes, as etapas envolvidas na admissão de chamada:

1. *Caracterização de tráfego e especificação da QoS desejada.* Para o roteador determinar se seus recursos são suficientes ou não para satisfazer a exigência da QoS de uma chamada, a chamada deve primeiro declarar sua exigência de QoS, bem como caracterizar o tráfego que será enviado à rede e para o qual é exigida uma garantia de QoS. Na arquitetura Intserv da Internet, o denominado Rspec (R de reserva) [RFC 2215] define que uma QoS específica está sendo solicitada por uma chamada; o denominado Tspec (T de tráfego) [RFC 2210] caracteriza o tráfego que o emissor enviará à rede ou que o receptor receberá da rede, respectivamente. A forma específica de Rspec e Tspec vão variar, dependendo do serviço requisitado, como é visto abaixo. Em redes ATM, a descrição do tráfego do usuário e os elementos de informação de parâmetro QoS carregam informações para propósitos similares aos elementos Tspec e Rspec, respectivamente.
2. *Sinalização para estabelecimento de ligação.* O descritor do tráfego e a requisição QoS de uma ligação precisam ser carregados aos roteadores em que os recursos serão reservados para a ligação. Na Internet, o protocolo RSVP [RFC 2210] é usado para este propósito dentro da arquitetura Intserv. Nas redes ATM, o protocolo Q2931b carrega a informação entre os comutadores das redes ATM e o ponto de chegada.
3. *Admissão de chamada por elemento.* Uma vez que o roteador recebe a especificação do tráfego e da QoS, ele precisa determinar se admitirá a chamada ou não. Esta decisão de admissão de chamada dependerá da especificação de tráfego, do tipo de serviço requisitado e dos compromissos de recursos existentes já feitos pelo roteador em chamadas em andamento. Lembre-se de que na Seção 7.5.3, por exemplo, vimos como a combinação de uma fonte controlada por leaky bucket e uma WFQ pode ser usada para determinar o atraso máximo de enfileiramento para aquela fonte. Admissão de chamada por elemento é mostrada na Figura 7.33.

Para maiores informações sobre configuração e admissão de chamadas, veja [Breslau, 2000; Roberts, 2004].

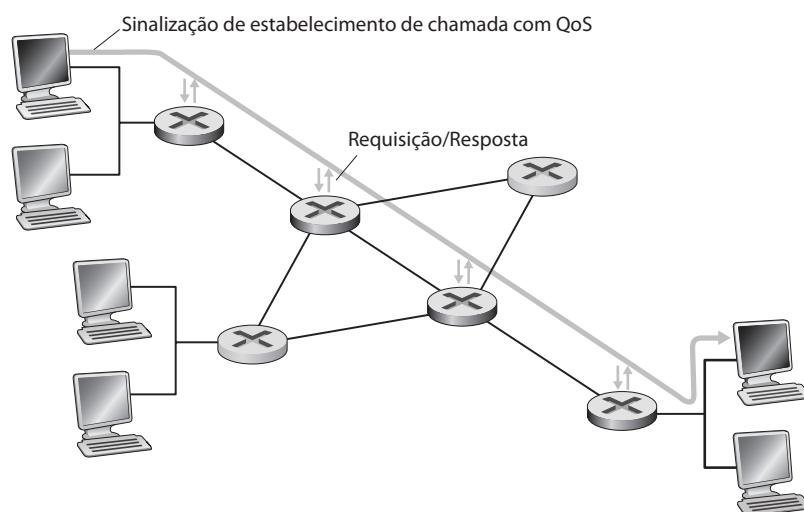


Figura 7.32 O processo de estabelecimento de chamada



Princípios na prática

O princípio do estado soft

O RSVP é usado para criar um estado (reservas de largura de banda) em roteadores, e é conhecido como um protocolo de estado soft. Em termos gerais, associamos o termo estado soft com abordagens de sinalização nas quais o estado criado se extingue (e é removido) por temporização, a menos que seja renovado periodicamente pelo recebimento de uma mensagem de sinalização (caracteristicamente da entidade que criou o estado no início) indicando que o estado deve continuar e permanecer instalado. Como um estado não renovado será extinto a qualquer momento, a sinalização de estado soft não requer remoção explícita de estado nem um procedimento para remover estados órfãos caso o criador de estados falhe. De modo semelhante, visto que criação de estado e mensagens de renovação serão seguidas por mensagens de renovação periódicas subsequentes, não é necessário sinalização confiável. O termo estado soft foi cunhado por Clark [Clark, 1988], que descreveu a noção de mensagens periódicas de renovação de estado enviadas por um sistema final e sugeriu que, com essas mensagens de renovação, o estado podia ser perdido devido a uma falha e então restaurado automaticamente por mensagens de renovação subsequentes — tudo transparente para o sistema final, e sem invocar nenhum procedimento de recuperação de falha:

... a informação de estado não seria crítica para a manutenção do tipo de serviço desejado associado com o fluxo. Em vez disso, aquele tipo de serviço seria imposto pelas extremidades, que enviariam mensagens periodicamente para assegurar que o tipo de serviço adequado está sendo associado com o fluxo. Dessa maneira, a informação de estado associada com o fluxo poderia ser perdida devido a uma falha sem causar a disruptão permanente das características de serviço que estão sendo usadas. Eu denomino esse conceito 'estado soft', e ele pode perfeitamente permitir que alcancemos nossas metas primárias de resiliência e flexibilidade...

Grosseiramente falando, a essência de uma abordagem de estado soft é a utilização pelo criador de estado do serviço de melhor esforço para criação e renovação de estado e remoção de estado por esgotamento de temporizador no mantenedor do estado. Abordagens de estado soft foram adotadas em inúmeros protocolos, entre eles RSVP, PIM (Seção 4.7), SIP (Seção 7.4.3) e IGMP (Seção 4.7) e em tabelas de repasse e pontes transparentes (Seção 5.6).

Sinalização de estado hard adota a abordagem inversa do estado soft — o estado criado permanece efetivo a menos que explicitamente removido pelo recebimento de uma mensagem de remoção de estado enviada pelo criador de estado. Visto que o estado permanece efetivo a menos que explicitamente removido, a sinalização de estado hard requer um mecanismo para remover um estado órfão que permanece após um criador de estado falhar ou sair sem ter removido o estado. De modo semelhante, uma vez que criação e remoção de estado são realizadas apenas uma vez (e sem renovação de estado ou esgotamento de temporização de estado), é importante que o criador de estado saiba quando o estado foi criado ou removido. Assim, protocolos de sinalização confiáveis (e não de melhor esforço) são comumente associados com protocolos de estado hard. Então, grosseiramente falando, a essência de uma abordagem de estado hard é a criação e remoção confiáveis e explícitas de informação de estado. Abordagens de estado hard foram adotadas em protocolos como ST-II [Partridge, 1992; RFC 1190] e Q.2931 [ITU-T Q.2931, 1994].

O RSVP oferece remoção de reserva explícita (se bem que opcional) desde que foi concebido.

Sinalização confiável baseada em ACK foi introduzida como uma extensão do RSVP no [RFC 2961] e também foi sugerida em [Pan, 1997]. Assim, o RSVP adotou, por opção, alguns elementos de uma abordagem de sinalização de estado hard. Para uma discussão e comparação de protocolos de estado soft versus estado hard, veja [Ji, 2003].

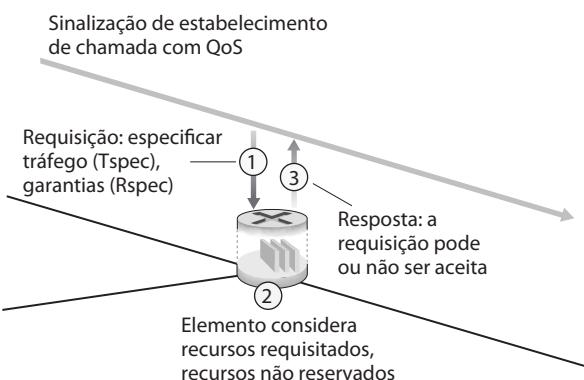


Figura 7.33 Comportamento de chamada por elemento

7.6.3 QoS garantida na Internet: Intserv e RSVP

A arquitetura de serviços integrados (Intserv) é um ambiente desenvolvido dentro da IETF para fornecer garantias individualizadas de QoS a sessões de aplicações individuais na Internet. A especificação de garantia de serviços Intserv, definida em [RFC 2212], determina limites estáveis (matematicamente comprováveis) sobre os atrasos de enfileiramentos que ocorrerão com um pacote em um roteador. Enquanto os detalhes por trás de um serviço garantido são complexos, a base da ideia é bem simples. A uma primeira aproximação, a caracterização de uma fonte de tráfego é dada por um leaky bucket (veja Seção 7.5.2) com parâmetros (r, b) e o serviço requisitado é caracterizado por uma velocidade de transmissão R , à qual os pacotes serão transmitidos. Em essência, uma chamada que requisita um serviço garantido está requisitando que seus bits nos pacotes sejam garantidos em uma taxa de transmissão de R bits/seg. Dado que o tráfego é especificado usando uma caracterização leaky bucket, e uma velocidade garantida de R está sendo requisitada, também é possível limitar o atraso máximo de enfileiramento no roteador. Lembre-se de que com a caracterização de tráfego leaky bucket, a soma do tráfego (em bits) gerado através de qualquer intervalo de tempo t é limitada por $rt + b$. Lembre-se também da Seção 7.5.2, onde vimos que quando uma fonte leaky bucket é alimentada em uma fila que garante que o tráfego enfileirado será servido a uma velocidade de R bits por segundo, o máximo de atraso de enfileiramento que o pacote sofrerá estará limitado por b/R , desde que R seja maior que r . Outra forma de garantia de serviço Intserv também foi definida, conhecida como carga de serviço controlada, que especifica que uma chamada receberá “uma qualidade de serviço aproximando a QoS que o fluxo que receberia de um elemento de rede descarregado” [RFC 2211].

O Protocolo Reserva de Recurso (RSVP) [RFC 2205; Zhang, 1993] é um protocolo de sinalização da Internet que pode ser usado para efetuar a sinalização de chamada exigida pela Intserv. O RSVP também foi usado juntamente com DiffServ para coordenar as funções DiffServ através de múltiplas redes, e também foi estendido e usado com um protocolo de sinalização em outras circunstâncias, talvez mais notavelmente na forma de RSVP-TE [RFC 3209] para sinalização MPLS, como visto na Seção 5.8.2.

No contexto Intserv, o protocolo RSVP permite que aplicações reservem a banda para o seu fluxo de dados. É usado por um hospedeiro, em benefício de um fluxo de dados de uma aplicação, para requisitar uma quantia específica de banda da rede. O RSVP também é usado por roteadores para o envio de pedidos de reserva de banda. Para implementar o RSVP, o software do RSVP precisa estar presente nos receptores, transmissores e roteadores ao longo do caminho fim-fim mostrado na Figura 7.32. As duas principais características do RSVP são:

Fornecer reservas para banda em árvores multicast, com o unicast sendo tratado como um caso degenerado de multicast. Isso é particularmente importante para aplicações multimídia como transmissão de fluxo contínuo de TV através de IP, onde muitos receptores podem querer receber o mesmo tráfego de multimídia enviado de uma mesma origem.

É baseado no receptor; ou seja, o receptor de um fluxo de dados inicia e mantém a reserva de recursos usada por aquele fluxo. A visão inovadora e centrada no recebimento usada pelo RSVP coloca os recep-

tores no controle do tráfego que recebem, por exemplo, permitindo que receptores diferentes recebam e vejam uma multimídia multicast em diferentes resoluções. Isto é constrastante se comparado à visão de sinalização centrada no transmissor adotada na recomendação Q2931b do ATM.

O padrão RSVP [RFC 2205] não especifica como a rede fornece a banda reservada aos fluxos de dados. É simplesmente um protocolo que permite que as aplicações reservem a borda necessária no enlace. Uma vez que as reservas estão no lugar certo, é função dos roteadores na Internet fornecer a banda reservada aos fluxos de dados. Este suprimento provavelmente seria feito usando os mecanismos de escalonamento e regulação (leaky bucket, escalonamento prioritário, enfileiramento ponderado justo) vistos na Seção 7.5. Para mais informações sobre RSVP, veja [RFC 2205; Zhang, 1993] e o material eletrônico online associado a este livro.

7.7 Resumo

A rede multimídia talvez seja um dos desenvolvimentos mais interessantes (com um potencial que ainda não se realizou totalmente) da Internet de hoje. Pessoas em todo o mundo estão passando menos tempo diante de seus aparelhos de rádio ou televisão e usando mais a Internet para receber transmissões de áudio e vídeo, ao vivo ou pré-gravadas. À medida que o acesso de alta velocidade for invadindo mais e mais residências, essa tendência continuará — os habituados aos sofás de todo o mundo acessarão seus programas favoritos em vídeo por meio da Internet, e não pelos canais tradicionais de distribuição *broadcast*. Além da distribuição de áudio e vídeo, a Internet está sendo usada para transmitir chamadas telefônicas. De fato, nos próximos dez anos ela poderá vir a substituir o quase obsoleto sistema de telefonia de comutação de circuitos em muitos países. A Internet não somente proverá serviço telefônico mais barato, mas também oferecerá numerosos serviços de valor agregado, como videoconferência, lista telefônica on-line, serviço de mensagens de viva-voz e integração com a Web.

Na Seção 7.1, classificamos aplicações de multimídia em três categorias: áudio e vídeo armazenados de fluxo contínuo; transmissão um-para-muitos de áudio e vídeo em tempo real e áudio e vídeo interativos em tempo real. Enfatizamos que aplicações multimídia são sensíveis a atraso e tolerantes à perda — características muito diferentes das aplicações de conteúdo estático, que são tolerantes ao atraso e intolerantes à perda. Discutimos também alguns obstáculos que o serviço de melhor esforço da Internet apresenta às aplicações de multimídia. Avaliamos diversas propostas para superar esses obstáculos, incluindo o simples aprimoramento da infraestrutura de rede existente (adicionando mais largura de banda, mais memórias temporárias de rede e mais nós CDN e disponibilizando o *multicast*), agregar mais funcionalidades à Internet, de modo que as aplicações possam reservar recursos fim a fim (para que a Internet possa honrar essas reservas) e, finalmente, introduzir classes de serviços para prover diferenciação de serviços.

Da seção 7.2 à 7.4, examinamos arquiteturas e mecanismos para rede multimídia com serviço de melhor esforço. Na Seção 7.2, avaliamos diversas arquiteturas para fluxo contínuo de áudio e vídeo armazenados. Discutimos interação com o usuário — como pausa/reinício, reposicionamento e avanço rápido visual — e apresentamos uma introdução ao RTSP, um protocolo que provê interação cliente-servidor para aplicações de fluxo contínuo. Na Seção 7.3, examinamos como aplicações interativas em tempo real podem ser projetadas para rodar sobre uma rede de serviço de melhor esforço. Vimos como uma combinação de buffers clientes, números de sequência de pacotes e marcas de tempo pode aliviar os efeitos da variação de atraso induzida pela rede. Também vimos como um CDN facilita o armazenamento do fluxo contínuo de multimídia ao pressionar proativamente a multimídia aos servidores CDN localizados próximos dos pontos de chegada do usuário.

Na Seção 7.5, introduzimos como vários mecanismos de rede (disciplinas de escalonamento de nível de enlace e condicionamento de tráfego) podem ser usados para fornecer serviços diferenciados entre diversas classes de tráfego. Finalmente, na Seção 7.6, investigamos como uma rede pode fazer garantias de qualidade de serviço às chamadas admitidas na rede. Aqui ainda foram necessários novos mecanismos de redes e de protocolos, incluindo reserva de recursos, admissão de chamada e sinalização de chamada. Juntos, esses novos elementos de rede fazem com que o QoS garantido de rede do futuro seja bem diferente (e muito mais complexo) do que a Internet de melhor esforço atual.



Exercícios de fixação

Questões de revisão

Seções 7.1 a 7.2

- O que significa interatividade no caso de fluxo contínuo de áudio/vídeo? E no caso de áudio/vídeo interativo em tempo real?
- Foram discutidos três campos para aprimorar a Internet de modo que ela possa dar melhor suporte às aplicações de multimídia. Faça um breve resumo das visões de cada campo. Qual deles você defende?
- Quais são algumas proporções típicas de compressão (proporção do número de bits em um objeto não comprimido ao número de bits na versão comprimida daquele objeto) para aplicações de imagem e áudio, e as técnicas de compressão discutidas na Seção 7.1?
- As figuras 7.1 e 7.2 apresentam dois esquemas para fluxo contínuo de mídia armazenada. Quais as vantagens e as desvantagens de cada esquema?

Seções 7.3 a 7.4

- Qual é a diferença entre atraso fim a fim e variação de atraso? Quais são as causas da variação de atraso?
- Por que um pacote recebido após seu tempo de reprodução programado é considerado perdido?
- Na Seção 7.3, descrevemos dois esquemas FEC. Faça um pequeno resumo deles. Ambos os esquemas aumentam a taxa de transmissão da corrente adicionando sobrecarga. A intercalação também aumenta a taxa de transmissão?
- Qual o papel do DNS em uma CDN? O DNS tem que ser modificado para dar suporte a uma CDN? Qual

informação, se existe alguma, a CDN deve fornecer ao DNS?

- Que informação é necessária para o dimensionamento de uma rede para que um determinado nível de qualidade seja alcançado?
- Como as diferentes correntes RTP em sessões diferentes são identificadas por um receptor? Como as diferentes correntes internas à mesma sessão são identificadas? Como são distinguidos os pacotes RTP e RTCP (como parte da mesma sessão)?
- Três tipos de pacotes RTCP foram descritos na Seção 7.4. Faça um pequeno resumo da informação contida em cada um desses tipos de pacotes.
- Qual é o papel de um registro SIP? Como o papel de um registro SIP é diferente de um home agent em IP móvel?

Seções 7.6 a 7.9

- Na Seção 7.5, discutimos o enfileiramento prioritário não preemptivo. O que seria um enfileiramento prioritário preemptivo? O enfileiramento prioritário preemptivo teria sentido em redes de computadores?
- Dê um exemplo de disciplina de escalonamento que não é conservadora de trabalho.
- Dê um exemplo de enfileiramentos que você vivencia em sua rotina diária, com disciplina FIFO, prioridade, RR e WFQ.
- Cite algumas dificuldades associadas ao modelo Intserv e à reserva de recursos por fluxo.



Problemas

- Navegue na Web e ache duas páginas que contenham fluxo contínuo de áudio e/ou vídeo armazenado(s). Para cada produto, use Wireshark para determinar:
 - se são usados metarquivos;
 - se o áudio/vídeo é enviado por UDP ou TCP;
 - se é usado RTP;
 - se é usado RTSP.
- Considere o buffer cliente mostrado na Figura 7.3. Suponha que o sistema de transmissão use a terceira opção, isto é, o servidor lança a mídia no socket o mais rapidamente possível. Suponha que a largura de banda TCP disponível seja $>> d$ na maior parte do

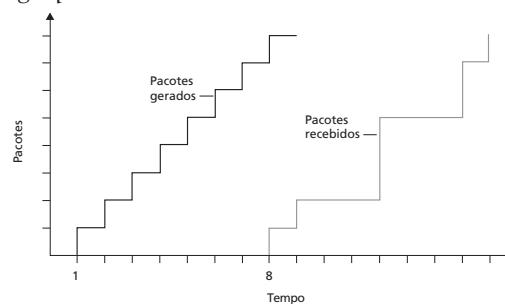
tempo. Suponha também que o buffer cliente pode conter somente cerca de um terço da mídia. Descreva como $x(t)$ e o conteúdo do buffer cliente evoluem com o tempo.

- O buffer de recebimento TCP e o buffer cliente do transdutor são a mesma coisa? Se não são, como eles interagem?
- No exemplo de telefone por Internet da Seção 7.3, seja h o número total de bytes de cabeçalho adicionado a cada porção, incluindo os cabeçalhos UDP e IP.
 - Admitindo que um datagrama IP seja emitido a cada 20 milissegundos, qual é a taxa de trans-

- missão em bits por segundo para os datagramas gerados por um lado dessa aplicação?
- b.** Qual é um valor típico de h quando é usado RTP?
- 5.** Considere o procedimento descrito na Seção 7.3 para estimar o atraso médio d_i . Suponha que $u = 0,1$. Seja $r_1 - t_1$ o atraso da amostra mais recente, seja $r_2 - t_2$ o atraso da amostra mais recente seguinte e assim por diante.
- a.** Para uma dada aplicação de áudio, suponha que quatro pacotes tenham chegado ao receptor com atrasos de amostra de $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$, $r_1 - t_1$. Expressa a estimativa de atraso d em termos das quatro amostras.
- b.** Generalize sua fórmula para n atrasos de amostra. Para a fórmula da parte (b), suponha que n tenda ao infinito e dê a fórmula resultante. Comente por que esse procedimento de determinação de média é denominado média móvel exponencial.
- c.** Para a fórmula da parte b, deixe n se aproximar do infinito e dê a fórmula resultante. Comente por que esse procedimento de média é chamado de média exponencial móvel.
- 6.** Repita as partes (a) e (b) da questão anterior para a estimativa do desvio médio do atraso.
- 7.** No exemplo do telefone por Internet na Seção 7.3, apresentamos um procedimento on-line (média móvel exponencial) para estimar atraso. Neste problema examinaremos um procedimento alternativo. Seja t_i a marca de tempo do i -ésimo pacote recebido; seja r_i o tempo em que o i -ésimo pacote é recebido. Seja d_n nossa estimativa do atraso médio após o recebimento do n -ésimo pacote. Após a recepção do primeiro pacote, estabelecemos a estimativa de atraso como $d_1 = r_1 - t_1$.
- a.** Suponha que gostaríamos que $d_n = (r_1 - t_1 + r_2 - t_2 + \dots + r_n - t_n)/n$ para todo n . Deduza uma fórmula recursiva para d_n em termos de d_{n-1} , r_n e t_n .
- b.** Descreva por que, para a telefonia por Internet, a estimativa de atraso descrita na Seção 7.3 é mais apropriada do que a estimativa esboçada na Parte a.
- 8.** Compare o procedimento descrito na Seção 7.3 para a estimativa do desvio médio do atraso com o procedimento descrito na Seção 3.5 para a estimativa do tempo de ida e volta. O que os procedimentos têm em comum? Em que são diferentes?
- 9.** Considere a estratégia de reprodução adaptativa descrita na Seção 7.3.
- a.** Como dois pacotes sucessivos recebidos no destino podem ter marcas de tempo que diferem em

mais de 20 milissegundos se os dois pacotes pertencem à mesma rajada de voz?

- b.** Como o receptor pode usar números de sequência para determinar se um pacote é o primeiro de uma rajada de voz? Seja objetivo em sua resposta.
- 10.** Considere a figura abaixo (que é similar a Figura 7.5). Um transmissor começa a enviar áudio empacotado periodicamente em $t = 1$. O primeiro pacote chega quando o transmissor está em $t = 8$.

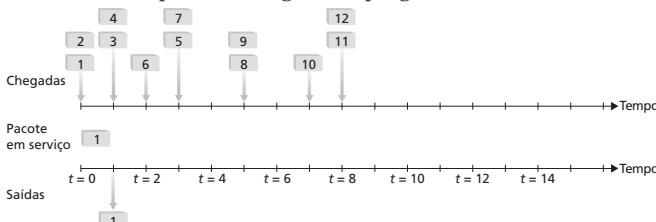


- a.** Quais são os atrasos (do transmissor ao receptor, ignorando qualquer atraso de transmissão) dos pacotes 2-8? Observe que cada segmento de linha vertical e horizontal na figura tem o comprimento de 1, 2 ou 3 unidades de tempo.
- b.** Se a transmissão de áudio começa assim que o receptor está em $t = 8$, qual dos primeiros oito pacotes enviados não chegará em tempo para a transmissão?
- c.** Se a transmissão de áudio começa assim que o receptor está em $t = 9$, qual dos primeiros oito pacotes enviados não chegará em tempo para a transmissão?
- d.** Qual o mínimo de atraso de transmissão no receptor que resulta em todos os primeiros oito pacotes chegarem a tempo para a transmissão.

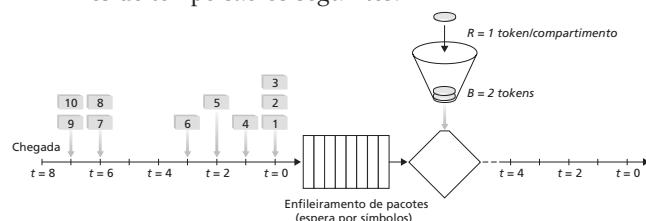
- 11.** Considere novamente a figura em P10, que mostra os tempos de transmissão dos pacotes de áudio e os tempos de recepção
- a.** Calcule o atraso estimado para os pacotes de 2 a 8, usando a fórmula para d_i , da Seção 7.3.2. Use o valor de $u = 0,1$.
- b.** Calcule o desvio estimado do atraso para a média estimada para os pacotes 2 a 8, usando a fórmula para v_i da Seção 7.3.2. Use o valor $u = 0,1$.
- 12.** Recorde os dois esquemas FEC para telefone por Internet descritos na Seção 7.3. Suponha que o primeiro esquema gere uma porção redundante para cada quatro porções originais. Suponha que o segundo esquema use uma codificação de baixa taxa de bits, cuja taxa de transmissão seja 25 por cento da taxa de transmissão da corrente nominal.

- a.** Quanta largura de banda adicional cada esquema requer? E quanto atraso de reprodução cada esquema adiciona?
- b.** Como os dois esquemas funcionarão se em cada grupo de cinco pacotes o primeiro for perdido? Qual esquema terá melhor qualidade de áudio?
- c.** Como os dois esquemas funcionarão se em cada grupo de dois pacotes o primeiro for perdido? Qual esquema terá melhor qualidade de áudio?
- 13.** Dado que uma CDN não aumenta a quantia de capacidade de enlace em uma rede (supondo que a CDN use os enlaces existentes para distribuir seu conteúdo entre os nós CDN), como um CDN aprimora seu desempenho visto pelos hospedeiros? Dê um exemplo.
- 14.** É possível para uma CDN fornecer um desempenho pior a um hospedeiro requisitando um objeto multimídia do que um hospedeiro requisitando um objeto de um servidor de origem? Explique.
- 15.** Como é calculada a variação do atraso entre chegadas no relatório de recepção RTCP? (Dica: leia o RFC do RTP.)
- 16.**
 - a.** Suponha que enviamos dois datagramas IP para a Internet, cada um portando um segmento UDP diferente. O primeiro datagrama tem endereço IP de fonte A1, endereço IP de destino B, porta de fonte P1 e porta de destino T. O segundo datagrama tem endereço IP de fonte A2, endereço IP de destino B, porta de fonte P2 e porta de destino T. Suponha que A1 é diferente de A2 e P1 é diferente de P2. Admitindo que ambos os datagramas cheguem a seu destino final, os dois datagramas UDP serão recebidos pelo mesmo socket? Justifique sua resposta.
 - b.** Suponha que Alice, Bob e Claire querem fazer uma audioconferência usando SIP e RTP. Para Alice enviar e receber pacotes RTP de e para Bob e Claire, somente uma porta UDP é suficiente (além da porta necessária para as mensagens SIP)? Caso a resposta seja positiva, então como o cliente SIP de Alice distingue entre os pacotes RTP recebidos de Bob e Claire?
- 17.** Considere uma sessão RTP com quatro usuários, todos eles enviando e recebendo pacotes RTP para o mesmo endereço *multicast*. Cada usuário envia vídeo a 100 kbps.
 - a.** O RTCP limitará seu tráfego a qual taxa?
 - b.** Quanta largura de banda RTCP será alocada a um determinado receptor?
 - c.** Quanta largura de banda RTCP será alocada a um determinado remetente?
- 18.**
 - a.** Quais as semelhanças entre o RTSP e o HTTP? O RTSP tem métodos? O HTTP pode ser usado para requisitar uma corrente de dados?
 - b.** Quais as diferenças entre RTSP e HTTP? Por exemplo, o HTTP é dentro da banda ou fora da banda? O RTSP mantém informação de estado sobre o cliente (considere a função pausa/reinício)?
- 19.** Verdadeiro ou falso:
 - a.** Se um vídeo armazenado é entregue diretamente de um servidor Web a um transdutor, então a aplicação está usando TCP como protocolo de transporte subjacente.
 - b.** Ao usar RTP, é possível que um remetente mude a codificação no meio de uma sessão.
 - c.** Todas as aplicações que usam RTP devem usar a porta 87.
 - d.** Suponha que uma sessão RTP tenha uma corrente separada de áudio e vídeo para cada remetente. Então, as correntes de áudio e vídeo usam o mesmo SSRC.
 - e.** Em serviços diferenciados, ainda que o comportamento por salto defina diferenças de desempenho entre classes, ele não impõe nenhum mecanismo particular para alcançar esses desempenhos.
 - f.** Suponha que Alice quer estabelecer uma sessão SIP com Bob. Ela inclui, na sua mensagem INVITE, a linha *m = audio 48753 RTP/AVP 3* (AVP 3 denota áudio GSM). Portanto, Alice indicou em sua mensagem que ela deseja enviar áudio GSM.
 - g.** Com referência à declaração anterior, Alice indicou em sua mensagem INVITE que enviará áudio para a porta 48753.
 - h.** Mensagens SIP são enviadas tipicamente entre entidades SIP usando um número default para a porta SIP.
 - i.** Para manter seu registro, clientes SIP têm de enviar mensagens REGISTER periodicamente.
 - j.** SIP impõe que todos os clientes SIP suportem codificação de áudio G.711.
- 20.** Suponha que a política de escalonamento WFQ seja aplicada a um buffer que suporta três classes; suponha que os pesos para essas três classes sejam 0,5, 0,25 e 0,25.
 - a.** Suponha que cada classe tenha um grande número de pacotes no buffer. Em que sequência poderiam ser atendidas essas três classes para atingir os pesos WFQ descritos? (Para escalonamento por varredura cíclica, uma sequência natural é 123123123...)

- b.** Suponha que as classes 1 e 2 tenham um grande número de pacotes no buffer e que não haja pacotes de classe 3 no buffer. Em que sequência as três classes poderiam ser atendidas para alcançar os pesos WFQ descritos?
- 21.** Considere a figura à seguir, similar às Figuras 7.22 à 7.25. Responda as seguintes perguntas:



- a.** Supondo que o serviço é FIFO, indique o tempo em que os pacotes de 2 a 12 deixam a fila. Para cada pacote, qual o atraso entre a chegada e o início do compartimento no qual é transmitido? Qual o atraso médio sobre todos os 12 pacotes?
- b.** Suponha agora um serviço com prioridades, e admita que os números ímpares são de prioridade de alta, e os pares de prioridade baixa. Indique o tempo em que cada pacote de 2 a 12 deixará a fila. Para cada pacote, qual o atraso entre a chegada e o início do compartimento no qual é transmitido? Qual o atraso médio sobre todos os 12 pacotes?
- c.** Suponha agora um serviço em sequência, e admita que os pacotes 1, 2, 3, 6, 11 e 12 sejam de classe 1, e os pacotes 4, 5, 7, 8, 9 e 10 de classe 2. Indique o tempo em que cada pacote de 2 a 12 deixará a fila. Para cada pacote, qual o atraso entre a chegada e a partida? Qual o atraso médio para todos os 12 pacotes?
- d.** Suponha agora a disciplina de serviço de enfileiramento justo ponderado (WFQ), e admita que os números ímpares são de prioridade alta, e os pares de prioridade baixa. A classe 1 tem um peso WFQ de 2, enquanto a classe 2 tem um peso WFQ de 1. Observe que pode não ser possível alcançar uma sincronização WFQ idealizada como vimos no texto, então indique por que você escolheu o pacote específico para ir à serviço em cada tempo de entrada. Para cada pacote, qual o atraso entre a chegada e a partida? Qual o atraso médio para todos os 12 pacotes?
- e.** O que você pode observar sobre o tempo de atraso médio nos quatro casos (FIFO, RR, prioridade e WFQ)?
- 22.** Considere novamente a figura de P21.
- a.** Suponha um serviço prioritário, com os pacotes 1, 4, 5, 6 e 11 sendo de prioridade alta. Os pa-
- tes restantes são de prioridade baixa. Indique os compartimentos nos quais cada pacote de 2 a 12 deixará a fila.
- b.** Agora suponha que um serviço em sequência é usado, com os pacotes 1, 4, 5, 6 e 11 pertencentes a uma classe de tráfego, e os restantes pertencendo a uma segunda classe de tráfego. Indique os compartimentos nos quais cada pacote de 2 a 12 deixará a fila.
- c.** Suponha agora um serviço WFQ, com os pacotes 1, 4, 5, 6 e 11 pertencentes a uma classe de tráfego, e os restantes pertencendo a uma segunda classe de tráfego. A classe 1 tem um peso WFQ de 1, enquanto a classe 2 tem um peso WFQ de 2 (observe que estes pesos são diferentes do que na questão anterior). Indique os compartimentos nos quais cada pacote de 2 a 12 deixará o enfileiramento. Veja também a advertência na questão acima relativa ao serviço WFQ.
- 23.** Considere a figura abaixo, que mostra um leaky bucket sendo alimentado por um fluxo contínuo de pacotes. O buffer de tokens pode segurar no máximo dois tokens e está inicialmente cheio em $t = 0$. Novos tokens chegam em uma velocidade de um símbolo por compartimento. A velocidade do enlace de saída é tal que se dois pacotes obtém tokens no ínicio de um tempo de compartimento, ambos podem ir ao enlace de saída no mesmo compartimento. Os detalhes de tempo são os seguintes:



- Os pacotes chegam, no começo do compartimento. Sendo assim, na figura, os pacotes 1, 2 e 3 chegam no compartimento 0. Se já existirem pacotes na fila, os pacotes que vão chegar se juntam no final da fila. Os pacotes prosseguem em direção ao começo da fila de modo FIFO.
- Depois das chegadas serem adicionadas à fila, se já existirem pacotes na fila, um ou dois desses pacotes (dependendo do número de tokens disponíveis) irão remover um token do buffer de tokens e irão ao enlace de saída durante esse compartimento. Sendo assim, os pacotes 1 e 2 removem um token do buffer (já que existem inicialmente 2 tokens) e vão ao enlace de saída durante o comportamento 0.
- Um novo token é adicionado ao buffer de tokens se este não estiver cheio, já que a velocidade de

- criação de tokens é de $r = 1$ token/compartimento.
4. Tempo avança à próxima entrada, e as etapas se repetem.
- Responda às seguintes perguntas:
- a. Para cada compartimento de tempo, identifique os pacotes que estão na fila e o número de tokens no bucket, imediatamente depois que as chegadas foram processadas (etapa 1 acima) mas antes que os pacotes passem a fila e removam um token. Sendo assim, para o compartimento $t = 0$ no exemplo acima, os pacotes 1, 2 e 3 estão na fila, e existem dois tokens no buffer.
 - b. Para cada compartimento de tempo, indique qual pacote aparecerá na saída depois que os tokens forem removidos da fila. Sendo assim, para o compartimento $t = 0$ no exemplo acima, os pacotes 1 e 2 aparecem no enlace de saída do leaky buffer durante a entrada 0.
24. Repita o problema 23, mas suponha que $r = 2$. Admita novamente que o bucket está cheio inicialmente.
25. Considere o problema 24, mas suponha agora que $r = 3$, e que $b = 2$ como antes. Sua resposta para a questão acima é diferente?
26. Considere o sistema de leaky bucket (discutido na Seção 7.7), que regula a taxa média e o tamanho da rajada de um fluxo de pacotes. Queremos agora também regular a taxa de pico p . Mostre como a saída desse regulador leaky bucket pode ser alimentada para um segundo regulador de leaky bucket, de modo que os dois, em série, regulem a taxa média, a taxa de pico e o tamanho da rajada. Não esqueça de atribuir um tamanho e uma taxa de geração de permissão ao balde do segundo regulador.
27. Diz-se que um fluxo de pacote está de acordo com a especificação de um leaky bucket (r,b) com tamanho de rajada b e taxa média r se o número de pacotes que chega ao leaky bucket for menor do que $rt + b$ pacotes a cada intervalo de tempo de duração t para todo t . Um fluxo de pacotes que esteja de acordo com a especificação do leaky bucket (r,b) alguma vez terá de esperar em um regulador leaky bucket com parâmetros r e b ? Justifique sua resposta.
28. Demonstre que, enquanto $r_1 < R w_1 / (\sum w_j)$, então d_{\max} é, na verdade, o atraso máximo que qualquer pacote do fluxo 1 sofrerá na fila WFQ.



Questões dissertativas

1. Encontre uma empresa que distribua fluxo contínuo de vídeo ao vivo usando distribuição P2P. Escreva um artigo sobre a fundamentação da tecnologia.
2. Na sua opinião, é melhor transmitir fluxo contínuo de áudio/vídeo por TCP ou por UDP?
3. Escreva um relatório sobre os produtos SIP da Cisco.
4. O problema de fornecimento de garantias de QoS pode ser resolvido simplesmente fornecendo largura de banda suficiente, isto é, apenas aumentando todas as capacidades de enlace, de modo que as limitações de largura de banda deixem de ser uma preocupação?
5. Um interessante mercado emergente está usando o telefone por Internet e a LAN de alta velocidade de uma empresa para substituir seu PABX (*private branch exchange* — central privada de comutação telefônica). Faça um relatório de uma página sobre esse assunto que inclua as seguintes questões:
6. Pense em como uma rede de rodovias é dimensionada (por exemplo, como o número de pistas em rodovias conduzida a uma cidade grande é determinado). Faça uma lista com as quatro etapas que você acha que um engenheiro de transportes levaria em consideração quando dimensionando tal rodovia. Quais são as etapas análogas ao dimensionar uma rede de computadores?



Wireshark Lab

Neste laboratório você implementará um servidor e um cliente de vídeo de fluxo contínuo. O cliente usará o protocolo de fluxo contínuo em tempo real (RTSP)

para controlar as ações do servidor. O servidor usará o protocolo de tempo real (RTP) para empacotar o vídeo para transporte por UDP.



Você receberá um código Java que implementará parcialmente RTSP e RTP no cliente e no servidor. Sua tarefa será concluir o código para o cliente e também para o servidor. Quando terminar, você terá criado uma aplicação cliente-servidor que faz o seguinte:

- O cliente envia comandos RTSP SETUP, PLAY, PAUSE e TEARDOWN e o servidor responde aos comandos.
- Quando o servidor estiver no estado de reprodução, ele pega periodicamente um quadro JPEG armazenado, empacota o quadro com RTP e envia o pacote RTP para um socket UDP.
- O cliente recebe os pacotes RTP, extrai os quadros JPEG, descomprime os quadros e os apresenta no seu monitor.

O código que você receberá implementa o protocolo RTSP no servidor e o desempacotamento RTP no cliente e também cuida da apresentação do vídeo transmitido. Você precisará implementar RTSP no cliente e RTP no servidor.

Esta tarefa de programação aprimorará significativamente a compreensão de RTP, RTSP e vídeo em fluxo contínuo para o aluno. Recomendamos veementemente que ela seja executada. A tarefa também sugere vários exercícios opcionais, incluindo implementação do comando RTSP DESCRIBE no cliente e também no servidor. Você encontrará informações completas sobre a tarefa, bem como snippets importantes de código Java, no Companion Website deste livro, www.aw.com/kurose_br.



Entrevista

Henning Schulzrinne

Henning Schulzrinne é professor, diretor do Departamento de Ciências da Computação e chefe do Internet Real-Time Laboratory da Universidade Columbia. É coautor dos protocolos RTSP, RTP e SIP, protocolos fundamentais para comunicação de áudio e vídeo pela Internet. É graduado em engenharia elétrica e industrial pela Universidade de Darmstadt, na Alemanha, mestre em engenharia elétrica e de computação pela Universidade de Cincinnati e doutor em engenharia elétrica pela Universidade de Massachusetts, em Amherst.



O que o fez se decidir pela especialização em tecnologia de rede multimídia?

Aconteceu quase por acidente. Quando fazia doutorado, acabei me envolvendo com a DARTnet, uma rede experimental que abrangia os Estados Unidos com linhas T1. A DARTnet era usada como campo de prova para ferramentas *multicast* e de Internet em tempo real. Isso me levou a implementar minha primeira ferramenta de áudio, a NeVoT. Por intermédio de alguns participantes da DARTnet, acabei me envolvendo com a IETF, com o grupo de trabalho Audio Video Transport, que se formara havia pouco tempo naquela época. O grupo terminou por padronizar o RTP.

Qual foi o seu primeiro emprego no setor de computação? O que implicava?

No meu primeiro emprego no setor de computação fiz a soldagem das peças de um computador Altair. Eu ainda era estudante do ensino médio em Livermore, na Califórnia. Quando voltei à Alemanha, montei uma pequena empresa de consultoria que projetou um programa de gerenciamento de endereços para uma agência de viagens — armazenando dados em fitas cassete para nosso TRS-80 e usando uma máquina de escrever elétrica Selectric da IBM com uma interface de hardware feita em casa servindo de impressora.

Meu primeiro emprego verdadeiro foi no AT&T Bell Laboratories, no desenvolvimento de um emulador de rede para a construção de redes experimentais em ambiente de laboratório.

Quais os objetivos do Internet Real-Time Laboratory?

Nosso objetivo é fornecer componentes e construir blocos para a Internet como uma única infraestrutura de comunicação. Isso inclui o desenvolvimento de novos protocolos, como o GIST (para sinalização de camadas de rede) e o LoST (para encontrar recursos por locais), ou aprimorar os protocolos nos quais já trabalhamos anteriormente, como o SIP, através de trabalhos de presença aprimorada, sistemas peer-to-peer, chamadas de emergência de próxima geração e ferramentas de criação de serviços. Recentemente, também voltamos nossa atenção aos sistemas sem fio VoIP, como as rede 802.11b e 802.11n e talvez as redes WiMax se tornem importantes tecnologias de última milha para telefonia. Também estamos tentando aprimorar a capacidade de os usuários diagnosticarem falhas nos confusos equipamentos e provedores, usando um sistema de diagnóstico de falhas peer-to-peer chamado DYSWIS (Do You See what I See [Você vê o que eu vejo]). Tentamos fazer trabalhos relevantes, ao construir protótipos e sistemas de fonte aberta, ao medir o desempenho de sistemas reais, e contribuindo para os padrões da IETF.

Em sua opinião, qual é o futuro da rede multimídia?

Estamos agora em uma fase de transição; poucos anos nos separam do IP como plataforma universal para serviços de multimídia, do IPTV ao VoIP. Esperamos que o rádio, o telefone e a TV continuem funcionando mesmo durante tempestades de neve e terremotos; portanto, quando a Internet assumir os papéis dessas redes dedicadas, os usuários poderão esperar o mesmo nível de confiabilidade.

Teremos que aprender a projetar tecnologias de rede em um ecossistema de carreiras concorrentes, provedores de serviço e conteúdo, servindo muitos usuários sem treinamento e defendendo-os de um pequeno, porém des-trutivo, punhado de usuários maliciosos e criminoso. A mudança de protocolos está se tornando difícil. Estão se tornando também mais complexas, já que precisam levar em conta os lucros competitivos de negócios, segurança, privacidade e a falta de transparência das redes, causada por firewalls e tradutores de endereços de rede. Desde que a rede multimídia se tornou a base para todo entretenimento ao consumidor, existirá uma ênfase na administração de grandes redes, a um preço baixo. Os usuários esperarão um uso fácil, como ter o mesmo conteúdo em todos seus dispositivos.

Por que o SIP tem um futuro promissor?

Enquanto prossegue a evolução das redes sem fio existentes hoje para 3G, há a esperança de um único mecanismo de sinalização para multimídia que abranja todos os tipos de rede, desde modem a cabo até redes telefônicas de empresas e redes públicas sem fio. Juntamente com softwares de rádio, isso possibilitará que, no futuro, um dispositivo único, tal como um telefone Bluetooth sem fio, possa ser utilizado em uma rede residencial, em uma rede empresarial via 802.11 e em redes de longa distância por meio de redes 3G. Mesmo antes de existir tal dispositivo único, universal, sem fio, os mecanismos de mobilidade pessoal permitem ocultar as diferenças entre redes. Um identificador torna-se o meio universal de alcançar uma pessoa, em vez de ter de lembrar ou ter de passar por meia dúzia de números de telefones específicos para tecnologias ou localizações.

O SIP também separa o fornecimento de transporte de voz (bit) dos serviços de voz. Agora ficou tecnicamente possível acabar com o monopólio local de telefonia, no qual uma única empresa fornece transporte neutro de bits enquanto outras fornecem ‘o tom de discar’ IP e os serviços clássicos de telefonia, como gateways, transferência de chamadas e identificador de chamadas.

Além da sinalização de multimídia, o SIP oferece um novo serviço que estava faltando na Internet: notificação de eventos. Algo parecido com esse serviço já vem sendo oferecido por soluções improvisadas HTTP e por e-mail, mas nunca foi muito satisfatório. Uma vez que eventos são uma abstração comum para sistemas distribuídos, isso pode simplificar a construção de novos serviços.

Você pode dar algum conselho aos estudantes que estão ingressando no campo das redes?

O trabalho com redes é quase interdisciplinar. Ele tira subsídios da engenharia elétrica, da ciência da computação, da pesquisa operacional e de outras disciplinas. Assim, os pesquisadores de redes têm de estar familiarizados com assuntos que estão fora de suas áreas centrais de interesse. Já que as redes estão se tornando partes tão importantes do nosso dia a dia, estudantes que procuram fazer a diferença no campo deveriam pensar em limitações de novos recursos em rede: tempo e esforço humano, em vez de banda larga e armazenamento.

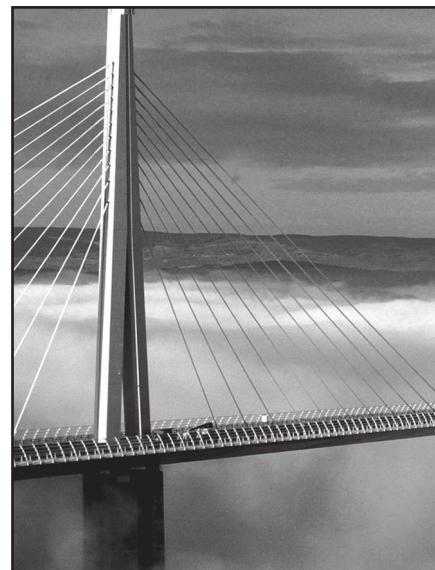
O trabalho com redes pode trazer imensa satisfação, já que se trata de permitir que as pessoas se comuniquem e troquem ideias, um dos valores essenciais do ser humano. A Internet se tornou a terceira maior infraestrutura global, seguindo o sistema de transporte e a distribuição de energia. Poucos setores da economia conseguem trabalhar sem redes de alto desempenho, então deverão existir muitas oportunidades para o futuro próximo.





Capítulo 8

Segurança em redes de computadores



Na Seção 1.6 descrevemos algumas das categorias mais predominantes e prejudiciais dos ataques na Internet, incluindo ataques malware, recusa de serviço, analisador de pacotes, disfarce da fonte e modificação e exclusão de mensagem. Embora tenhamos aprendido muito sobre rede de computadores, ainda não analisamos como protegê-las dos ataques descritos na Seção 1.6. Com nosso conhecimento recém-adquirido em rede de computadores e protocolos da Internet, estudaremos minuciosamente a comunicação segura e, em particular, como as redes de computadores podem ser protegidas desses vilões.

Queremos lhe apresentar Alice e Bob, duas pessoas que desejam se comunicar, porém ‘com segurança’. Como esse texto refere-se a redes, gostaríamos de observar que Alice e Bob podem ser dois roteadores que querem trocar tabelas de roteamento com segurança, um cliente e um servidor que querem estabelecer uma conexão de transporte segura ou duas aplicações de e-mail que querem trocar e-mails com segurança — todos esses tópicos serão examinados mais adiante neste capítulo. Alice e Bob são componentes conhecidos da comunidade de segurança, talvez porque o nome deles seja mais interessante do que uma entidade genérica denominada ‘A’ que quer se comunicar com segurança com uma entidade genérica denominada ‘B’. Amores proibidos, comunicações em tempo de guerra e transações financeiras são as necessidades dos seres humanos comumente citadas quando o assunto é segurança nas comunicações; preferimos a primeira necessidade às duas últimas, e vamos usar, com muito prazer, Alice e Bob como nosso remetente e nosso destinatário e imaginá-los nesse primeiro cenário.

Dissemos que Alice e Bob querem se comunicar, porém ‘com segurança’, mas o que isso significa exatamente? Como veremos, a segurança (assim como o amor) é repleta de maravilhas; isto é, a segurança tem muitas facetas. É certo que Alice e Bob gostariam que o conteúdo de sua comunicação permanecesse secreto, a salvo de um bisbilhoteiro (digamos, um marido, ou esposa, ciumento). Provavelmente gostariam também de ter certeza de que estão realmente se comunicando um com o outro e de que, caso algum bisbilhoteiro interfira na comunicação, essa interferência seja detectada. Na primeira parte deste capítulo, estudaremos as técnicas que permitem criptografar/decriptar comunicações, autenticar a parte com quem estamos nos comunicando e assegurar a integridade da mensagem.

Na segunda parte deste capítulo, examinaremos como os princípios da criptografia podem ser usados para criar protocolos de rede seguros. Utilizando mais uma vez uma abordagem top-down, examinaremos os protocolos seguros em cada uma das (quatro principais) camadas, iniciando pela camada de aplicação. Verificaremos como proteger o e-mail e uma conexão TCP, como prover segurança total na camada de rede, e como proteger uma LAN sem fio. Na terceira parte deste capítulo, avaliaremos a segurança operacional, que tem o objetivo de proteger redes organizacionais de ataques. Verificaremos, de forma minuciosa, como os firewalls e os sistemas de detecção de invasão podem aprimorar a segurança de uma rede organizacional.

8.1 O que é segurança na rede?

Vamos iniciar nosso estudo de segurança de redes voltando aos namorados citados no início, Alice e Bob, que querem se comunicar ‘com segurança’. O que isso significa exatamente? Com certeza, Alice quer que somente Bob entenda a mensagem que ela enviou, mesmo que eles estejam se comunicando por um meio inseguro, em que um intruso (Trudy, a intrusa) pode interceptar qualquer dado que seja transmitido de Alice a Bob. Bob também quer ter certeza de que a mensagem que recebe de Alice foi de fato enviada por ela, e Alice quer ter certeza de que a pessoa com quem está se comunicando é de fato Bob. Alice e Bob também querem ter certeza de que o conteúdo de suas mensagens não foi alterado em trânsito. Também querem, antes de mais nada, ter certeza de que podem se comunicar (isto é, de que ninguém lhes negue acesso aos recursos necessários para a comunicação). Dadas essas considerações, podemos identificar as seguintes propriedades desejáveis da **comunicação segura**:

Confidencialidade. Somente o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida. O fato de abelhos poderem interceptar a mensagem exige, necessariamente, que esta seja **cifrada** de alguma maneira para impedir que uma mensagem interceptada seja entendida por um interceptador. Esse aspecto de confidencialidade é, provavelmente, o significado mais comumente percebido na expressão *comunicação segura*. Estudaremos técnicas de criptografia para cifrar e decifrar dados na Seção 8.2.

Autenticação do ponto final. O remetente e o destinatário precisam confirmar a identidade da outra parte envolvida na comunicação — confirmar que a outra parte realmente é quem alega ser. A comunicação pessoal entre seres humanos resolve facilmente esse problema por reconhecimento visual. Quando entidades comunicantes trocam mensagens por um meio pelo qual não podem ver a outra parte, a autenticação não é assim tão simples. Por que, por exemplo, você deveria acreditar que o e-mail que recebeu e que contém uma sentença afirmando que aquele e-mail veio de um amigo seu realmente veio daquele amigo?

Integridade de mensagem. Mesmo que o remetente e o destinatário consigam se autenticar reciprocamente, eles também querem assegurar que o conteúdo de sua comunicação não seja alterado, por acidente ou por má intenção, durante a transmissão. Extensões das técnicas de soma de verificação que encontramos em protocolos de transporte e de enlace confiáveis podem ser utilizadas para proporcionar integridade à mensagem. Estudaremos autenticação do ponto de chegada e integridade da mensagem na Seção 8.3.

Segurança operacional. Hoje quase todas as organizações (empresas, universidades etc.) possuem redes conectadas à Internet pública. Essas redes podem ser comprometidas potencialmente por atacantes que ganham acesso a essas redes por meio da Internet pública. Os atacantes podem tentar colocar worms nos hospedeiros na rede, adquirir segredos corporativos, mapear as configurações da rede interna e lançar ataques DoS. Veremos na Seção 8.8 que os mecanismos operacionais, como firewalls e sistemas de detecção de invasão, são usados para deter ataques contra redes de organizações. Um firewall localiza-se entre a rede da organização e a rede pública, controlando os acessos de pacote para e da rede. Um sistema de detecção de invasão realiza uma “profunda inspeção de pacote”, alertando os administradores da rede sobre alguma atividade suspeita.

Agora que já determinamos o que significa segurança na rede, vamos considerar em seguida quais são, exatamente, as informações às quais um intruso pode ter acesso e que ações podem ser executadas por ele. A Figura 8.1 ilustra o cenário. Alice, a remetente, quer enviar dados a Bob, o destinatário. Para trocar dados com segurança, além de atender aos requisitos de confidencialidade, autenticação e integridade de mensagens, Alice e Bob trocarão mensagens de controle e de dados (algo muito semelhante ao modo como remetentes e destinatários TCP trocam segmentos de controle e segmentos de dados). Todas ou algumas dessas mensagens normalmente serão criptografadas. Um intruso passivo pode, potencialmente, fazer o seguinte:

monitorar — identificar e gravar as mensagens de controle e de dados no canal;

modificar, inserir ou eliminar mensagens ou conteúdo de mensagens.

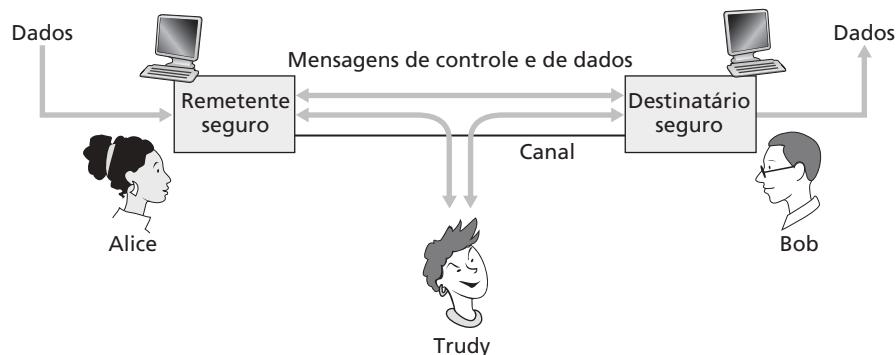


Figura 8.1 Remetente, destinatário e intruso (Alice, Bob e Trudy)

Como veremos, a menos que sejam tomadas contramedidas adequadas, essas capacidades permitem que um intruso monte uma grande variedade de ataques à segurança: monitorar comunicações (e, possivelmente, roubar senhas e dados), fazer-se passar por outra entidade, sequestrar uma sessão em curso, recusar serviço a usuários legítimos da rede sobrecarregando os recursos do sistema e assim por diante. O CERT Coordination Center [CERT, 2009] mantém um resumo de ataques comunicados. Veja também [Cisco Security, 2009; Voydock, 1983; Bhimani, 1996; Skoudis, 2006].

Agora que já temos certeza de que há ameaças reais à solta na Internet, quais são os equivalentes de Alice e Bob na Internet, esses nossos amigos que precisam se comunicar com segurança? Certamente, Bob e Alice podem ser dois usuários humanos em dois sistemas finais, por exemplo, uma Alice real e um Bob real que de fato querem trocar e-mails seguros. Eles podem também ser os participantes de uma transação de comércio eletrônico. Por exemplo, um Bob real pode querer transmitir com segurança o número de seu cartão de crédito a um servidor Web para comprar um produto pela rede. As partes que necessitam de uma comunicação segura podem fazer parte de uma infraestrutura da rede Lembre-se de que o sistema de nomes de domínio (DNS, veja a Seção 2.5) ou daemons roteadores que trocam informações de roteamento (veja a Seção 4.6) requerem comunicação segura entre dois participantes. O mesmo é válido para aplicações de gerenciamento de rede, um tópico que examinaremos no Capítulo 9. Um intruso que conseguisse interferir em consultas do DNS (como discutido na Seção 2.5) processamentos de roteamento [Murphy, 2003] ou funções de gerenciamento de rede [RFC 2574] poderia causar uma devastação na Internet.

Estabelecida a estrutura, apresentadas algumas das definições mais importantes e justificada a necessidade de segurança na rede, vamos examinar a criptografia. Embora a utilização da criptografia para prover confidencialidade seja evidente por si só, veremos em breve que ela também é essencial para prover autenticação, integridade de mensagem, não repudiação e controle de acesso — o que faz da criptografia uma pedra fundamental da segurança na rede.

8.2 Princípios de criptografia

Embora a criptografia tenha uma longa história que remonta, no mínimo, a Júlio César, técnicas modernas de criptografia, incluindo muitas das usadas na Internet, são baseadas em progressos feitos nos últimos 30 anos. O livro de Kahn, *The codebreakers* [Kahn, 1967], e o livro de Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh, 1999], nos oferecem um panorama fascinante dessa longa história. Uma discussão completa sobre a criptografia exige um livro inteiro [Kaufman, 1995; Schneier, 1995]; portanto, trataremos apenas de seus aspectos essenciais, em particular do modo como as técnicas criptográficas são postas em prática na Internet. Observamos também que, conquanto nessa seção focalizaremos a utilização da criptografia aplicada à confidencialidade, em breve veremos que as técnicas criptográficas estão inextricavelmente entrelaçadas com a autenticação, a integridade de mensagens, a não repudiação e outras.

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário, é claro, deve estar habilitado a recuperar os dados originais a partir dos dados disfarçados. A Figura 8.2 apresenta alguns dos componentes mais importantes da terminologia usada em criptografia.

Suponha agora que Alice queira enviar uma mensagem a Bob. A mensagem de Alice em sua forma original (por exemplo, "Bob, I love you. Alice") é conhecida como **texto aberto** ou **texto claro**. Alice criptografa sua mensagem em texto aberto usando um **algoritmo de criptografia**, de modo que a mensagem criptografada, conhecida como **texto cifrado**, pareça ininteligível para qualquer intruso. O interessante é que em muitos sistemas criptográficos modernos, incluindo os usados na Internet, a técnica de codificação é *conhecida* — publicada, padronizada e disponível para qualquer um (por exemplo, [RFC 1321, RFC 2437, RFC 2420; NIST 2001]), mesmo para um intruso em potencial! Evidentemente, se todos conhecem o método para codificar dados, então deve haver alguma informação secreta que impede que um intruso decifre os dados transmitidos. É aqui que entra a chave.

Na Figura 8.2, Alice fornece uma **chave**, K_A , uma cadeia de números ou de caracteres, como entrada para o algoritmo de criptografia. O algoritmo de criptografia pega essa chave e o texto aberto da mensagem, m , como entrada e produz texto cifrado como saída. A notação $K_A(m)$ refere-se à forma do texto cifrado (criptografado usando a chave K_A) da mensagem em texto aberto, m . O algoritmo criptográfico propriamente dito, que usa a chave K_A , ficará evidente do próprio contexto. De maneira semelhante, Bob fornecerá uma chave, K_B , ao **algoritmo de decriptação**, que pega o texto cifrado e a chave de Bob como entrada e produz o texto aberto original como saída. Isto é, se Bob receber uma mensagem criptografada $K_A(m)$, ele a decriptará calculando $K_B(K_A(m)) = m$. Em **sistemas de chaves simétricas**, as chaves de Bob e de Alice são idênticas e secretas. Em **sistemas de chaves públicas**, é usado um par de chaves. Uma das chaves é conhecida por Bob e por Alice (na verdade, é conhecida pelo mundo inteiro). A outra chave é conhecida apenas por Bob ou por Alice (mas não por ambos). Nas duas subseções seguintes, examinaremos com mais detalhes sistemas de chaves simétricas e de chaves públicas.

8.2.1 Criptografia de chaves simétricas

Todos os algoritmos criptográficos envolvem a substituição de um dado por outro, como tomar um trecho de um texto aberto e então, calculando e substituindo esse texto por outro cifrado apropriado, criar uma mensagem cifrada. Antes de estudar um sistema criptográfico moderno baseado em chaves, vamos abordar um algoritmo de chaves simétricas muito antigo, muito simples, atribuído a Júlio César e conhecido como **cifra de César** (uma cifra é um método para criptografar dados).

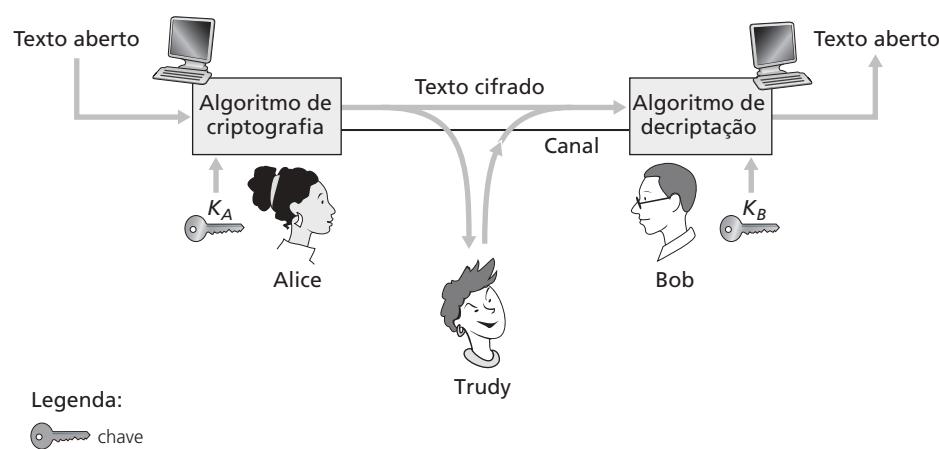


Figura 8.2 Componentes criptográficos

A cifra de César funciona tomando cada letra da mensagem do texto aberto e substituindo-a pela k -ésima letra sucessiva do alfabeto (permitindo a rotatividade do alfabeto, isto é, a letra ‘z’ seria seguida novamente da letra ‘a’). Por exemplo, se $k = 3$, então a letra ‘a’ do texto aberto fica sendo ‘d’ no texto cifrado; ‘b’ no texto aberto se transforma em ‘e’ no texto cifrado, e assim por diante. Nesse caso, o valor de k serve de chave. Por exemplo, a mensagem “bob, i love you. alice.” se torna “ere, l oryh brx. dolfh.” em texto cifrado. Embora o texto cifrado na verdade pareça não ter nexo, você não levaria muito tempo para quebrar o código se soubesse que foi usada a cifra de César, pois há somente 25 valores possíveis para as chaves.

Um aprimoramento da cifra de César é a **cifra monoalfabética**, que também substitui uma letra do alfabeto por outra. Contudo, em vez de substituir as letras seguindo um padrão regular (por exemplo, substituição por um deslocamento de k para todas as letras), qualquer letra pode ser substituída por qualquer outra, contanto que cada letra tenha uma única letra substituta e vice-versa. A regra de substituição apresentada na Figura 8.3 mostra uma regra possível para codificar textos abertos.

A mensagem do texto aberto “bob, I love you. alice.” se torna “nkn, s gktc wky. mgsbc.” Assim, como aconteceu no caso da cifra de César, o texto parece sem nexo. A cifra monoalfabética também parece ser melhor que a cifra de César, pois há $26!$ (da ordem de 10^{26}) possíveis pares de letras, em vez de 25 possíveis!

Uma abordagem de força bruta que experimentasse todos os 10^{26} pares possíveis demandaria um esforço demasiadamente grande e impediria que esse fosse um método viável para quebrar o algoritmo criptográfico e decodificar a mensagem. Contudo, pela análise estatística da linguagem do texto aberto, por exemplo, e sabendo que as letras ‘e’ e ‘t’ são as mais frequentes em textos em inglês (13 por cento e 9 por cento das ocorrências de letras, respectivamente) e sabendo também que determinados grupos de duas e de três letras aparecem com bastante frequência em inglês (por exemplo, ‘in’, ‘it’, ‘the’, ‘ion’, ‘ing’, e assim por diante), torna-se relativamente fácil quebrar esse código. Se o intruso tiver algum conhecimento sobre o possível texto da mensagem, então ficará mais fácil ainda quebrar o código. Por exemplo, se a intrusa Trudy for a esposa de Bob e suspeitar que ele está tendo um caso com Alice, ela poderá facilmente imaginar que os nomes ‘bob’ e ‘alice’ apareçam no texto. Se Trudy tivesse certeza de que esses dois nomes aparecem no texto cifrado e tivesse uma cópia do texto cifrado da mensagem do exemplo, então ela poderia determinar imediatamente sete dos 26 pares de letras, o que resultaria em 10^9 possibilidades a menos para verificar pelo método da força bruta. Na verdade, se Trudy suspeitasse que Bob estava tendo um caso, ela poderia muito bem esperar encontrar algumas outras palavras preferenciais no texto também.

Considerando como seria fácil para Trudy quebrar o código criptográfico de Bob e Alice, podemos distinguir três cenários diferentes, dependendo do tipo de informação que o intruso tem:

Ataque exclusivo a texto cifrado. Em alguns casos, o intruso pode ter acesso somente ao texto cifrado interceptado, sem ter nenhuma informação exata sobre o conteúdo do texto aberto. Já vimos como a análise estatística pode ajudar o **ataque exclusivo ao texto cifrado** em um esquema criptográfico.

Ataque com texto aberto conhecido. Vimos anteriormente que, se Trudy, por alguma razão, tivesse certeza de que ‘bob’ e ‘alice’ apareciam no texto cifrado, ela poderia determinar os pares (texto cifrado, texto aberto) para as letras *a*, *b*, *i*, *c*, *e*, *b* e *o*. Trudy poderia também ser muito sortuda e ter gravado todas as transmissões de texto cifrado e descoberto uma versão decifrada pelo próprio Bob escrita em um pedaço de papel. Quando um intruso conhece alguns dos pares (texto aberto, texto cifrado), referimo-nos a isso como ataque ao esquema criptográfico a partir de **texto aberto conhecido**.

Ataque com texto aberto escolhido. Nesse tipo de ataque, o intruso pode escolher a mensagem em texto aberto e obter seu texto cifrado correspondente. Para os algoritmos criptográficos simples que vimos até aqui, se Trudy conseguisse que Alice enviasse a mensagem “The quick fox jumps over the

Letra no texto aberto:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Letra no texto cifrado:	m n b v c x z a s d f g h j k l p o i u y t r e w q

Figura 8.3 Uma cifra monoalfabética



lazy brown dog", ela poderia decifrar completamente o esquema criptográfico. Veremos em breve que, para técnicas de criptografia mais sofisticadas, um ataque com um texto aberto escolhido não significa necessariamente que a técnica criptográfica possa ser decifrada.

Quinhentos anos atrás foram inventadas técnicas que aprimoravam a cifra monoalfabética, conhecidas como cifras polialfabéticas. A ideia subjacente à **criptografia polialfabética** é usar várias cifras monoalfabéticas com uma cifra monoalfabética específica para codificar uma letra em uma posição específica no texto aberto da mensagem. Assim, a mesma letra, quando aparece em posições diferentes no texto aberto da mensagem, pode ser codificada de maneira diferente. Um exemplo de esquema criptográfico polialfabético é mostrado na Figura 8.4, na qual há duas cifras de César (com $k = 5$ e $k = 19$), que aparecem nas linhas da figura. Podemos optar pelo uso dessas duas cifras de César, C_1 e C_2 , seguindo o modelo de repetição C_1, C_2, C_1, C_2 . Isto é, a primeira letra do texto deve ser cifrada usando-se C_1 , a segunda e a terceira, C_2 , a quarta, C_1 e a quinta, C_2 . O modelo, então, se repete, com a sexta letra sendo cifrada usando-se C_1 ; a sétima, com C_2 , e assim por diante. Dessa maneira, a mensagem em texto aberto "bob, I love you." é cifrada como "ghu, n etox dhz.". Note que o primeiro 'b' da mensagem em texto aberto é cifrado usando-se C_1 , ao passo que o segundo 'b' é cifrado usando-se C_2 . Nesse exemplo, a 'chave' da codificação e da decodificação é o conhecimento das duas cifras de César ($k = 5, k = 19$) e do modelo C_1, C_2, C_1, C_2 .

Cifras de bloco

Vamos agora nos direcionar a tempos modernos e analisar como a criptografia de chave simétrica é feita atualmente. Existem duas classes amplas de técnicas de criptografia simétrica: cifras de fluxo e cifras de bloco. Examinaremos as cifras de fluxo, de forma breve, na Seção 8.7 ao investigarmos a segurança para LANs sem fio. Nesta seção, focaremos em cifras de bloco, que são utilizadas em muitos protocolos seguros da Internet, incluindo PGP (para e-mail seguro), SSL (para conexões TCP seguras) e Ipsec (para proteger o transporte da camada de rede).

Na cifra de bloco, a mensagem a ser criptografada é processada em blocos de k bits. Por exemplo, se $k = 64$, então a mensagem é dividida em blocos de 64 bits, e cada bloco é criptografado de maneira independente. Para criptografar um bloco, a cifra utiliza um mapeamento um para um para mapear o bloco de k bits de *texto aberto* para um bloco de k bits de *texto cifrado*. Vamos examinar um exemplo. Suponha que $k = 3$, de modo que a cifra de bloco mapeie entradas de 3 bits (*texto aberto*) para saídas de 3 bits (*texto cifrado*). Um possível mapeamento é determinado na Tabela 8.1. Observe que esse é um mapeamento um para um; ou seja, há uma saída diferente para cada entrada. Essa cifra de bloco divide a mensagem em blocos de até 3 bits e criptografa cada bloco de acordo com o mapeamento acima. Você deve verificar que a mensagem 010110001111 é criptografada para 101000111001.

Ainda no exemplo do bloco de 3 bits, observe que o mapeamento na Tabela 8.1 é um de muitos possíveis mapeamentos. Quantos possíveis mapeamentos existem? Para responder a essa questão, observe que um mapeamento não é nada mais do que uma permutação de todas as possíveis entradas. Existem $2^3 (= 8)$ possíveis entradas (relacionadas nas colunas de entrada). Essas oito entradas podem ser permutadas em $8! = 40.320$ formas diferentes. Uma vez que cada uma dessas permutações especifique um mapeamento, há 40.320 mapeamentos possíveis. Podemos ver cada um desses mapeamentos como uma chave — se Alice e Bob sabem o mapeamento (a chave), eles podem criptografar e decriptografar as mensagens enviadas entre eles.

O ataque de força bruta para essa cifra é tentar decriptografar o *texto cifrado* usando todos os mapeamentos. Com apenas 40.320 mapeamentos (quando $k = 3$), isso pode ser rapidamente realizado em um computador de mesa. Para impedir os ataques de força bruta, as cifras de bloco normalmente usam blocos muito maiores, consis-

Letra do texto aberto:	a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k = 5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k = 19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s

Figura 8.4 Uma cifra polialfabética que utiliza duas cifras de César

Entrada	Saída	Entrada	Saída
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

Tabela 8.1 Requisitos de aplicações de rede selecionadas

tindo em $k = 64$ bits ou ainda maior. Observe que o número de mapeamentos possíveis para uma cifra de bloco k geral é $2^{k!}$, o qual é extraordinário para até mesmo valores moderados de k (como $k = 64$).

Embora as cifras de bloco da tabela completa, como descritas, com valores moderados de k possam produzir esquemas robustos de criptografia de chave simétrica, eles infelizmente são difíceis de implementar. Para $k = 64$ e para um determinado mapeamento, Alice e Bob precisariam manter uma tabela com 2^{64} valores de entrada, uma tarefa impraticável. Além disso, se Alice e Bob quiserem trocar chaves, cada um teria de renovar a tabela. Assim, a cifra de bloco da tabela completa, que fornecem mapeamentos predeterminados entre todas as entradas e saídas (como no exemplo acima), está simplesmente fora de cogitação.

Em vez disso, as cifras de bloco normalmente utilizam funções que simulam, de maneira aleatória, tabelas permutadas. Um exemplo (adaptado de [Kaufman, 1995]) de tal função para $k = 64$ bits é mostrado na Figura 8.5. A função primeiramente divide um bloco de 64 bits em 8 blocos, com cada bloco consistindo de 8 bits. Cada bloco de 8 bits é processado por uma tabela de 8 bits para 8 bits, a qual possui um tamanho controlável. Por exemplo, o primeiro bloco é processado pela tabela denotada por T_1 . Em seguida, os oito blocos de saída são reunidos em um bloco de 64 bits. As posições dos 64 bits no bloco são, então, permutadas para produzir uma saída de 64 bits. Essa saída é devolvida à entrada de 64 bits, onde se inicia outro ciclo. Após n ciclos, a função apresenta um bloco de 64 bits de texto cifrado. O objetivo de cada ciclo é fazer com que cada bit de entrada afete a maioria (se não todos) dos bits finais de saída. (Se somente um ciclo fosse usado, um determinado bit de entrada afetaria somente 8 dos 64 bits de saída.) A chave para esse algoritmo das cifras de bloco seria as oito tabelas de permutação (admitindo que a função de permutação seja publicamente conhecida).

Hoje, existem diversas cifras de bloco conhecidas, incluindo DES (abreviação de *Data Encryption Standard* [*Padrão de Criptografia de Dados*]), 3DES e AES (abreviação de *Advanced Encryption Standard* [*Padrão de Criptografia Avançada*]). Cada um desses padrões utiliza funções em vez de tabelas predeterminadas, segundo a

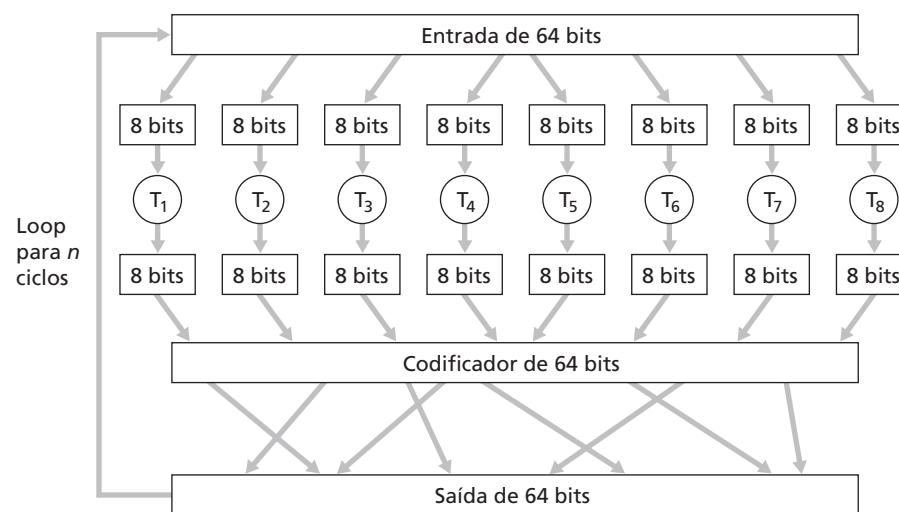
**Figura 8.5** Exemplo de uma cifra de bloco

Figura 8.5 (embora mais complexa e específica para cada cifra). Cada um desses algoritmos também utiliza uma cadeia de bits para chave. Por exemplo, o DES usa blocos de 64 bits com uma chave de 56 bits. O AES usa blocos de 128 bits e pode operar com chaves de 128, 192 e 256 bits de comprimento. Uma chave do algoritmo determina os mapeamentos da “minitabela” e permutações dentro do algoritmo. O ataque de força bruta para cada uma dessas cifras é percorrer todas as chaves, aplicando o algoritmo de decriptografia com cada chave. Observe que com o comprimento de chave n , há 2^n chaves possíveis. NIST [NIST, 2001] estima que uma máquina que pudesse decifrar um DES de 56 bits em um segundo (ou seja, testar 2^{56} chaves em um segundo) levaria, aproximadamente, 149 trilhões de anos para decifrar uma chave AES de 128 bits.

Encadeamento de blocos de cifras

Em aplicações de redes de computadores, normalmente precisamos criptografar mensagens longas (ou fluxos de dados longos). Se aplicarmos uma cifra de bloco, como descrita, simplesmente partindo a mensagem em blocos de k bits e criptografando, independentemente, cada bloco, um problema sutil, mas importante ocorrerá. Para entendê-lo, note que dois ou mais blocos de texto aberto podem ser idênticos. Por exemplo, o texto aberto em dois ou mais blocos poderia ser “HTTP/1.1”. Em relação a esses blocos idênticos, uma cifra de bloco produziria, é claro, o mesmo texto cifrado. Um atacante poderia eventualmente adivinhar o texto aberto ao ver blocos de texto cifrados idênticos e ser capaz de decriptografar a mensagem inteira identificando os blocos de texto cifrado idênticos e usando o que sabe sobre a estrutura do protocolo [Kaufman, 1995].

Para abordar esse problema, podemos associar um pouco de aleatoriedade ao texto cifrado para que blocos de texto aberto idênticos produzam blocos de texto cifrado diferentes. Para explicar essa ideia, $m(i)$ representará o i -ésimo bloco de texto aberto, $c(i)$ representará o i -ésimo bloco de texto cifrado, e $a \oplus b$ representará o ou-exclusivo (XOR) das duas cadeias de bits, a e b . (Lembre-se de que $0 \oplus 0 = 1 \oplus 1 = 0$ e $0 \oplus 1 = 1 \oplus 0 = 1$, e o XOR das duas cadeias de bits é feito em uma base de bit por bit. Então, por exemplo, $10101010 \oplus 11110000 = 01011010$.) Ademais, denote o algoritmo de criptografia da cifra de bloco com a chave S como K_S . Eis a ideia básica. O emissor cria um número aleatório $r(i)$ de k bits para o i -ésimo bloco e calcula $c(i) = K_S(m(i) \oplus r(i))$. Observe que um novo número aleatório de k bits é escolhido para cada bloco. O emissor envia, então, $c(1)$, $r(1)$, $c(2)$, $r(2)$, $c(3)$, $r(3)$, etc. Visto que o receptor recebe $c(i)$ e $r(i)$, ele pode recuperar cada bloco de texto aberto computando $m(i) = K_S(c(i)) + r(i)$. É importante observar que, embora $r(i)$ seja enviado inocentemente e, portanto, pode ser analisada por Trudy, ela não pode obter blocos de texto aberto $m(i)$, uma vez que ela não conhece a chave K_S . Observe também que se dois blocos de texto aberto $m(i)$ e $m(j)$ são iguais, os blocos de texto cifrado correspondentes $c(i)$ e $c(j)$ serão diferentes (contanto que os números $r(i)$ e $r(j)$ aleatórios sejam diferentes, o que acontece com uma alta probabilidade).

Como um exemplo, considere a cifra de bloco de 3 bits na Tabela 8.1. Suponha que o texto aberto seja 010010010. Se Alice criptografa-lo diretamente, sem incluir a aleatoriedade, o texto cifrado resultante se torna 101101101. Se Trudy analisar esse texto cifrado, por cada uma das três cifras de blocos ser igual, ela pode pensar que cada um dos blocos de texto aberto são iguais. Agora suponha que em vez disso Alice cria blocos aleatórios $r(1) = 001$, $r(2) = 111$, e $r(3) = 100$ e use a técnica acima para criar o texto cifrado $c(1) = 100$, $c(2) = 010$ e $c(3) = 000$. Observe que os três blocos de texto cifrado são diferentes mesmo se os blocos de texto aberto são iguais. Alice então envia $c(1)$, $r(1)$, $c(2)$ e $r(2)$. Você deve verificar que Bob pode obter o texto aberto original usando a chave K_S compartilhada.

O leitor perspicaz observará que introduzir a aleatoriedade resolve o problema, mas cria outro: ou seja, Alice deve transmitir duas vezes mais bits que antes. Realmente, para cada bit de cifra, ela deve agora enviar um bit aleatório, dobrando a largura de banda requerida. Para obtermos o melhor dos dois mundos, as cifras de bloco normalmente usam uma técnica chamada Encadeamento do Bloco de Cifra (CBC). A ideia básica é enviar somente *um valor aleatório junto com a primeira mensagem e, então, fazer com que o emissor e o receptor usem blocos codificados em vez do número aleatório subsequente*. O CBC opera, como segue:

1. Antes de criptografar a mensagem (ou o fluxo de dados), o emissor cria uma cadeia de k bits, chamada Vetor de Inicialização (IV). Denote esse vetor de inicialização por $c(0)$. O emissor envia o IV ao receptor em texto aberto.

2. Em relação ao primeiro bloco, o emissor calcula $m(1) + c(0)$, ou seja, calcula o ou-exclusivo do primeiro bloco de texto aberto com o IV. Ele então codifica o resultado através do algoritmo de cifra de bloco para obter o bloco de texto cifrado correspondente; ou seja $c(1) = K_s(m(1) \oplus c(0))$. O emissor envia o bloco criptografado $c(1)$ ao receptor.
3. Para o i -ésimo bloco, o emissor cria o i -ésimo bloco de texto cifrado de $c(i) = K_s(m(i) \oplus c(i-1))$.

Vamos agora examinar algumas das consequências desta abordagem. Primeiro, o receptor ainda será capaz de recuperar a mensagem original. Realmente, quando o receptor recebe $c(i)$, ele o decriptografa com K_s para obter $s(i) = m(i) + c(i-1)$; uma vez que o receptor também conhece $c(i-1)$, ele então obtém o bloco de texto aberto de $m(i) = s(i) + c(i-1)$. Segundo, mesmo se dois blocos de texto aberto forem idênticos, os texto cifrados correspondentes (quase sempre) serão diferentes. Terceiro, embora o emissor envie o IV aberto, um invasor ainda não será capaz de decriptografar os blocos de texto cifrado, visto que o invasor não conhece a chave secreta, S . Por fim, o emissor somente envia um bloco auxiliar (o IV), fazendo aumentar, de forma insignificante, o uso da largura de banda para longas mensagens (consistindo de centenas de blocos).

Como um exemplo, vamos determinar o texto cifrado para uma cifra de bloco de 3 bits na Tabela 8.1 com texto aberto 010010001 e IV = $c(0) = 001$. O emissor primeiro usa o IV para calcular $c(1) = K_s(m(1) + c(0)) = 100$. O emissor calcula, então, $c(2) = K_s(m(2) + c(1)) = K_s(010 + 100) = 000$, e $c(3) = K_s(m(3) + c(2)) = K_s(010 + 000) = 101$. O leitor deve verificar que o receptor, conhecendo o IV e K_s , pode recuperar o texto aberto original.

O CBC possui uma consequência importante para os protocolos de rede seguros: precisaremos fornecer um mecanismo dentro do protocolo para distribuir o IV do emissor ao receptor. Veremos como isso é feito para vários protocolos mais adiante, neste capítulo.

8.2.2 Criptografia de chave pública

Por mais de dois mil anos (desde a época da cifra de César até a década de 1970), a comunicação cifrada exigia que as duas partes comunicantes compartilhassem um segredo em comum — a chave simétrica usada para cifrar e decifrar. Uma dificuldade dessa abordagem é que as duas partes têm de concordar, de alguma maneira, com a chave compartilhada, mas, para fazê-lo, é preciso comunicação (presumivelmente segura)! Talvez as partes pudessem se encontrar antes, escolher a chave pessoalmente (por exemplo, dois dos centuriões de César poderiam se encontrar nos banhos romanos) e, mais tarde, se comunicar de modo cifrado. Em um mundo em rede, contudo, o mais provável é que as partes comunicantes nunca possam se encontrar e jamais possam conversar a não ser pela rede. É possível que elas se comuniquem por criptografia sem compartilhar uma chave comum secreta conhecida com antecedência? Em 1976, Diffie e Hellman [Diffie, 1976] apresentaram um algoritmo (conhecido como Troca de Chaves Diffie Hellman — Diffie Hellman Key Exchange) que faz exatamente isso — uma abordagem da comunicação segura radicalmente diferente e de uma elegância maravilhosa que levou ao desenvolvimento dos atuais sistemas de criptografia de chaves públicas. Veremos em breve que os sistemas de criptografia de chaves públicas também têm diversas propriedades maravilhosas que os tornam úteis não somente para criptografia, mas também para autenticação e assinaturas digitais. É interessante que, recentemente, veio à luz que ideias semelhantes às de [Diffie, 1976] e às da [RSA, 1978] foram desenvolvidas independentemente no início da década de 1970 em uma série de relatórios secretos escritos por pesquisadores do Grupo de Segurança para Comunicação e Eletrônica (Communications-Electronics Security Group) do Reino Unido [Ellis, 1987]. Como acontece com frequência, grandes ideias podem surgir de modo independente em diversos lugares; felizmente, os progressos da criptografia de chaves públicas ocorreram não apenas no âmbito privado, mas também no público.

A utilização de criptografia de chaves públicas, como conceito, é bastante simples. Suponha que Alice queira se comunicar com Bob. Como mostra a Figura 8.6, em vez de Bob e Alice compartilharem uma única chave secreta (como no caso dos sistemas de chaves simétricas), Bob (o destinatário das mensagens de Alice) tem duas chaves — uma **chave pública**, que está à disposição do *mundo todo* (inclusive à disposição de Trudy, a intrusa), e uma **chave privada**, que apenas ele (Bob) conhece. Usaremos a notação K_B^+ e K_B^- para nos referirmos às chaves pública e privada de Bob, respectivamente. Para se comunicar com Bob, Alice busca primeiramente a chave pública de Bob.

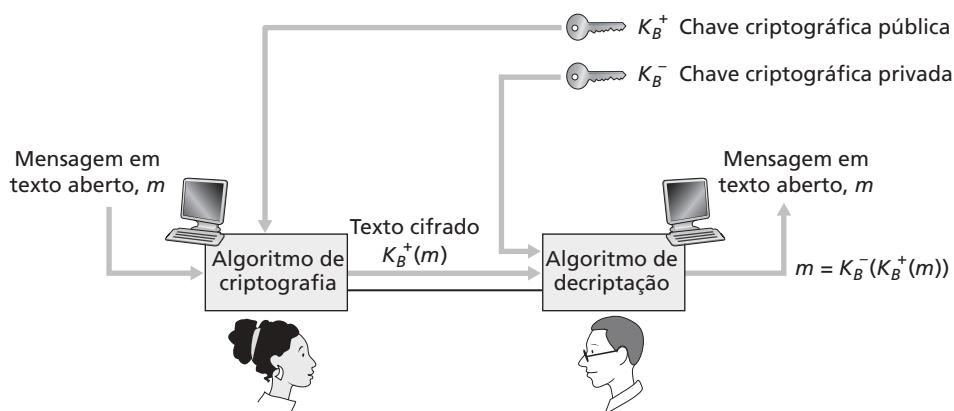


Figura 8.6 Criptografia de chaves públicas

Em seguida, ela criptografa sua mensagem, m , usando a chave pública de Bob e um algoritmo criptográfico conhecido (por exemplo, padronizado), isto é, Alice calcula $K_B^+(m)$. Bob recebe a mensagem criptografada de Alice e usa sua chave privada e um algoritmo de decriptação conhecido (por exemplo, padronizado) para decifrar a mensagem de Alice, isto é, Bob calcula $K_B^-(K_B^+(m))$. Veremos, mais adiante, que há algoritmos e técnicas de criptografia/decriptação para escolher chaves públicas e privadas de modo que $K_B^-(K_B^+(m)) = m$; isto é, aplicando a chave pública de Bob, K_B^+ , à mensagem m , para obter $K_B^+(m)$, e então aplicando a chave privada de Bob, K_B^- , à versão criptografada de m , isto é, calculando $K_B^-(K_B^+(m))$, obtemos m novamente. Esse resultado é notável! Dessa maneira Alice pode utilizar a chave de Bob disponível publicamente para enviar uma mensagem secreta a Bob sem que nenhum deles tenha de permutar nenhuma chave secreta! Veremos em breve que nós podemos permutar a chave pública e a chave privada de criptografia e obter o mesmo resultado notável, isto é, $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$.

A utilização de criptografia de chave pública é, portanto, conceitualmente simples. Mas há duas preocupações que podem nos ocorrer imediatamente. A primeira preocupação é que, embora um intruso que intercepte a mensagem cifrada de Alice veja apenas dados ininteligíveis, ele conhece tanto a chave (a chave pública de Bob, que está disponível a todos) quanto o algoritmo que Alice usou para a criptografia. Assim, Trudy pode montar um ataque com texto aberto escolhido utilizando o algoritmo criptográfico padronizado conhecido e a chave criptográfica de acesso público de Bob para codificar a mensagem que quiser! Ela pode até tentar, por exemplo, codificar mensagens, ou partes delas, que suspeita que Alice poderia enviar. Fica claro que, para a criptografia de chave pública funcionar, a escolha de chaves e de códigos de criptografia/decriptação deve ser feita de tal maneira que seja impossível (ou, ao menos, tão difícil que se torne quase impossível) para um intruso determinar a chave privada de Bob ou conseguir decifrar ou adivinhar a mensagem de Alice a Bob. A segunda preocupação é que, como a chave criptográfica de Bob é pública, qualquer um pode enviar uma mensagem cifrada a Bob, incluindo Alice ou alguém *que se passa* por Alice. No caso de uma única chave secreta compartilhada, o fato de o remetente conhecer a chave secreta identifica implicitamente o remetente para o destinatário. No caso da criptografia de chave pública, contudo, isso não acontece, já que qualquer um pode enviar uma mensagem cifrada a Bob usando a chave dele, que está publicamente disponível a todos. É preciso uma assinatura digital, um tópico que estudaremos na Seção 8.3, para vincular um remetente a uma mensagem.

RSA

Embora existam muitos algoritmos e chaves que tratam dessas preocupações, o **algoritmo RSA** (cujo nome se deve a seus inventores, Ron Rivest, Adi Shamir e Leonard Adleman) tornou-se quase um sinônimo de criptografia de chave pública. Em primeiro lugar, vamos ver como o RSA funciona e, depois, examinar por que ele funciona.

O RSA faz uso extensivo das operações aritméticas usando a aritmética de módulo- n . Vamos revisar de maneira breve a aritmética modular. Lembre que $x \bmod n$ simplesmente significa o resto de x quando dividido por

n , então, por exemplo, $19 \bmod 5 = 4$. Na aritmética modular, uma pessoa executa as operações comuns de adição, multiplicação e exponenciação. Entretanto, o resultado de cada operação é substituído pelo resto inteiro que sobra quando o resultado é dividido por n . A adição e a multiplicação com a aritmética modular são facilitadas com as seguintes propriedades úteis:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

Segue da terceira propriedade que $(a \bmod n)^d \bmod n = a^d \bmod n$, uma identidade que em breve acharemos muito útil.

Agora suponha que Alice queira enviar a Bob uma mensagem criptografada por meio do RSA, conforme ilustrado na Figura 8.6. Em nossa discussão sobre o RSA, vamos sempre manter em mente que uma mensagem não é nada mais do que um padrão de bit, e cada padrão de bit pode ser representado unicamente por um número inteiro (junto com o comprimento do padrão de bits). Por exemplo, suponha que uma mensagem tenha o padrão de bit 1001; essa mensagem pode ser representada pelo número inteiro decimal 9. Assim, criptografar uma mensagem com RSA é equivalente a criptografar um número inteiro que representa a mensagem.

Existem dois componentes inter-relacionados do RSA:

A escolha da chave pública e da chave privada.

O algoritmo de criptografia/decriptação.

Para escolher as chaves pública e privada, Bob deve executar as seguintes etapas:

1. Escolher dois números primos grandes, p e q . Que ordem de grandeza devem ter p e q ? Quanto maiores os valores, mais difícil será quebrar o RSA, mas mais tempo se levará para realizar a codificação e a decodificação. O RSA Laboratories recomenda que o produto de p e q seja da ordem de 1.024 bits. Para uma discussão sobre como achar números primos grandes, consulte [Caldwell, 2007].
2. Computar $n = pq$ e $z = (p - 1)(q - 1)$.
3. Escolher um número e menor do que n que não tenha fatores comuns (exceto o 1) com z . (Nesse caso, dizemos que e e z são números primos entre si.) A letra ‘e’ é usada já que esse valor será utilizado na criptografia (‘encryption’, em inglês).
4. Achar um número d , tal que $ed - 1$ seja divisível exatamente (isto é, não haja resto na divisão) por z . A letra ‘d’ é usada porque seu valor será utilizado na decriptação. Em outras palavras, dado e , escolhemos d tal que

$$ed \bmod z = 1$$

5. A chave pública que Bob põe à disposição de todos, K_B^+ , é o par de números (n, e) ; sua chave privada, K_B^- é o par de números (n, d) .

A criptografia feita por Alice e a decriptação feita por Bob acontecem como segue:

Suponha que Alice queira enviar a Bob um padrão de bits, ou número m , tal que $m < n$. Para codificar, Alice calcula a potência m^e e, então, determina o resto inteiro da divisão de m^e por n . Assim, o valor cifrado, c , da mensagem em texto aberto de Alice, m , é:

$$c = m^e \bmod n.$$

O padrão de bit correspondente a esse texto cifrado c é enviado a Bob.

Para decifrar a mensagem em texto cifrado recebida, c , Bob processa

$$m = c^d \bmod n,$$

que exige o uso de sua chave secreta (n, d) .

Como exemplo simples de RSA, suponha que Bob escolha $p = 5$ e $q = 7$. (Admitimos que esses valores são muito pequenos para ser seguros.) Então, $n = 35$ e $z = 24$. Bob escolhe $e = 5$, já que 5 e 24 não têm fatores comuns. Por fim, ele escolhe $d = 29$, já que $5 \cdot 29 - 1$ (isto é, $ed - 1$) é divisível exatamente por 24. Ele divulga os dois valores, $n = 35$ e $e = 5$, e mantém em segredo o valor $d = 29$. Observando esses dois valores públicos,



suponha que Alice queira agora enviar as letras ‘I’, ‘o’, ‘v’ e ‘e’ a Bob. Interpretando cada letra como um número entre 1 e 26 (com ‘a’ sendo 1 e ‘z’ sendo 26), Alice e Bob realizam a criptografia e a decriptação mostradas nas tabelas 8.2 e 8.3, respectivamente. Observe que neste exemplo, consideramos cada uma das quatro letras como uma mensagem distinta. Um exemplo mais realista seria converter as quatro letras em suas representações ASCII de 8 bits e então codificar o número inteiro correspondente ao padrão de bit de 32 bits resultante. (Esse exemplo realista cria números muito longos para publicar neste livro!)

Dado que o exemplo fictício das tabelas 8.2 e 8.3 já produziu alguns números extremamente grandes e visto que sabemos, porque vimos anteriormente, que p e q devem ter, cada um, algumas centenas de bits de comprimento, várias questões práticas nos vêm à mente no caso do RSA. Como escolher números primos grandes? Como escolher e e d ? Como calcular exponenciais de números grandes? A discussão desses assuntos está além do escopo deste livro; consulte [Kaufman, 1995] e as referências ali citadas para mais detalhes.

Chave de sessão

Observamos aqui que a exponenciação exigida pelo RSA é um processo que consome tempo considerável. O DES, ao contrário, é, no mínimo, cem vezes mais veloz em software e entre mil e dez mil vezes mais veloz em hardware [RSA Fast, 2004]. Como resultado, o RSA é frequentemente usado na prática em combinação com a criptografia de chave simétrica. Por exemplo, se Alice quer enviar a Bob uma grande quantidade de dados cifrados a alta velocidade, ela pode fazer o seguinte. Primeiramente ela escolhe uma chave que será utilizada para codificar os dados em si; essa chave às vezes é denominada **chave de sessão**, representada por K_s . Alice deve informar a Bob essa chave de sessão, já que essa é a chave simétrica compartilhada que eles usarão com uma cifra de chave simétrica (por exemplo, DES ou AES). Alice criptografa o valor da chave de sessão usando a chave pública RSA de Bob, isto é, ela processa $c = (K_s)^e \text{ mod } n$. Bob recebe a chave de sessão codificada RSA, c , e a decifra para obter a chave de sessão K_s . Ele agora conhece a chave que Alice usará para transferir dados cifrados em DES.

Por que o RSA funciona?

A criptografia/decriptação do RSA parece mágica. Por que será que, aplicando o algoritmo de criptografia e , em seguida, o algoritmo de decriptação, podemos recuperar a mensagem original? Para entender por que o RSA funciona, tome novamente $n = pq$, onde p e q são os números primos grandes usados no algoritmo RSA.

Lembre-se de que na criptografia RSA uma mensagem (representada por um número inteiro) m é primeiramente elevada à potência e usando-se aritmética de módulo n , ou seja,

$$C = m^e \text{ mod } n$$

A decriptação é feita elevando-se esse valor à potência d , novamente usando a aritmética de módulo n . O resultado de uma etapa de criptografia, seguida de uma etapa de decriptação, é então $(m^e \text{ mod } n)^d \text{ mod } n$. Vamos ver agora o que podemos dizer sobre essa quantidade. Como mencionado anteriormente, uma propriedade importante da aritmética modular é $(a \text{ mod } n)^d \text{ mod } n = a^d \text{ mod } n$ para quaisquer valores a , n e d . Deste modo, usando $a = m^e$ nesta propriedade, temos:

$$(m^e \text{ mod } n)^d \text{ mod } n = m^{ed} \text{ mod } n.$$

Letra do texto aberto	m : representação numérica	m^e	Texto cifrado $c = m^e \text{ mod } n$
I	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Tabela 8.2 Criptografia RSA para Alice: $e = 5$, $n = 35$

Texto cifrado c	c^d	$m = c^d \bmod n$	Letra do texto aberto
17	4819685721067509150915091411825223071697	12	I
15	127834039403948858939111232757568359375	15	O
22	851643319086537701956194499721106030592	22	V
10	1000	5	E

Tabela 8.3 Decriptação RSA para Bob: $d = 29$, $n = 35$

Portanto, falta mostrar que $m^{ed} \bmod n = m$. Embora estejamos tentando eliminar um pouco da mágica do modo de funcionamento do RSA, para explicá-lo, precisaremos usar, aqui, outro resultado bastante mágico da teoria dos números. Especificamente, precisamos do resultado que diga que, se p e q forem primos, $n = pq$, e $z = (p - 1)(q - 1)$, então $x^y \bmod n$ será o mesmo que $x^{(y \bmod z)} \bmod n$ [Kaufman, 1995]. Aplicando esse resultado com $x = m$ e $y = ed$, temos

$$m^{ed} \bmod n \equiv m^{(ed \bmod z)} \bmod n.$$

Mas lembre-se de que escolhemos e e d tais que $ed \equiv 1 \pmod{z-1}$. Isso nos dá

$$m^{ed} \bmod n = m^1 \bmod n = m,$$

que é exatamente o resultado que esperávamos! Efetuando primeiramente a exponenciação da potência e (isto é, criptografando) e depois a exponenciação da potência d (isto é, decriptando), obtemos o valor original m . E mais notável ainda é o fato de que, se primeiramente elevarmos à potência d e, em seguida, à potência e , isto é, se invertermos a ordem da criptografia e da decriptação, realizando inicialmente a operação de decriptação e , em seguida, aplicando a operação de criptografia, também obteremos o valor original m . Esse resultado extraordinário resulta imediatamente da aritmética modular:

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

A segurança do RSA reside no fato de que não se conhecem algoritmos para fatorar rapidamente um número; nesse caso, o valor público n , em números primos p e q . Se alguém conhecesse os números p e q , então, dado o valor público e , poderia facilmente processar a chave secreta d . Por outro lado, não se sabe se existem ou não algoritmos rápidos para fatorar um número e, nesse sentido, a segurança do RSA não é garantida.

Outro conhecido algoritmo de criptografia de chave pública é o Diffie-Hellman, que será explorado resumidamente nos Problemas. O Diffie-Hellman não é tão versátil quanto o RSA, pois não pode ser usado para cifrar mensagens de comprimento arbitrário; ele pode ser usado, entretanto, para determinar uma chave de sessão simétrica, que, por sua vez, é utilizada para codificar mensagens.

8.3 Integridade de mensagem e autenticacão do ponto final

Na seção anterior, vimos como a criptografia pode ser usada para oferecer sigilo a duas entidades em comunicação. Nesta seção, nos voltamos ao assunto igualmente importante da criptografia que é prover integridade da mensagem (também conhecida como autenticação da mensagem). Além da integridade da mensagem, discutiremos dois assuntos análogos nesta seção: assinaturas digitais e autenticação do ponto final.

Definimos o problema de integridade da mensagem usando, mais uma vez, Alice e Bob. Suponha que Bob receba uma mensagem (que pode ser cifrada ou estar em texto aberto) acreditando que ela tenha sido enviada por Alice. Para autenticar a mensagem, Bob precisa verificar se:

1. A mensagem foi, realmente, enviada por Alice.
 2. A mensagem não foi alterada a caminho de Bob.

Veremos nas Seções 8.4 a 8.7 que esse problema de integridade da mensagem é uma preocupação importante em todos os protocolos de rede seguros.

Como um exemplo específico, considere uma rede de computadores que está utilizando um algoritmo de roteamento de estado de enlace (como OSPF) para determinar rotas entre cada dupla de roteadores na rede (veja Capítulo 4). Em um algoritmo de estado de enlace, cada roteador precisa transmitir uma mensagem de estado de enlace a todos os outros roteadores na rede. Uma mensagem de estado de enlace do roteador inclui uma relação de seus vizinhos diretamente conectados e os custos diretos a eles. Uma vez que o roteador recebe mensagens de estado de enlace de todos os outros roteadores, ele pode criar um mapa completo da rede, executar seu algoritmo de roteamento de menor custo e configurar sua tabela de repasse. Um ataque relativamente fácil no algoritmo de roteamento é Trudy distribuir mensagens de estado de enlace falsas com informações incorretas sobre o estado de enlace. Assim, a necessidade de integridade da mensagem — quando o roteador B recebe uma mensagem de estado de enlace do roteador A, esse deve verificar que o roteador A na verdade criou a mensagem e que ninguém alterou a mensagem em trânsito.

Nesta seção descrevemos uma técnica conhecida sobre integridade da mensagem usada por muitos protocolos de rede seguros. Mas antes disso, precisamos abordar outro importante tópico na criptografia — as funções de hash criptográficas.

8.3.1 Funções de hash criptográficas

Como mostrado na Figura 8.7, a função de hash recebe uma entrada, m , e computa uma cadeia de tamanho fixo $H(m)$ conhecida como hash. A soma de verificação da Internet (Capítulo 3) e os CRCs (Capítulo 4) satisfazem essa definição. Uma função hash criptográfica deve apresentar a seguinte propriedade adicional:

Em termos de processamento, é impraticável encontrar duas mensagens diferentes x e y tal que $H(x) = H(y)$.

Informalmente, essa propriedade significa que, em termos de processamento, é impraticável que um invasor substitua uma mensagem por outra que está protegida pela função hash. Ou seja, se $(m, H(m))$ é a mensagem e o hash da mensagem criada pelo emissor, então um invasor não pode forjar os conteúdos de outra mensagem, y , que possui o mesmo valor de hash que a mensagem original.

É bom nos convencermos de que uma simples soma de verificação, como a da Internet, daria um péssimo algoritmo de resumo de mensagem. Em vez de processar a aritmética de complemento de 1 (como é feito para a soma de verificação da Internet), vamos efetuar uma soma de verificação tratando cada caractere como um byte e somando os bytes usando porções de 4 bytes por vez. Suponha que Bob deva a Alice 100,99 dólares e lhe envie um vale contendo a sentença encadeada 'IOU100.99BOB.' (IOU — I Owe You — Eu lhe devo.) A representação ASCII (em notação hexadecimal) para essas letras é 49, 4F, 55, 31, 30, 30, 2E, 39, 39, 42, 4F, 42.

A Figura 8.8 (parte de cima) mostra que a soma de verificação de 4 bytes para essa mensagem é B2 C1 D2 AC. Uma mensagem ligeiramente diferente (e que sairia muito mais cara para Bob) é mostrada na parte de baixo da

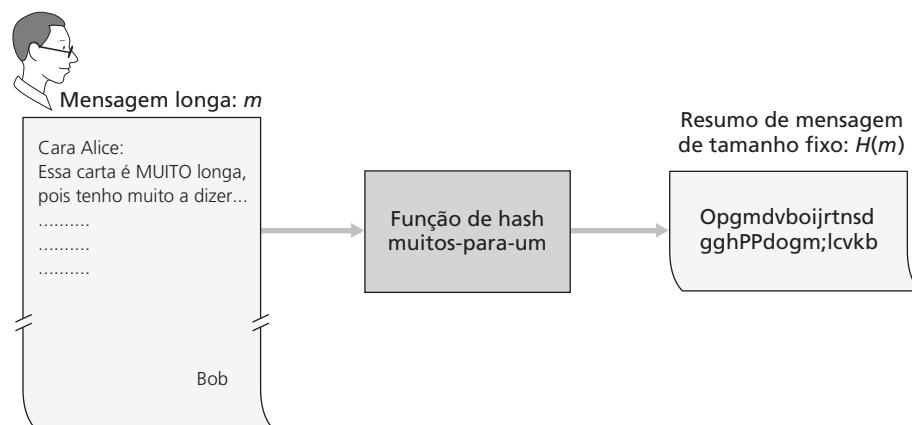


Figura 8.7 Funções de Hash

Mensagem	Representação ASCII				Soma de verificação
	B2	C1	D2	AC	
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
					Soma de verificação

Mensagem	Representação ASCII				Soma de verificação
	B2	C1	D2	AC	
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
					Soma de verificação

Figura 8.8 Mensagem inicial e mensagem fraudulenta têm a mesma soma de verificação!

Figura 8.8. As mensagens ‘IOU100.99BOB’ e ‘IOU900.19BOB’ têm a mesma soma de verificação. Assim, esse algoritmo simples viola as duas exigências citadas anteriormente. Fornecidos os dados originais, é simples descobrir outro conjunto de dados com a mesma soma de verificação. É claro que, para efeito de segurança, precisaremos de uma função de hash muito mais poderosa do que uma soma de verificação.

O algoritmo de hash MD5 de Ron Rivest [RFC 1321] é amplamente usado hoje. Ele processa um resumo de mensagem de 128 bits por meio de um processo de quatro etapas, constituído de uma etapa de enchimento (adição de um ‘um’ seguido de ‘zeros’ suficientes, de modo que o comprimento da mensagem satisfaça determinadas condições), uma etapa de anexação (anexação de uma representação de 64 bits do comprimento da mensagem antes do enchimento), uma etapa de inicialização de um acumulador e uma etapa final iterativa, na qual os blocos de 16 palavras da mensagem são processados (misturados) em quatro rodadas de processamento. Para uma descrição do MD5 (incluindo uma implementação em código de fonte C), consulte [RFC 1321].

O segundo principal algoritmo de hash em uso atualmente é o SHA-1 (*secure hash algorithm* — algoritmo de hash seguro) [FIPS, 1995]. Esse algoritmo se baseia em princípios similares aos usados no projeto do MD4 [RFC 1320], o predecessor do MD5. O uso do SHA-1, um padrão federal norte-americano, é exigido sempre que aplicações de âmbito federal precisarem de um algoritmo de resumo de mensagem seguro. Ele produz um resumo de mensagem de 160 bits. O resultado mais longo torna o SHA-1 mais seguro.

8.3.2 Código de autenticação da mensagem

Retornemos ao problema da integridade da mensagem. Agora que compreendemos as funções de hash, vamos tentar entender como podemos garantir a integridade da mensagem:

1. Alice cria a mensagem m e calcula o hash $H(m)$ (por exemplo, com SHA-1).
2. Alice então anexa $H(m)$ à mensagem m , criando uma mensagem extendida $(m, H(m))$, e a envia para Bob.
3. Bob recebe uma mensagem extendida (m, h) e calcula $H(m)$. Se $H(m) = h$, Bob conclui que está tudo certo.

Essa abordagem é, obviamente, errônea. Trudy pode criar uma mensagem m' falsa, passar-se por Alice, calcular $H(m')$ e enviar $(m', H(m'))$ a Bob. Quando Bob receber a mensagem, tudo se encaixa na etapa 3, então ele não suspeita de nada.

Para realizar a integridade da mensagem, além de usar as funções de hash criptográficas, Alice e Bob precisarão de um segredo compartilhado s , que não é nada mais do que uma cadeia de bits denominada chave de autenticação. Utilizando esse segredo compartilhado, a integridade da mensagem pode ser realizada da seguinte maneira:

1. Alice cria a mensagem m , concatena s com m para criar $m + s$, e calcula o hash $H(m+s)$ (por exemplo, com SHA-1). $H(m+s)$ é denominado o código de autenticação da mensagem (MAC).

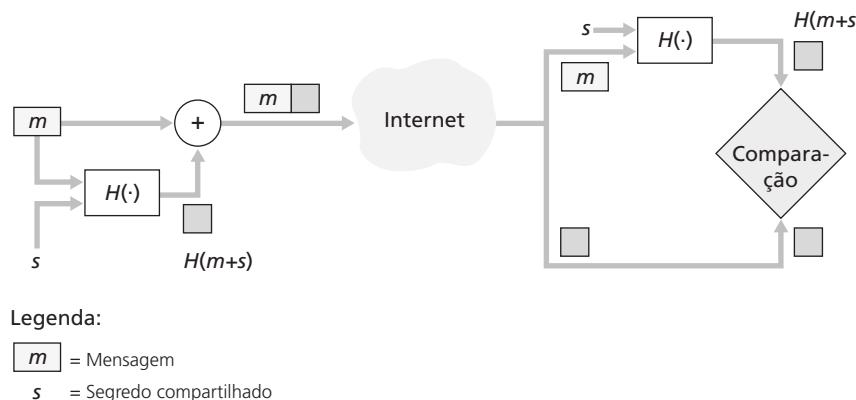


Figura 8.9 Código de autenticação de mensagem (MAC)

2. Alice então anexa o MAC à mensagem m , criando uma mensagem extendida $(m, H(m+s))$, e a envia para Bob.
3. Bob recebe uma mensagem extendida (m, h) e conhecendo s , calcula o MAC $H(m+s)$. Se $H(m+s) = h$, Bob conclui que está tudo certo.

Um resumo desse processo é ilustrado na Figura 8.9. É importante observar que o MAC (abreviação de *message authentication code* [código de autenticação da mensagem]), neste caso, não é o mesmo MAC utilizado nos protocolos da camada de enlace (abreviação de *medium access control* [controle de acesso ao meio])!

Um bom recurso do MAC é o fato de ele não exigir um algoritmo de criptografia. Realmente, em muitas aplicações, incluindo o algoritmo de roteamento de estado do enlace, descrito anteriormente, as entidades de comunicação somente estão preocupadas com a integridade da mensagem e não com o seu sigilo. Utilizando um MAC, as entidades podem autenticar as mensagens que enviam uma à outra sem ter de integrar algoritmos de criptografia complexos ao processo de integridade.

Como você pode esperar, muitos padrões diferentes para os MACs foram propostos ao longo dos anos. Atualmente, o mais popular é o HMAC, que pode ser usado com o MD5 ou SHA-1. O HMAC, na verdade, passa os dados e a chave de autenticação duas vezes pela função de hash [Kaufman, 1995; RFC 2104].

Ainda resta um assunto importante. Como distribuímos a chave de autenticação compartilhada às entidades de comunicação? No algoritmo de roteamento de estado do enlace, por exemplo, precisaríamos, de alguma forma, distribuir a chave de autenticação secreta a cada um dos roteadores no sistema independente. (Observe que os roteadores podem usar a mesma chave de autenticação.) Um administrador de rede poderia, de fato, conseguir esse feito visitando fisicamente cada um dos roteadores. Ou, se o administrador de rede for preguiçoso, e se cada roteador possuir sua própria chave pública, o administrador poderia distribuir a chave de autenticação a qualquer um dos roteadores criptografando-a com a chave pública do roteador e, então, enviar a chave cifrada por meio da rede ao roteador.

8.3.3 Assinaturas digitais

Pense no número de vezes em que você assinou seu nome em um pedaço de papel durante a última semana. Você assina cheques, comprovantes de operação de cartões de crédito, documentos legais e cartas. Sua assinatura atesta o fato de que você (e não outra pessoa) conhece o conteúdo do documento e/ou concorda com ele. No mundo digital, frequentemente se quer indicar o dono ou o criador de um documento ou deixar claro que alguém concorda com o conteúdo de um documento. A **assinatura digital** é uma técnica criptográfica usada para cumprir essas finalidades no mundo digital.

Exatamente como acontece com as assinaturas por escrito, a assinatura digital deve ser verificável e não falsificável. Isto é, deve ser possível provar que um documento assinado por um indivíduo foi na verdade assinado

por ele (a assinatura tem de ser verificável) e que somente aquele indivíduo poderia ter assinado o documento (a assinatura não pode ser falsificada).

Vamos considerar agora como podemos criar um método de assinatura digital. Observe que quando Bob assina uma mensagem, ele deve colocar algo nela que seja único para ele. Bob poderia adicionar um MAC à assinatura, sendo o MAC criado ao adicionar sua chave (única para ele) à mensagem e, depois, formar o hash. Mas para Alice verificar a assinatura, ela deve também ter uma cópia da chave, que não seria única para Bob. Portanto, os MACs não se incluem nesse processo.

Lembre-se de que com a criptografia de chave pública, Bob possui tanto uma chave pública como uma privada, as quais são únicas para Bob. Dessa maneira, a criptografia de chave pública é uma excelente candidata para prover assinaturas digitais. Vamos verificar como isso é feito.

Suponha que Bob queira assinar digitalmente um documento, m . Imagine que o documento seja um arquivo ou uma mensagem que Bob vai assinar e enviar. Como mostra a Figura 8.10, para assinar esse documento Bob simplesmente usa sua chave criptográfica privada K_B^- para processar $K_B^-(m)$. A princípio, pode parecer estranho que Bob esteja usando sua chave privada (que, como vimos na Seção 8.2, foi usada para decriptar uma mensagem que tinha sido criptografada com sua chave pública) para assinar um documento. Mas lembre-se de que criptografia e decriptação nada mais são do que uma operação matemática (exponenciação à potência e ou d no RSA; veja a Seção 8.2) e que a intenção de Bob não é embaralhar ou disfarçar o conteúdo do documento, mas assinar o documento de maneira que este seja verificável, não falsificável e contestável. Bob tem o documento, m , e sua assinatura digital do documento é $K_B^-(m)$.

A assinatura digital $K_B^-(m)$ atende às nossas exigências de ser verificável e não falsificável? Suponha que Alice tenha m e $K_B^-(m)$. Ela quer provar na Justiça (em ação litigiosa) que Bob de fato assinou o documento e que ele era a única pessoa que poderia tê-lo assinado. Alice pega a chave pública de Bob, $K_B^+(m)$, e a aplica à assinatura digital $K_B^-(m)$ associada ao documento m . Isto é, ela processa $K_B^+(K_B^-(m))$, e, voilà, com dramática encenação, produz m , que é uma reprodução exata do documento original! Ela então argumenta que somente Bob poderia ter assinado o documento pelas seguintes razões:

■ Quem quer que tenha assinado o documento deve ter usado a chave criptográfica privada, K_B^- , para processar a assinatura $K_B^-(m)$, de modo que $K_B^+(K_B^-(m)) = m$.

■ A única pessoa que poderia conhecer a chave privada K_B^- , é Bob. Lembre-se de que dissemos em nossa discussão do RSA, na Seção 8.2, que conhecer a chave pública, K_B^+ , não serve para descobrir a chave privada, K_B^- . Portanto, a única pessoa que poderia conhecer K_B^- é aquela que gerou o par de chaves, (K_B^-, K_B^+) , em primeiro lugar, Bob. (Note que, para isso, admitimos que Bob não passou K_B^- a ninguém e que ninguém roubou K_B^- de Bob.)

Também é importante notar que, se o documento original, m , for modificado para algum modelo alternativo m' , a assinatura que Bob criou para m não será válida para m' , já que $K_B^+(K_B^-(m))$ não é igual a m' . Assim, obser-

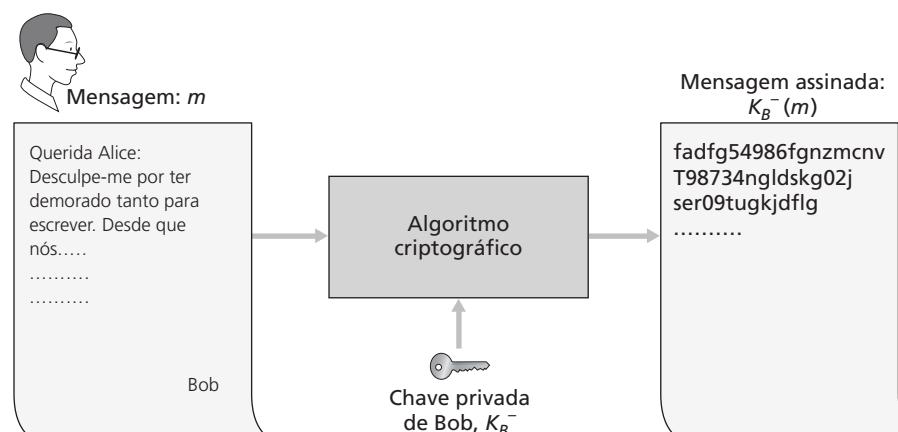


Figura 8.10 Criação de uma assinatura digital para um documento

vamos que as assinaturas digitais também oferecem integridade da mensagem, permitindo que o receptor verifique se a mensagem foi alterada, bem como sua fonte.

Uma preocupação com os dados de assinatura por criptografia é que a criptografia e a decriptação prejudicam o desempenho do computador. Dados os *overheads* de criptografia e decriptação, os dados de assinatura por meio de criptografia/decriptação completa podem ser excessivos. Uma abordagem mais eficiente é introduzir as funções de hash na assinatura digital. Na Seção 8.3.2, um algoritmo de hash pega uma mensagem, m , de comprimento arbitrário e computa uma “impressão digital” de comprimento fixo da mensagem, representada por $H(m)$. Usando uma função de hash, Bob assina o hash da mensagem em vez de assinar a própria mensagem, ou seja, Bob calcula $K_B^{-1}(H(m))$. Uma vez que $H(m)$ é geralmente muito menor do que a mensagem original m , o esforço computacional necessário para criar a assinatura digital é reduzido consideravelmente.

Em relação a Bob enviar uma mensagem para Alice, a Figura 8.11 apresenta um resumo do procedimento operacional de criar uma assinatura digital. Bob coloca sua longa mensagem original em uma função de hash. Ele, então, assina digitalmente o hash resultante com sua chave privada. A mensagem original (em texto claro) junto com o resumo de mensagem assinada digitalmente (de agora em diante denominada assinatura digital) é então enviada para Alice. A Figura 8.12 apresenta um resumo do processo operacional da assinatura. Alice aplica a função de hash à mensagem para obter um resultado de hash. Alice também aplica a função de hash à mensagem em texto claro para obter um segundo resultado de hash. Se dois hashes forem compatíveis, então Alice pode ter certeza sobre a integridade e o autor da mensagem.

Antes de seguirmos em frente, vamos fazer uma breve comparação entre as assinaturas digitais e os MACs, visto que eles são paralelos, mas também têm diferenças sutis e importantes. As assinaturas digitais e os MACs iniciam com uma mensagem (ou um documento). Para criar um MAC por meio de mensagem, inserimos uma chave de autenticação à mensagem e, depois, pegamos o hash do resultado. Observe que nem a chave pública nem a criptografia de chave simétrica estão envolvidas na criação do MAC. Para criar uma assinatura digital, primeiro pegamos o hash da mensagem e, depois, ciframos a mensagem com nossa chave privada (usando a criptografia de chave pública). Assim, uma assinatura digital é uma técnica “mais pesada”, uma vez que ela exige uma Infraestrutura de Chave Pública (PKI) subjacente com autoridades de certificação conforme descritas abaixo. Veremos na Seção 8.4 que o PGP — um sistema seguro de e-mail renomado — utiliza assinaturas digitais para a

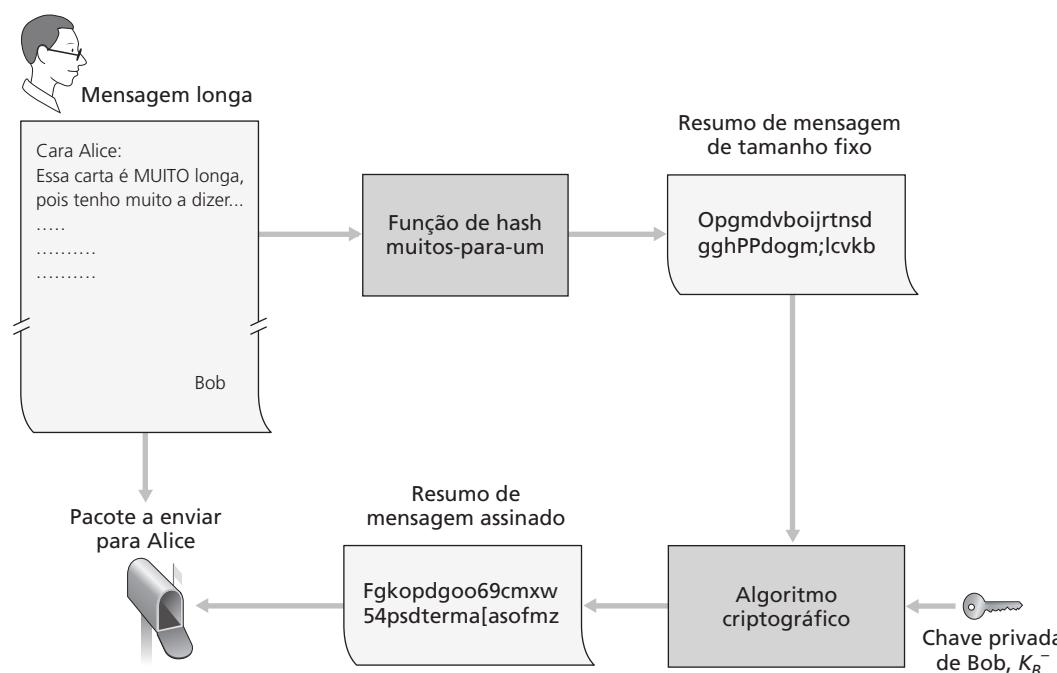


Figura 8.11 Envio de uma mensagem assinada digitalmente

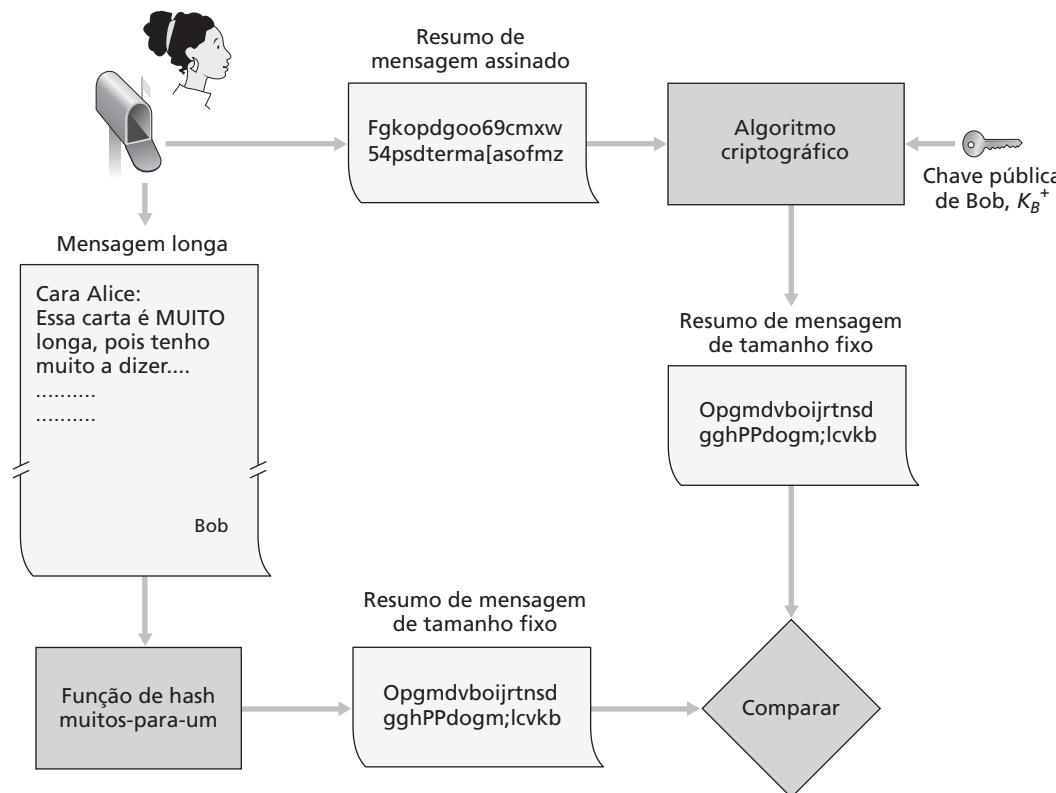


Figura 8.12 Verificação da integridade de uma mensagem assinada

integridade da mensagem. Já vimos que o OSPF usa MACs para a integridade da mensagem. Veremos nas seções 8.5 e 8.6 que os MACs são também usados pelos conhecidos protocolos de segurança da camada de transporte e da camada de rede.

Certificação de chaves públicas

Uma aplicação importante de assinaturas digitais é a certificação de chaves públicas, ou seja, certificar que uma chave pública pertence a uma entidade específica. A certificação de chaves públicas é usada em muitos protocolos de rede seguros, incluindo o IPsec e o SSL.

Para compreender mais sobre o problema, consideremos uma versão de comércio eletrônico para o clássico “trote da pizza”. Suponha que Alice trabalhe no ramo de pizzas para viagem e que aceite pedidos pela Internet. Bob, que adora pizza, envia a Alice uma mensagem em texto aberto que contém o endereço de sua casa e o tipo de pizza que quer. Nessa mensagem, ele inclui também uma assinatura digital (isto é, um hash assinado extraído da mensagem original em texto aberto) para provar a Alice que ele é a fonte verdadeira da mensagem. Para verificar a assinatura, Alice obtém a chave pública de Bob (talvez de um servidor de chaves públicas ou de uma mensagem de e-mail) e verifica a assinatura digital. Dessa maneira, ela se certifica de que foi Bob, e não algum adolescente brincalhão, quem fez o pedido.

Tudo parece caminhar bem até que entra em cena a esperta Trudy. Como mostrado na Figura 8.13, Trudy decide fazer uma travessura. Ela envia uma mensagem a Alice na qual diz que é Bob, fornece o endereço de Bob e pede uma pizza. Nessa mensagem ela também inclui sua (de Trudy) chave pública, embora Alice admita, naturalmente, que seja a de Bob. Trudy também anexa uma assinatura digital, a qual foi criada com sua própria (de Trudy) chave privada. Após receber a mensagem, Alice aplica a chave pública de Trudy (pensando que é a de Bob) à assinatura digital e concluirá que a mensagem em texto aberto foi, na verdade, criada por Bob. Este ficará muito surpreso quando o entregador aparecer em sua casa com uma pizza bem variada!

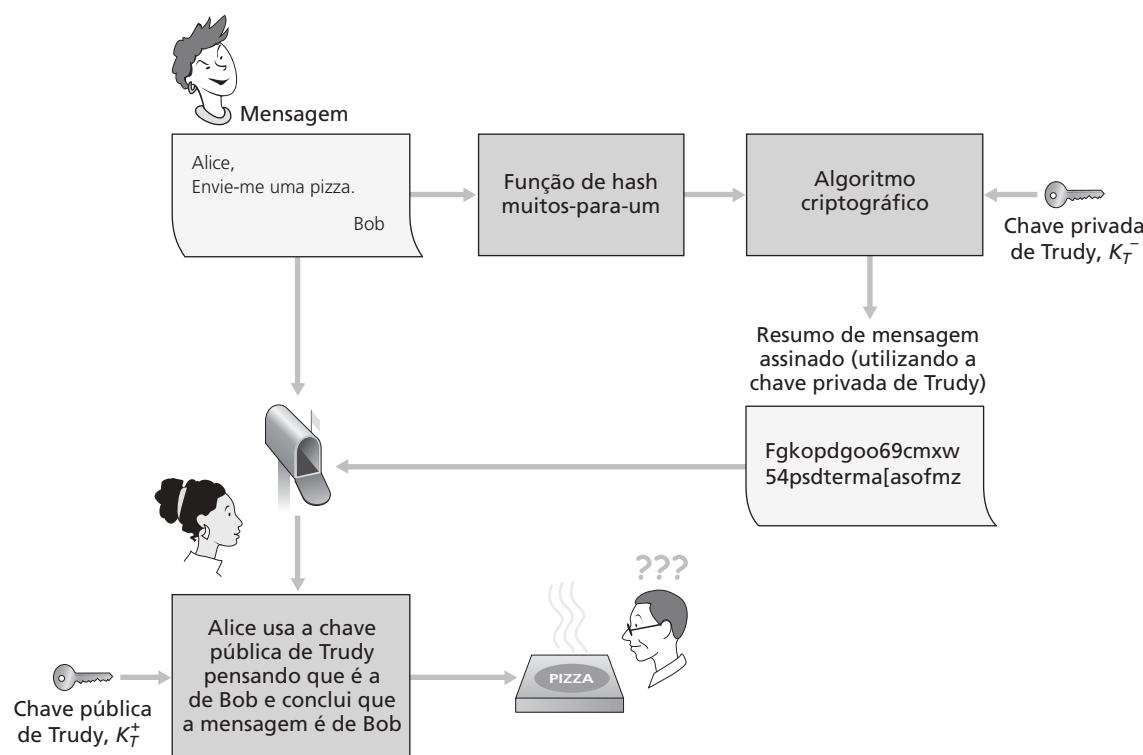


Figura 8.13 Trudy se passa por Bob usando criptografia de chaves públicas

Por esse exemplo, vemos que, para que a criptografia de chaves públicas seja útil, você precisa ser capaz de verificar se você tem a chave pública verdadeira da entidade (usuários, browsers, roteadores e outras) com a qual quer se comunicar. Por exemplo, quando Alice estiver se comunicando com Bob usando criptografia de chaves públicas, ela precisará saber, com certeza, que a chave pública que supostamente é de Bob é, de fato, dele.

A vinculação de uma chave pública a uma entidade particular é feita, tipicamente, por uma autoridade certificadora (*certification authority* — CA), cuja tarefa é validar identidades e emitir certificados. Uma CA tem as seguintes incumbências:

1. Uma CA verifica se uma entidade (pessoa, roteador e assim por diante) é quem diz ser. Não há procedimentos obrigatórios para o modo como deve ser feita a certificação. Quando tratamos com uma CA, devemos confiar que ela tenha realizado uma verificação de identidade adequadamente rigorosa. Por exemplo, se Trudy conseguisse entrar na autoridade certificadora Fly-by-Night, e simplesmente declarasse “Eu sou Alice” e recebesse certificados associados à identidade de Alice, então não se deveria dar muita credibilidade a chaves públicas certificadas pela autoridade certificadora Fly-by-Night. Por outro lado, seria mais sensato (ou não!) estar mais inclinado a confiar em uma CA que faz parte de um programa federal ou estadual. O grau de confiança que se tem na identidade associada a uma chave pública equivale apenas ao grau de confiança depositada na CA e em suas técnicas de verificação de identidades. Que rede emaranhada de confiança estamos tecendo!
2. Tão logo verifique a identidade da entidade, a CA cria um **certificado** que vincula a chave pública da entidade à identidade verificada. O certificado contém a chave pública e a informação exclusiva que identifica mundialmente o proprietário da chave pública (por exemplo, o nome de alguém ou um endereço IP). Essas etapas são mostradas na Figura 8.14.

Vejamos, agora, como certificados podem ser usados para combater os espertinhos das pizzas, como Trudy e outros indesejáveis. Quando Bob faz seu pedido, ele também envia seu certificado assinado por uma CA. Alice usa a chave pública da CA para verificar a validade do certificado de Bob e extrair a chave pública dele.

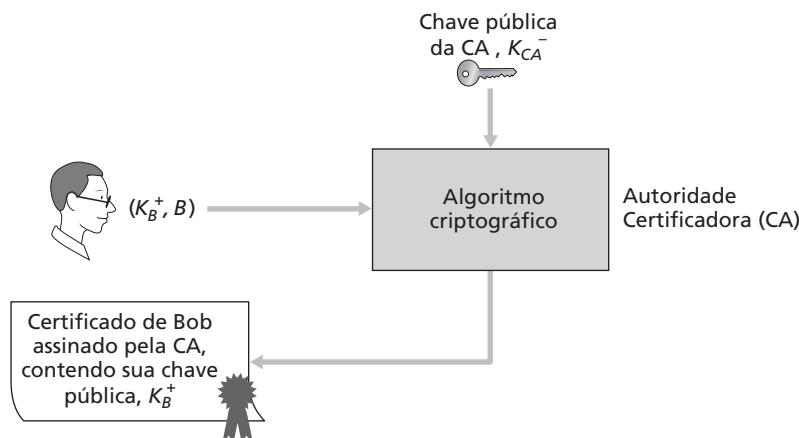


Figura 8.14 Bob obtém sua chave pública certificada pela CA

Tanto a International Telecommunication Union (ITU) como a IETF desenvolveram padrões para autoridades certificadoras. Na recomendação ITU X.509 [ITU, 1993], encontramos a especificação de um serviço de autenticação, bem como uma sintaxe específica para certificados.

O RFC 1422 descreve um gerenciamento de chaves baseado em CA para utilização com o e-mail seguro pela Internet. Essa recomendação é compatível com X.509, mas vai além desta, pois estabelece procedimentos e convenções para uma arquitetura de gerenciamento de chaves. A Tabela 8.4 apresenta alguns campos importantes de um certificado.

8.3.4 Autenticação do ponto final

A autenticação do ponto final é o processo de provar a identidade de uma pessoa a alguém. Como seres humanos, autenticamos uns aos outros de diversas maneiras: reconhecemos o rosto um dos outros ao nos encontrarmos. Reconhecemos a voz um do outro no telefone. Somos autenticados por um oficial da Alfândega que compara a nossa foto com a do passaporte. Mas ao realizar a autenticação através da rede, as partes em comunicação não podem confiar nas informações biométricas, como uma aparência visual ou um timbre de voz. Realmente, muitas vezes os elementos da rede, como roteadores e processos cliente-servidor, devem autenticar uns aos outros somente baseados na troca de mensagens e dados.

A primeira questão que abordamos é se os MACs (estudados na Seção 8.3.2) podem ser usados para a autenticação do ponto final. Suponha que Alice e Bob compartilhem um segredo comum s , e que Alice queira enviar uma mensagem (por exemplo, um segmento TCP) a Bob e ele queira ter certeza de que essa mensagem tenha sido

Nome do campo	Descrição
Versão	Número da versão da especificação X.509
Número de série	Identificador exclusivo emitido pela CA para um certificado
Assinatura	Especifica o algoritmo usado pela CA para assinar esse certificado
Nome do emissor	Identidade da CA que emitiu o certificado, em formato de nome distinto (DN) [RFC 2253]
Período de validade	Início e fim do período de validade do certificado
Nome do sujeito	Identidade da entidade cuja chave pública está associada a esse certificado, em formato DN
Chave pública do sujeito	A chave pública do sujeito, bem como uma indicação do algoritmo de chave pública (e parâmetros do algoritmo) a ser usado com essa chave

Tabela 8.4 Campos selecionados de uma chave pública X.509 e RFC 1422

enviada por Alice. A abordagem natural baseada no MAC é a seguinte. Alice cria um MAC usando a mensagem e o segredo compartilhado, anexa o MAC à mensagem e envia a “mensagem estendida” resultante para Bob. Quando Bob recebe a mensagem extendida, ele usa o MAC interior para verificar a fonte e a integridade da mensagem. De fato, como Alice e Bob compartilham um segredo, os cálculos do MAC de Bob apresentam um resultado semelhante ao MAC na mensagem estendida, então Bob sabe, com certeza, que Alice enviou a mensagem (e que a mensagem não foi alterada em trânsito).

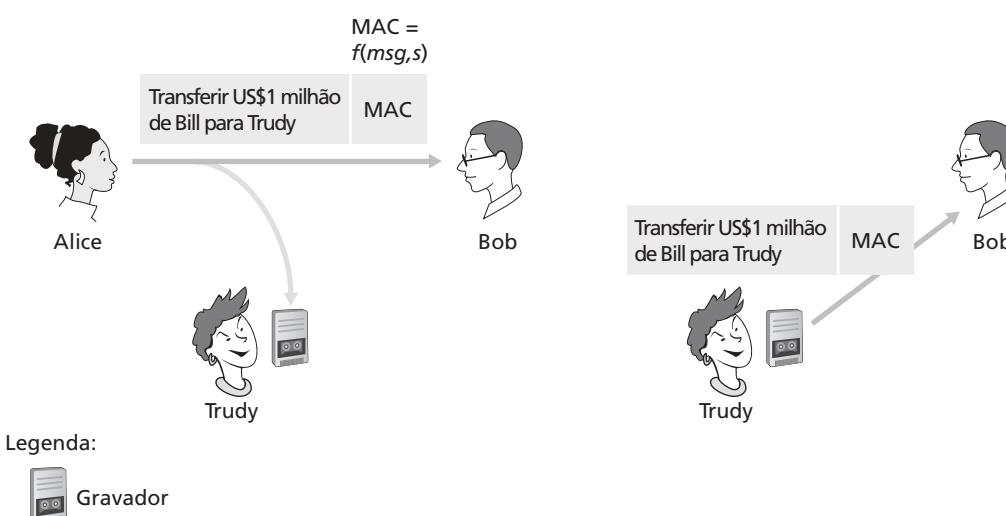
Ataque de reprodução e nonces

Será que ele sabe mesmo? Bob não tem 100% de certeza sobre a fonte da mensagem porque ainda há a possibilidade de ele estar sendo enganado com um ataque de reprodução. Como mostrado na Figura 8.15, Trudy precisa somente analisar e gravar a mensagem estendida de Alice e reproduzi-la algum tempo depois. A mensagem repetida poderia ser “Concordo em transferir um milhão de dólares da conta de Bill para a conta de Trudy.”, fazendo com que um total de dois milhões de dólares sejam transferidos; ou a mensagem poderia ser “O enlace do Roteador Alice para o Roteador Charlie parou de funcionar”, que, se enviada após o enlace ter sido consertado, poderia causar configurações errôneas nas tabelas de repasse.

O cenário de falha na Figura 8.15 resultou do fato de Bob não conseguir distinguir entre a mensagem original enviada por Alice e a reprodução da mensagem original de Alice. Ou seja, Bob não conseguiu saber se Alice estava ao vivo (isto é, realmente estava no outro ponto da conexão) ou se a mensagem que ele recebeu foi uma reprodução. O leitor muito (*muito*) atento se lembrará de que o protocolo de apresentação de três vias precisou abordar o mesmo problema — o lado do servidor de uma conexão TCP não queria aceitar uma conexão se o segmento SYN recebido fosse uma cópia antiga (retransmissão) de um segmento SYN de uma conexão anterior. Como o lado do servidor TCP resolveu o problema de determinar se o cliente estava mesmo ao vivo? Ele escolheu um número de sequência inicial que não havia sido usado por um bom tempo, enviou esse número ao cliente, e então aguardou que o cliente respondesse com um segmento ACK contendo esse número. Podemos adotar a mesma ideia para fins de autenticação.

Um **nonce** é um número que o protocolo usará somente uma vez por toda a vida. Ou seja, uma vez que o protocolo utilizar um nonce, esse número nunca mais será utilizado. Como mostrado na Figura 8.16, nosso novo protocolo usa um nonce da seguinte forma:

1. Bob escolhe um nonce, R , e o envia a Alice. Observe que o nonce é enviado em aberto. Alice agora cria o MAC utilizando sua mensagem original, o segredo compartilhado, s , e o nonce, R . (Para criar, o MAC Alice pode, por exemplo, concatenar o segredo compartilhado e o nonce com a mensagem e obter um



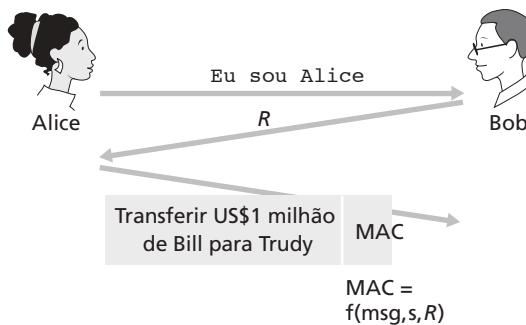


Figura 8.16 Defesa contra o ataque de reprodução com um nonce

resumo através de um hash.) Alice anexa o MAC à mensagem, cria uma mensagem estendida e a envia para Bob.

2. Bob calcula o MAC da mensagem (contido na mensagem estendida), um nonce R e segredo compartilhado s . Se o MAC resultante for igual ao MAC da mensagem extendida, Bob sabe não somente que Alice criou a mensagem mas também que ela criou a mensagem *após* Bob ter enviado um nonce, visto que o valor do nonce é necessário para computar o MAC correto.

Como veremos na Seção 8.5 e 8.6, quando discutimos sobre SSL e IPsec, a combinação de nonces e MACs muitas vezes é utilizada em protocolos de rede seguros para prover uma integridade da mensagem e uma autenticação do ponto final.

Mas e se Alice quiser enviar uma série de mensagens — por exemplo, uma série de segmentos TCP? Bob terá de enviar a Alice um novo nonce para cada uma das mensagens? Na prática, somente um nonce é realmente necessário. Como veremos na Seção 8.5, quando discutirmos sobre SSL, um único nonce combinado a números de sequência permitirá que Bob verifique a originalidade de todas as mensagens que ele recebe de Alice.

Autenticação com criptografia de chave pública

O uso de um nonce e de um segredo compartilhado compuseram a base de um protocolo de autenticação bem-sucedido. Uma questão natural é se podemos usar um nonce e uma criptografia de chave pública para resolver o problema da autenticação. O uso de uma abordagem sobre chave pública evitaria uma dificuldade em qualquer sistema de segredo compartilhado — preocupando-se, primeiramente, em como as duas partes sabem o valor do segredo compartilhado. Um protocolo natural que usa essa criptografia de chave pública é:

1. Alice envia a Bob a mensagem “Eu sou Alice”.
2. Bob escolhe um nonce, R , e o envia para Alice. O nonce será usado, novamente, para se assegurar de que Alice está ao vivo.
3. Alice usa sua chave privada K_A^- para cifrar o nonce e enviar o valor resultante $K_A^-(R)$ para Bob. Visto que somente Alice sabe sua chave privada, só ela pode criar $K_A^-(R)$.
4. Bob aplica a chave pública de Alice, K_A^+ , à mensagem recebida; ou seja, Bob computa $K_A^+(K_A^-(R))$. Lembre-se de nossa discussão sobre a criptografia de chave pública RSA na Seção 8.2 que $K_A^+(K_A^-(R)) = R$. Assim, Bob computa r e autentica Alice.

A operação desse protocolo é ilustrada na Figura 8.17. Esse protocolo é seguro? Uma vez que usa técnicas da chave pública, é necessário que Bob recupere a chave pública de Alice. Isto nos leva a um cenário interessante, mostrado na Figura 8.18, na qual Trudy pode ser capaz de se passar por Alice para Bob.

1. Trudy envia a Bob a mensagem “Eu sou Alice”.
2. Bob escolhe um nonce, R , e o envia para Alice, mas a mensagem é captada por Trudy.

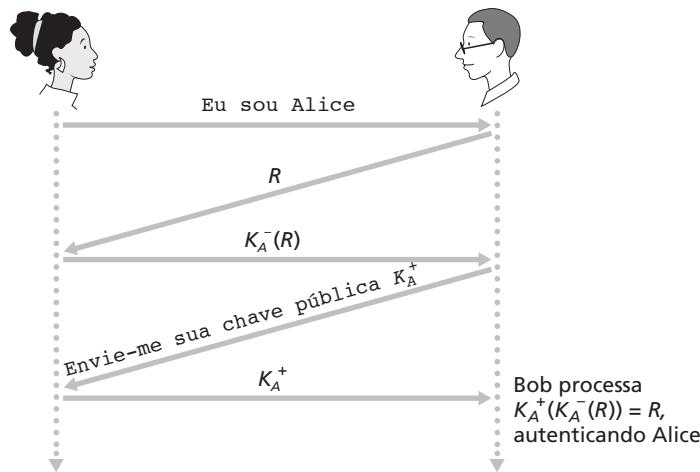


Figura 8.17 Funcionamento correto do protocolo de autenticação da chave pública

3. Trudy usa sua chave privada K_T^- para cifrar um nonce e envia o valor resultante, $K_T^-(R)$, para Bob. Para ele, $K_T^-(R)$ é apenas um grupo de bits e ele não sabe se os bits representam $K_T^-(R)$ ou $K_A^-(R)$.
4. Bob deve obter agora a chave pública de Alice para aplicar K_A^+ ao valor que ele acaba de receber. Ele envia uma mensagem a Alice solicitando o K_A^+ (Bob pode também recuperar a chave pública de Alice de seu site Web). Trudy intercepta essa mensagem também e responde a Bob com K_T^+ , ou seja, a chave pública de Trudy. Bob computa $K_T^+(K_T^-(R)) = R$, autenticando, assim, Trudy como Alice!

A partir desse cenário, está claro que o protocolo de autenticação de chave pública é somente tão seguro quanto a distribuição de chaves públicas. Felizmente, podemos usar certificados para distribuir chaves públicas de forma segura, como vimos na Seção 8.3.

No cenário da Figura 8.18, Bob e Alice podem, finalmente, descobrir que há algo errado, pois Bob afirmará que interagiu com Alice, mas Alice sabe que nunca interagiu com Bob. Existe um ataque ainda mais traiçoeiro que impediria essa detecção. Na Figura 8.19, Alice e Bob estão conversando, mas para aproveitar o mesmo buraco no protocolo de autenticação, Trudy é capaz de se interpor *de forma transparente* entre Alice e Bob. Em particular, se Bob começar a enviar dados cifrados para Alice usando a chave de criptografia, que ele recebe de Trudy, esta pode

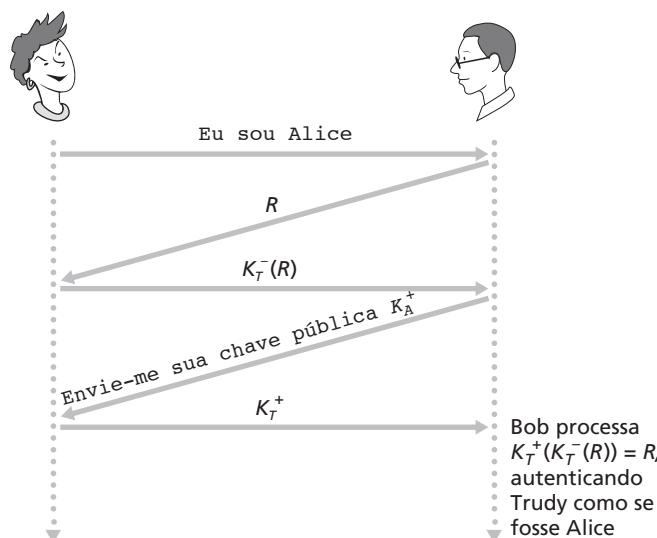


Figura 8.18 Buraco de segurança no protocolo de autenticação da chave pública

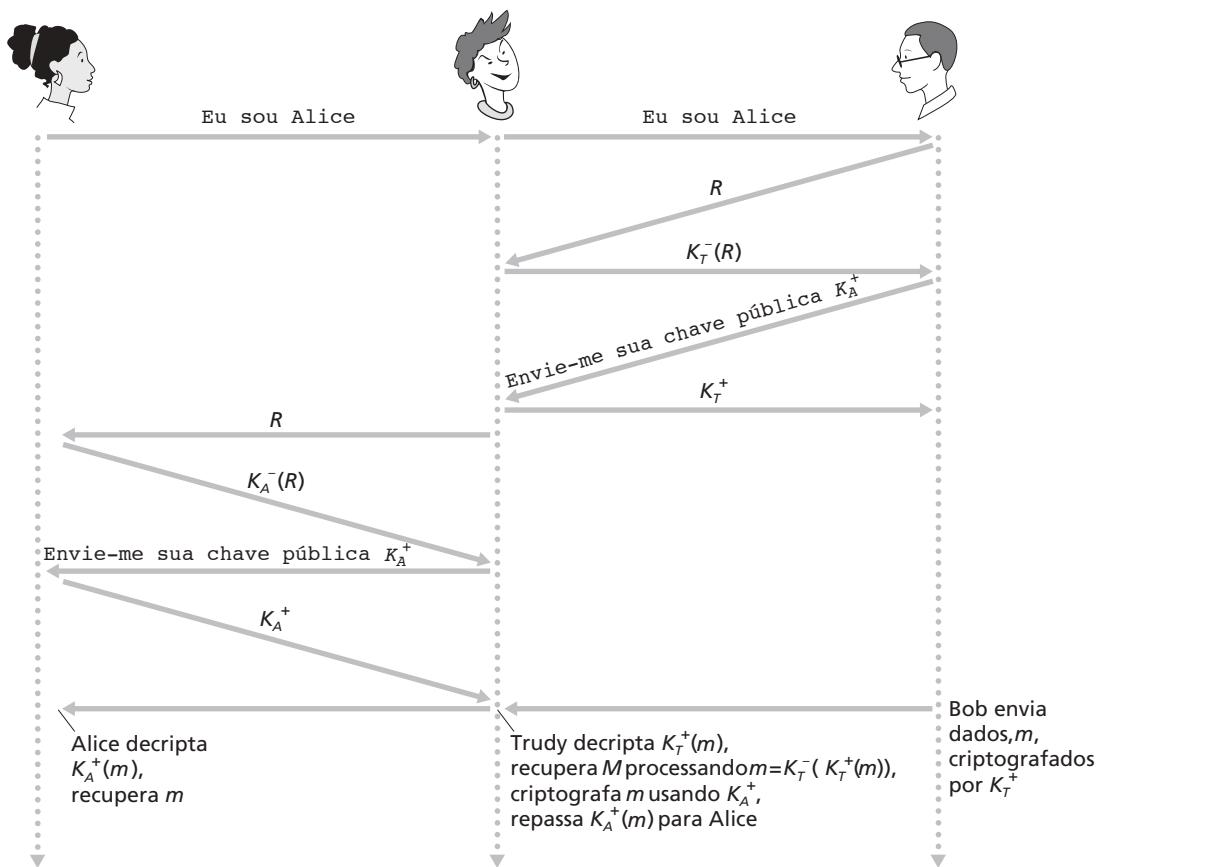


Figure 8.19 O ataque do homem do meio

recuperar o texto aberto da comunicação de Bob a Alice. Ao mesmo tempo, Trudy pode encaminhar os dados de Bob para Alice (após recodificar os dados usando a verdadeira chave pública de Alice).

Bob está feliz por enviar dados cifrados e Alice está feliz por recebê-los utilizando sua própria chave pública; ambos não sabem da presença de Trudy. Caso Bob e Alice se encontrem depois e discutam sobre sua interação, Alice terá recebido exatamente o que Bob enviou, então nada será detectado como suspeito. Esse é um exemplo do que denominamos ataque do homem do meio (mais apropriadamente nesta seção, um “ataque da mulher do meio”).

8.4 Protegendo o e-mail

Nas seções anteriores analisamos as questões fundamentais em segurança de redes, incluindo a criptografia de chave simétrica e de chave pública, autenticação do ponto final, distribuição de chave, integridade da mensagem e assinaturas digitais. Agora vamos analisar como essas ferramentas estão sendo usadas para oferecer segurança na Internet.

Curiosamente, é possível oferecer serviços de segurança em cada uma das quatro principais camadas da pilha de protocolos da Internet. Quando a segurança é oferecida para um protocolo específico da camada de aplicação, a aplicação que usa o protocolo utilizará um ou mais serviços de segurança, como sigilo, autenticação ou integridade. Quando a segurança é oferecida para um protocolo da camada de transporte, todas as aplicações que usam esse protocolo aproveitam os serviços de segurança do protocolo de transporte. Quando a segurança é oferecida na camada de rede em base hospedeiro para hospedeiro, todos os segmentos da camada de transporte (e, por-



tanto, todos os dados da camada de aplicação) aproveitam os serviços de segurança da camada de rede. Quando a segurança é oferecida em um enlace, então os dados em todos os quadros que percorrem o enlace recebem os serviços de segurança do enlace.

Na Seção 8.4 a 8.7 verificamos como as ferramentas de segurança estão sendo usadas nas camadas de enlace, rede, transporte e aplicação. Obedecendo à estrutura geral deste livro, começamos no topo da pilha de protocolo e discutimos segurança na camada de aplicação. Nossa abordagem é usar uma aplicação específica, e-mail, como um estudo de caso para a segurança da camada de aplicação. Então, descemos a pilha de protocolo. Examinaremos o protocolo SSL (que provê segurança na camada de transporte), o IPsec (que provê segurança na camada de rede) e a segurança do protocolo LAN IEEE 802.11 sem fio.

Você deve estar se perguntando por que a funcionalidade da segurança está sendo fornecida em mais de uma camada na Internet. Já não bastaria prover essa funcionalidade na camada de rede? Há duas respostas para essa pergunta. Primeiro, embora a segurança na camada de rede possa oferecer “cobertura total” cifrando todos os dados nos datagramas (ou seja, todos os segmentos da camada de transporte) e autenticando todos os endereços IP destinatários, ela não pode prover segurança no nível do usuário. Por exemplo, um site de comércio não pode confiar na segurança da camada IP para autenticar um cliente que vai comprar mercadorias nesse site. Assim, existe a necessidade de uma funcionalidade da segurança em camadas superiores bem como cobertura total em canais inferiores. Segundo, geralmente é mais fácil implementar serviços da Internet, incluindo serviços de segurança nas camadas superiores da pilha de protocolo. Enquanto aguardamos a ampla implementação da segurança na camada de rede, o que ainda levará muitos anos, muitos criadores de aplicação “já fazem isso” e introduzem a funcionalidade da segurança em suas aplicações favoritas. Um exemplo clássico é o *Pretty Good Privacy* (PGP), que oferece e-mail seguro (discutido mais adiante nesta seção). Necessitando de apenas um código de aplicação do cliente e do servidor, o PGP foi uma das primeiras tecnologias de segurança a ser amplamente utilizada na Internet.

8.4.1 E-mail seguro

Agora usamos os princípios de criptografia das Seções 8.2 a 8.3 para criar um sistema de e-mail seguro. Criamos esse projeto de alto nível de maneira incremental, introduzindo, a cada etapa, novos serviços de segurança. Em nosso projeto de um sistema de e-mail seguro, vamos manter em mente o exemplo picante apresentado na Seção 8.1 — o caso de amor entre Alice e Bob. Imagine que Alice quer enviar uma mensagem de e-mail para Bob e Trudy quer bisbilhotar.

Antes de avançar e projetar um sistema de e-mail seguro para Alice e Bob, devemos considerar quais características de segurança seriam as mais desejáveis para eles. A primeira característica, e a mais importante, é a confidencialidade. Como foi discutido na Seção 8.1, nem Alice nem Bob querem que Trudy leia a mensagem de e-mail de Alice. A segunda característica que Alice e Bob provavelmente gostariam de ver no sistema de e-mail seguro é a autenticação do remetente. Em particular, quando Bob receber a seguinte mensagem de Alice: “Eu não o amo mais. Nunca mais quero vê-lo. Da anteriormente sua, Alice”, ele naturalmente gostaria de ter certeza de que a mensagem veio de Alice, e não de Trudy. Outra característica de segurança de que os dois amantes gostariam de dispor é a integridade de mensagem, isto é, a certeza de que a mensagem que Alice enviar não será modificada no trajeto até Bob. Por fim, o sistema de e-mail deve fornecer autenticação do receptor, isto é, Alice quer ter certeza de que ela de fato está enviando a mensagem para Bob, e não para outra pessoa (por exemplo, Trudy) que possa estar se passando por Bob.

Portanto, vamos começar abordando a preocupação mais premente de Alice e Bob, a confidencialidade. A maneira mais direta de conseguir confidencialidade é Alice criptografar a mensagem com tecnologia de chaves simétricas (como DES ou AES) e Bob decriptar a mensagem ao recebê-la. Como discutido na Seção 8.2, se a chave simétrica for suficientemente longa e se somente Alice e Bob possuírem a chave, então será extremamente difícil que alguém (incluindo Trudy) leia a mensagem. Embora essa seja uma abordagem direta, ela apresenta a dificuldade fundamental que discutimos na Seção 8.2 — é difícil distribuir uma chave simétrica de modo que apenas Bob e Alice tenham cópias dela. Portanto, é natural que consideremos uma abordagem alternativa — a

criptografia de chaves públicas (usando, por exemplo, RSA). Na abordagem de chaves públicas, Bob disponibiliza publicamente sua chave pública (por exemplo, em um servidor de chaves públicas ou em sua página Web pessoal) e Alice criptografa sua mensagem com a chave pública de Bob, e envia a mensagem criptografada para o endereço de e-mail de Bob. (A mensagem criptografada é encapsulada com cabeçalhos MIME e enviada por SMTP comum, como discutimos na Seção 2.4.) Quando Bob recebe a mensagem, ele simplesmente a decodifica com sua chave privada. Admitindo que Alice tenha certeza de que aquela chave pública é a de Bob (e que é suficientemente longa), essa abordagem é um meio excelente de fornecer a confidencialidade desejada. Um problema, contudo, é que a criptografia de chaves públicas é relativamente ineficiente, sobretudo para mensagens longas.

Para superar o problema da eficiência, vamos fazer uso de uma chave de sessão (discutida na Seção 8.2.2). Em particular, Alice (1) escolhe uma chave simétrica, K_S , aleatoriamente, (2) criptografa sua mensagem m com a chave simétrica K_S , (3) criptografa a chave simétrica com a chave pública de Bob, K_B^+ , (4) concatena a mensagem criptografada e a chave simétrica criptografada de modo que formem um ‘pacote’ e (5) envia o pacote ao endereço de e-mail de Bob. Os passos estão ilustrados na Figura 8.20. (Nessa figura e nas subsequentes, o sinal ‘+’ dentro de um círculo representa formar a concatenação e o sinal ‘-’ dentro de um círculo, desfazer a concatenação.) Quando Bob receber o pacote, ele (1) usará sua chave privada K_B^- , para obter a chave simétrica K_S , e (2) utilizará a chave simétrica K_S para decodificar a mensagem m .

Agora que projetamos um sistema de e-mail seguro que fornece confidencialidade, vamos desenvolver um outro sistema que forneça autenticação do remetente e também integridade de mensagem. Vamos supor, no momento, que Alice e Bob não estejam mais preocupados com confidencialidade (querem compartilhar seus sentimentos com todos!) e que estejam somente preocupados com a autenticação do remetente e com a integridade da mensagem. Para realizar essa tarefa, usaremos assinaturas digitais e resumos de mensagem, como descrito na Seção 8.3. Especificamente, Alice (1) aplica uma função de hash H (por exemplo, MD5) à sua mensagem m , para obter um resumo de mensagem, (2) assina o resultado da função de hash com sua chave privada K_A^- para criar uma assinatura digital, (3) concatena a mensagem original (não criptografada) com a assinatura para criar um pacote e (4) envia o pacote ao endereço de e-mail de Bob. Quando Bob recebe o pacote, ele (1) aplica a chave pública de Alice, K_A^+ , ao resumo de mensagem assinado e (2) compara o resultado dessa operação com o próprio hash H da mensagem. As etapas são ilustradas na Figura 8.21. Como discutimos na Seção 8.3, se os dois resultados forem iguais, Bob poderá ter razoável certeza de que a mensagem veio de Alice e não foi alterada.

Vamos considerar agora o projeto de um sistema de e-mail que forneça confidencialidade, autenticação de remetente e integridade de mensagem. Isso pode ser feito pela combinação dos procedimentos das figuras 8.20 e 8.21. Em primeiro lugar, Alice cria um pacote preliminar, exatamente como mostra a Figura 8.21, constituído de sua mensagem original junto com um hash da mensagem assinado digitalmente. Em seguida, ela trata esse pacote preliminar como uma mensagem em si e envia essa nova mensagem seguindo as etapas do remetente mostradas na Figura 8.20, criando um novo pacote, que é enviado a Bob. As etapas seguidas são mostradas na Figura 8.22. Quando Bob recebe o pacote, ele aplica primeiramente seu lado da Figura 8.20 e depois seu lado da Figura 8.21.

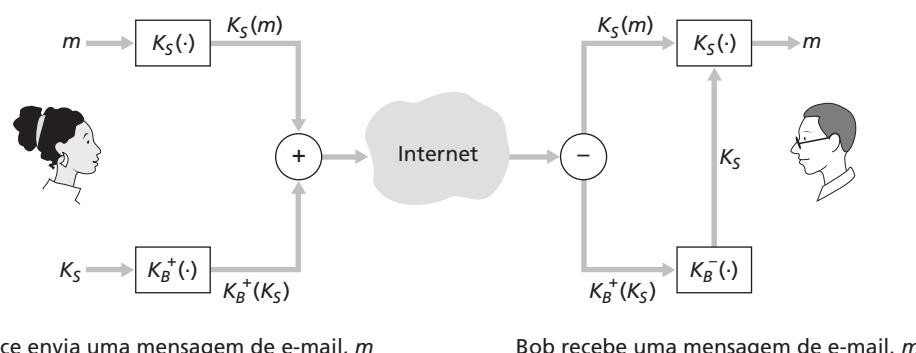


Figura 8.20 Alice usou uma chave de sessão simétrica, K_S , para enviar um e-mail secreto para Bob

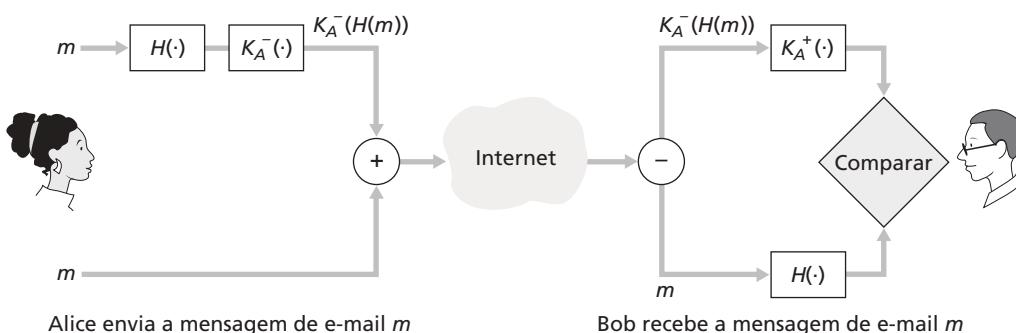


Figura 8.21 Utilização de funções de hash e assinaturas digitais para fornecer autenticação de remetente e integridade de mensagem

É preciso ficar claro que esse projeto atinge o objetivo de fornecer confidencialidade, autenticação de remetente e integridade de mensagem. Note que nesse esquema Alice utiliza criptografia de chaves públicas duas vezes: uma vez com sua chave privada e uma vez com a chave pública de Bob. De maneira semelhante, Bob também aplica a criptografia de chaves públicas duas vezes — uma vez com sua chave privada e uma vez com a chave pública de Alice.

O projeto de e-mail seguro ilustrado na Figura 8.22 provavelmente fornece segurança satisfatória para os usuários de e-mail na maioria das ocasiões. Mas ainda resta uma questão importante a ser abordada. O projeto da Figura 8.22 requer que Alice obtenha a chave pública de Bob e que Bob obtenha a chave pública de Alice. A distribuição dessas chaves é um problema nada trivial. Por exemplo, Trudy poderia se disfarçar de Bob dar a Alice sua própria chave pública, dizendo que é a chave pública de Bob. Como aprendemos na Seção 8.3, uma abordagem popular para distribuir chaves públicas com segurança é certificar as chaves públicas usando uma autoridade certificadora.


História

Phil Zimmermann e o PGP

Philip R. Zimmermann é o criador do Pretty Good Privacy (PGP). Por causa disso, ele foi alvo de uma investigação criminal que durou três anos, porque o governo norte-americano sustentava que as restrições sobre a exportação de software de criptografia tinham sido violadas quando o PGP se espalhou por todo o mundo após sua publicação como software de uso livre em 1991. Após ter sido liberado como software compartilhado, alguém o colocou na Internet e estrangeiros o descarregaram. Nos Estados Unidos, os programas de criptografia são classificados como artefatos militares por lei federal e não podem ser exportados.

A despeito da falta de financiamento, da inexistência de representantes legais, da falta de uma empresa para lhe dar apoio e das intervenções governamentais, mesmo assim o PGP se tornou o software de criptografia para e-mail mais usado no mundo. O extraordinário é que o governo dos Estados Unidos pode ter contribuído inadvertidamente para a disseminação do PGP devido ao caso Zimmermann.

O governo norte-americano arquivou o caso no início de 1996. O anúncio foi festejado pelos ativistas da Internet. O caso Zimmermann tinha se transformado na história de um inocente que lutava por seus direitos contra os desmandos de um governo poderoso. A desistência do governo foi uma notícia bem-vinda, em parte por causa da campanha em favor da censura na Internet desencadeada pelo Congresso e em parte por causa dos esforços feitos pelo FBI para conseguir maior amplitude para a espionagem governamental.

Após o governo ter arquivado o caso, Zimmermann fundou a PGP Inc., que foi adquirida pela Network Associates em dezembro de 1997. Zimmermann é agora membro do conselho da Network Associates, bem como consultor independente para assuntos de criptografia.

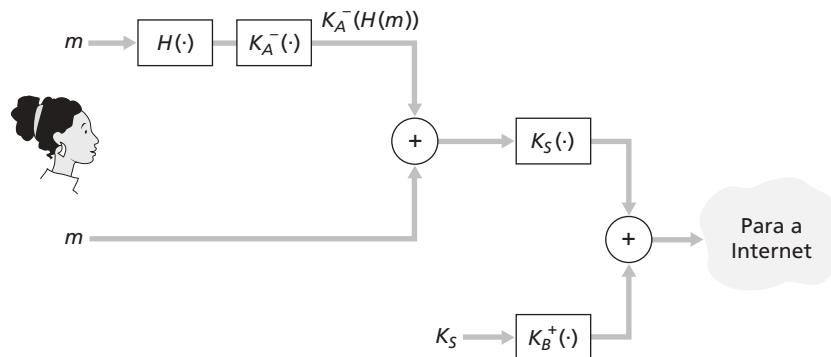


Figura 8.22 Alice usa criptografia de chaves simétricas, criptografia de chaves públicas, uma função de hash e uma assinatura digital para fornecer confidencialidade, autenticação de remetente e integridade de mensagem

8.4.2 PGP

Projetado originalmente por Phil Zimmermann em 1991, o PGP (**Pretty Good Privacy** — privacidade razoável) é um esquema de criptografia para e-mail que se tornou um padrão de fato. O site do PGP é acessado mais de um milhão de vezes por mês por usuários de 166 países [PGPI, 2007]. Versões do PGP estão disponíveis em domínio público; por exemplo, você pode encontrar o software PGP para sua plataforma favorita, bem como grande quantidade de material de leitura interessante, na home page internacional do PGP [PGPI, 2007]. (Um ensaio particularmente interessante, escrito pelo autor do PGP, é encontrado em [Zimmermann, 2007].) O projeto do PGP é, em essência, idêntico ao projeto apresentado na Figura 8.22. Dependendo da versão, o software do PGP usa MD5 ou SHA para processar o resumo de mensagem; CAST, DES triplo ou IDEA para criptografar chaves simétricas, e RSA para criptografar chaves públicas.

Quando o PGP é instalado, o software cria um par de chaves públicas para o usuário. A chave pública pode então ser colocada no site do usuário ou em um servidor de chaves públicas. A chave privada é protegida pelo uso de uma senha. A senha tem de ser informada todas as vezes que o usuário acessar a chave privada. O PGP oferece ao usuário a opção de assinar digitalmente a mensagem, criptografar a mensagem ou, ainda, ambas as opções: assinar digitalmente e criptografar a mensagem. A Figura 8.23 mostra uma mensagem PGP assinada. Essa mensagem aparece após o cabeçalho MIME. Os dados codificados da mensagem correspondem a $K_A^-(H(m))$, isto é, ao resumo de mensagem assinado digitalmente. Como discutimos antes, para que Bob verifique a integridade da mensagem, ele precisa ter acesso à chave pública de Alice.

A Figura 8.24 mostra uma mensagem PGP secreta. Essa mensagem também aparece após o cabeçalho MIME. É claro que a mensagem em texto aberto não está incluída na mensagem de e-mail secreta. Quando um remetente

```

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Bob:
Can I see you tonight?
Passionately yours, Alice
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset: noconv
yhHJRhhGJGhgg/12EpJ+1o8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
  
```

Figura 8.23 Uma mensagem PGP assinada

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX681iKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=1KhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmK1ok8iy6gTh1p
-----END PGP MESSAGE
```

Figura 8.24 Uma mensagem PGP secreta

(como Alice) quer não apenas a confidencialidade, mas também a integridade, o PGP contém uma mensagem como a da Figura 8.24 dentro da mensagem da Figura 8.23.

O PGP também fornece um mecanismo para certificação de chaves públicas, mas esse mecanismo é bem diferente daquele da CA mais convencional. As chaves públicas do PGP são certificadas por uma *rede de confiança*. A própria Alice pode certificar qualquer par chave/usuário quando ela achar que esse par realmente está correto. Além disso, o PGP permite que Alice declare que ela confia em outro usuário para atestar a autenticidade de mais chaves. Alguns usuários do PGP assinam reciprocamente suas chaves montando grupos de assinatura de chaves. Os usuários se reúnem fisicamente, trocam disquetes contendo as chaves públicas e certificam suas chaves reciprocamente, assinando-as com suas chaves privadas.

8.5 Protegendo conexões TCP: SSL

Na seção anterior, vimos como as técnicas criptográficas podem prover sigilo, integridade dos dados e autenticação do ponto final a aplicações específicas, ou seja, e-mail. Nesta seção, desceremos uma camada na pilha de protocolo e examinar como a criptografia pode aprimorar o TCP com os serviços de segurança, incluindo sigilo, integridade dos dados e autenticação do ponto final. Essa versão aprimorada do TCP é denominada Camada Segura de Sockets[Secure Sockets Layer] (SSL). Uma versão levemente modificada da versão 3 do SSL, denominada Segurança na Camada de Transporte [Transport Layer Security] (TLS), foi padronizada pelo IETF [RFC 2246].

O SSL foi originalmente projetado pelo Netscape, mas as ideias básicas por trás da proteção do TCP antecederam o trabalho do Netscape (por exemplo, consulte Woo [Woo, 1994]). Desde sua concepção, o SSL obteve uma ampla implementação. Ele é suportado por todos os browsers Web e servidores Web, e usado basicamente por todos os sites populares (incluindo Amazon, eBay, Yahoo!, MSN etc.). Dezenas de bilhões de dólares são gastos com o SSL a cada ano. Na verdade, se você já comprou qualquer coisa pela Internet com seu cartão de crédito, a comunicação entre seu browser e servidor para essa compra foi quase certamente por meio do SSL. (Você pode identificar que o SSL está sendo usado por seu browser quando a URL se iniciar com https: em vez de http:.)

Para entender a necessidade do SSL, vamos examinar um cenário de comércio pela Internet típico. Bob está navegando na Web e acessa o site Alice Incorporated, que está vendendo perfume. O site Alice Incorporated exibe um formulário no qual Bob deve inserir o tipo de perfume e quantidade desejados, seu endereço e o número de seu cartão de crédito. Bob insere essas informações, clica em Enviar, e espera pelo recebimento (via correio postal comum) de seus perfumes; ele também espera pelo recebimento de uma cobrança pelo seu pedido na próxima fatura do cartão de crédito. Até o momento, tudo bem, mas se nenhuma medida de segurança for tomada, Bob poderia esperar por surpresas.

- Se nenhum sigilo (criptografia) for utilizado, um invasor poderia interceptar o pedido de Bob e receber suas informações sobre o cartão. O invasor poderia, então, fazer compras à custa de Bob.
- Se nenhuma integridade de dados for utilizada, um invasor poderia modificar o pedido de Bob, fazendo-o comprar dez vezes mais frascos de perfumes que o desejado.
- Finalmente, se nenhuma autenticação do servidor for utilizada, um servidor poderia exibir o famoso logotipo de Alice Incorporated, quando na verdade o site é mantido por Trudy, que está se passando por Alice Incorporated. Após receber o pedido de Bob, Trudy poderia ficar com o dinheiro dele e sumir. Ou Trudy poderia realizar um roubo de identidade obtendo o nome, endereço e número do cartão de crédito de Bob.

O SSL aborda essas questões aprimorando o TCP com sigilo, integridade dos dados, autenticação do servidor e autenticação do cliente.

Muitas vezes, o SSL é usado para oferecer segurança em transações que ocorrem através do HTTP. Entretanto, como o SSL protege o TCP, ele pode ser empregado por qualquer aplicação que execute o TCP. O SSL provê uma Interface de Programação de Aplicação (API) com sockets, semelhante ao API do TCP. Quando uma aplicação quer empregar o SSL, ela inclui classes/bibliotecas SSL. Como mostrado na Figura 8.25, embora o SSL resida tecnicamente na camada de aplicação, da perspectiva do desenvolvedor, ele é um protocolo de transporte que provê serviços do TCP aprimorados com serviços de segurança.

8.5.1 Uma visão abrangente

Começamos descrevendo uma versão simplificada do SSL, que nos permitirá obter uma visão abrangente de *por que* e *como* funciona o SSL. Iremos nos referir a essa versão simplificada do SSL como “quase-SSL”. Após descrever o quase-SSL, na próxima subseção, descreveremos o SSL de verdade, preenchendo as lacunas. O quase-SSL (e o SSL) possui três fases: *apresentação (handshake)*, *derivação de chave* e *transferência de dados*. Agora descreveremos essas três fases para uma sessão de comunicação entre um cliente (Bob) e um servidor (Alice), o qual possui um par de chave pública/privada e um certificado que associa sua identidade à chave pública.

Apresentação (*Handshake*)

Durante a fase de apresentação, Bob precisa (a) estabelecer uma conexão TCP com Alice, (b) verificar se Alice é realmente Alice, e (c) enviar para ela uma chave secreta mestre, que será utilizada por Alice e Bob para criar todas as chaves simétricas de que eles precisam para a sessão SSL. Essas três etapas estão ilustradas na Figura 8.26. Observe que uma vez a conexão TCP é estabelecida, Bob envia a Alice uma mensagem “hello”. Alice, então, responde com seu certificado, que contém sua chave pública. Conforme discutido na Seção 8.3, como o certificado foi assinado por uma CA, Bob sabe, com certeza, que a chave pública no certificado pertence a Alice. Bob então cria um Segredo Mestre (MS) (o qual somente será usado para esta sessão SSL), codifica o MS com a chave pública de Alice para criar o Segredo Mestre Cifrado (EMS), enviando-o para Alice, que o decodifica com sua chave privada para obter o MS. Após esta fase, Bob e Alice (e mais ninguém) sabem o segredo mestre para esta sessão SSL.

Derivação de chave

A princípio, o MS, agora compartilhado por Bob e Alice, poderia ser usado como a chave de sessão simétrica para toda a verificação subsequente de criptografia e integridade dos dados. Entretanto, geralmente é considerado mais seguro para Alice e Bob usarem, individualmente, chaves criptográficas diferentes, bem como chaves diferentes para criptografia e verificação da integridade. Assim, Alice e Bob usam o MS para criar quatro chaves:

E_B – chave de criptografia de sessão para dados enviados de Bob para Alice

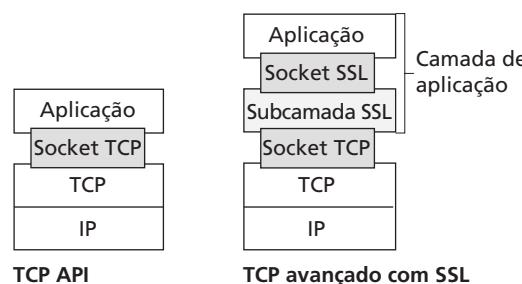


Figura 8.25 Embora o SSL resida tecnicamente na camada de aplicação, da perspectiva do desenvolvedor, ele é um protocolo da camada de transporte

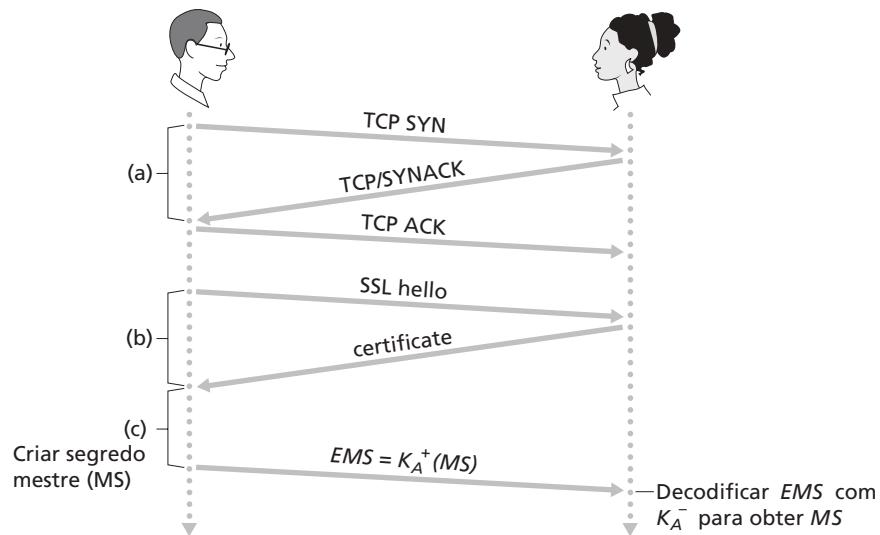


Figura 8.26 A apresentação quase-SSL, iniciando com uma conexão TCP

M_B = chave MAC de sessão para dados enviados de Bob para Alice

E_A = chave de criptografia de sessão para dados enviados de Alice para Bob

M_A = chave MAC de sessão para dados enviados de Alice para Bob

Alice e Bob podem criar quatro chaves a partir do MS. Isso poderia ser feito simplesmente dividindo o MS em quatro chaves. (Mas na verdade, o SSL é um pouco mais complicado, como veremos.) Ao final da fase de derivação de chave, Alice e Bob têm todas as quatro chaves. As duas chaves de criptografia serão usadas para cifrar dados; as duas chaves MAC serão usadas para verificar a integridade dos dados.

Transferência de dados

Agora que Alice e Bob compartilham as mesmas quatro chaves de sessão (E_B , M_B , E_A e M_A), eles podem começar a enviar dados protegidos um ao outro por meio da conexão TCP. Como o TCP é um protocolo de fluxo de bytes, uma abordagem natural seria o SSL cifrar dados da aplicação enquanto são enviados e, então, transmitir os dados cifrados para o TCP. Mas se fizéssemos isso, onde colocaríamos o MAC para a verificação da integridade? Certamente não queremos esperar até o fim da sessão TCP para verificar a integridade de todos os dados de Bob que foram enviados por toda a sessão! Para abordar essa questão, o SSL divide o fluxo de dados em *registros*, anexa um MAC a cada registro para a verificação da integridade, e cifra o registro+MAC. Para criar o MAC, Bob insere os dados de registro junto com a chave M_B em uma função de hash, conforme discutido na Seção 8.3. Para cifrar o pacote registro+MAC, Bob usa sua chave de criptografia de sessão E_B . Esse pacote cifrado é então transmitido ao TCP para o transporte através da Internet.

Embora essa abordagem vá longe, o fornecimento de integridade dos dados para um fluxo inteiro de mensagem ainda não é à prova de falhas. Em particular, suponha que Trudy seja uma mulher no meio e possua a habilidade de inserir, deletar e substituir segmentos no fluxo dos segmentos TCP enviados entre Alice e Bob. Trudy, por exemplo, poderia capturar dois segmentos enviados por Bob, inverter sua ordem, ajustar os números de sequência TCP (que não estão cifrados) e enviar dois segmentos na ordem inversa para Alice. Admitindo que cada segmento TCP encapsula exatamente um registro, vamos verificar como Alice processaria esses segmentos,

1. O TCP que está sendo executado em Alice pensaria que está tudo bem e passaria os dois registros para a subcamada SSL.
2. O SSL em Alice decriptografaria os dois registros.

3. O SSL em Alice usaria o MAC em cada registro para verificar a integridade dos dados dos dois registros.
4. O SSL passaria, então, os fluxos de bytes decifrados dos dois registros à camada de aplicação; mas o fluxo de bytes completo recebido por Alice não estaria na ordem correta devido à inversão dos registros!

Recomendamos que você analise cenários diferentes para quando Trudy remove segmentos e quando Trudy repete segmentos.

A solução para esse problema, como você deve ter imaginado, é usar números de sequência. O SSL os utiliza da seguinte maneira. Bob mantém um contador de números de sequência, que se inicia no zero e vai aumentando para cada registro SSL que ele envia. Bob, na verdade, não inclui um número de sequência no próprio registro, mas no cálculo do MAC. Desse modo, o MAC é agora um hash dos dados mais uma chave MAC M_B mais o número de sequência atual. Alice rastreia os números de sequência de Bob, permitindo que ela verifique a integridade dos dados de um registro incluindo o número de sequência apropriado no cálculo do MAC. O uso de números de sequência SSL impede que Trudy realize um ataque da mulher do meio, como reordenar ou repetir os segmentos. (Por quê?)

Registro SSL

O registro SSL (assim como o registro quase-SSL) é ilustrado na Figura 8.27. O registro consiste em um campo de tipo, um campo de versão, um campo de comprimento, um campo de dados e um campo MAC. Observe que os primeiros três campos não estão cifrados. O campo de tipo indica se o registro é uma mensagem de apresentação ou uma mensagem que contém dados da aplicação. É também usado para encerrar a conexão SSL, como discutido abaixo. Na extremidade receptora, o SSL usa o campo de comprimento para extrair os registros SSL do fluxo de byte TCP de entrada. O campo de versão não precisa de explicações.

8.5.2 Uma visão mais completa

A subseção anterior abordou o protocolo quase-SSL: serviu para nos dar uma compreensão básica de por que e como funciona o SSL. Agora que temos essa compreensão básica sobre o SSL, podemos nos aprofundar mais e examinar os princípios básicos do verdadeiro protocolo SSL. Além da leitura desta descrição sobre o SSL, recomendamos que você complete o Wireshark SSL lab, disponível no companion Web site deste livro.

Apresentação SSL

O SSL não exige que Alice e Bob usem um algoritmo específico de chave simétrica, um algoritmo específico de chave pública ou um MAC específico. Em vez disso, o SSL permite que Alice e Bob combinem os algoritmos criptográficos no início da sessão SSL, durante a fase de apresentação. Ademais, durante a fase de apresentação, Alice e Bob enviam nonces um ao outro, que são usados na criação de chaves de sessão (E_B , M_B , E_A e M_A). As etapas da verdadeira apresentação SSL são:

1. O cliente envia uma lista de algoritmos criptográficos que ele suporta, junto com um nonce do cliente.
2. A partir da lista, o servidor escolhe um algoritmo simétrico (por exemplo, AES), um algoritmo de chave pública (por exemplo, RSA com um comprimento de chave específico) e um algoritmo MAC. Ele devolve ao cliente suas escolhas, bem como um certificado e um nonce do servidor.
3. O cliente verifica o certificado, extrai a chave pública do servidor, gera um Segredo Pré-Mestre (PMS), cifra o PMS com a chave pública do servidor e envia o PMS cifrado ao servidor.

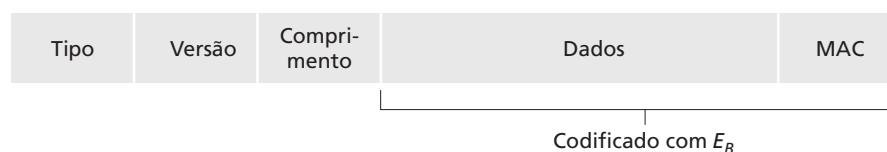


Figura 8.27 Formato de registro para o SSL

4. Utilizando a mesma função de derivação de chave (conforme especificado pelo padrão SSL), o cliente e o servidor computam independentemente o Segredo Mestre (MS) do PMS e dos nonces. O PMS é então dividido para gerar as duas chaves de criptografia e duas chaves MAC. Além disso, quando a cifra simétrica selecionada emprega o CBC (como 3DES ou AES), os dois Vetores de Inicialização (IVs) — um para cada lado da conexão — são também obtidos do PMS. De agora em diante, todas as mensagens enviadas entre o cliente e o servidor são cifradas e autenticadas (com o MAC).
5. O cliente envia um MAC de todas as mensagens de apresentação.
6. O servidor envia um MAC de todas as mensagens de apresentação.

As duas últimas etapas protegem a apresentação da adulteração. Para entender isso, observe que no passo 1, o cliente normalmente oferece uma lista de algoritmos — alguns fortes e outros fracos. Essa lista é enviada em texto aberto, visto que os algoritmos de criptografia e chaves ainda não foram consentidos. Trudy, como uma mulher no meio, poderia deletar os algoritmos mais fortes da lista, obrigando o cliente a selecionar um algoritmo fraco. Para evitar o ataque de adulteração, na etapa 5 o cliente envia um MAC da concatenação de todas as mensagens de apresentação que ele enhou e recebeu. O servidor pode comparar esse MAC com o MAC das mensagens de apresentação que ele enhou e recebeu. Se houver uma inconsistência, o servidor pode finalizar a conexão. De maneira semelhante, o servidor envia um MAC das mensagens de apresentação que encontrou, permitindo que o cliente verifique a presença de inconsistências.

Você pode estar se perguntando, por que existem nonces nas etapas 1 e 2? Os números de sequência são suficientes para prevenir o ataque de repetição de segmento? A resposta é sim, mas eles não previnem sozinhos “o ataque de repetição de segmento”. Considere o seguinte ataque de repetição de segmento. Suponha que Trudy analise todas as mensagens entre Alice e Bob. No dia seguinte, Trudy se passa por Bob e envia a Alice exatamente a mesma sequência de mensagens que Bob enhou a Alice no dia anterior. Se Alice não usar os nonces, ela responderá com exatamente a mesma sequência de mensagem que enhou no dia anterior. Alice não suspeitará de nada estranho, pois cada mensagem que ela recebe passará pela verificação de integridade. Se Alice for um servidor de comércio eletrônico, ela pensará que Bob está efetuando um segundo pedido (para exatamente a mesma coisa). Por outro lado, ao incluir um nonce no protocolo, Alice enviará nonces diferentes para cada sessão TCP, fazendo com que as chaves de criptografia sejam diferentes nos dois dias. Portanto, quando Alice recebe os registros SSL repetidos de Trudy, eles falharão na verificação de integridade, e a transação do site de comércio eletrônico falso não acontecerá. Em resumo, no SSL, os nonces são usados para proteger o “ataque de repetição de conexão”, e os números de sequência são usados para defender a repetição de pacotes individuais durante uma sessão em andamento.

Encerramento de conexão

Em algum momento, Bob ou Alice desejarião finalizar a sessão SSL. Um método seria deixar Bob finalizar a sessão SSL simplesmente encerrando a conexão TCP subjacente — ou seja, enviando um segmento TCP FIN para Alice. Mas esse plano ingênuo prepara o terreno para o ataque por truncamento através do qual Trudy, mais uma vez, se localiza no meio de uma sessão SSL em andamento e finaliza a sessão antes da hora com um TCP FIN. Se esse fosse o objetivo de Trudy, Alice acharia que ela recebeu todos os dados de Bob quando, na verdade, ela somente recebeu uma parte deles. A solução para esse problema é indicar no campo de tipo se o registro serve para encerrar a sessão SSL. (Embora o tipo SSL seja enviado em aberto, ele é autenticado no receptor usando o MAC do registro). Ao incluir esse campo, se Alice fosse receber um TCP FIN antes de receber um registro SSL de encerramento, ela saberia que algo estranho estava acontecendo.

Isso conclui nossa introdução ao SSL. Vimos que ele usa muitos dos princípios da criptografia discutidos nas Seções 8.2 e 8.3. Os leitores que querem explorar o SSL mais profundamente podem consultar o livro de Rescorla sobre o SSL [Rescorla, 2001].

8.6 Segurança na camada de rede: IPsec e redes virtuais privadas

O protocolo IP de segurança, mais conhecido como IPsec, provê segurança na camada de rede. O IPsec protege os datagramas IP entre quaisquer entidades da camada de rede, incluindo hospedeiros e roteadores. Como descreveremos em breve, muitas instituições (corporações, órgãos do governo, organizações sem fins lucrativos etc.) usam o IPsec para criar redes virtuais privadas (VPNs) que executadas por meio da Internet pública.

Antes de falar sobre o IPsec, vamos voltar e considerar o que significa prover sigilo na camada de rede. Com o sigilo da camada de rede entre um par de entidades da rede (por exemplo, entre dois roteadores, entre dois hospedeiros, ou entre um roteador e um hospedeiro), a entidade remetente cifra as cargas úteis de todos os datagramas que envia à entidade destinatária. A carga útil cifrada poderia ser um segmento TCP, um segmento UDP e uma mensagem ICMP etc. Se esse serviço da camada de rede estivesse em funcionamento, todos os dados de uma entidade a outra — incluindo e-mail, páginas Web, mensagens de apresentação TCP e mensagens de gerenciamento (como ICMP e SNMP) — estariam ocultos de qualquer terceira parte que possa estar analisando a rede. Por esta razão, a segurança na camada de rede é conhecida por prover “cobertura total”.

Além do sigilo, um protocolo de segurança da camada de rede poderia potencialmente prover outros serviços de segurança. Por exemplo, ele poderia fornecer autenticação da fonte de modo que a entidade destinatária possa verificar a fonte do datagrama protegido. Um protocolo de segurança da camada de rede poderia oferecer integridade dos dados, de modo que a entidade destinatária poderia verificar qualquer adulteração do datagrama que pudesse ter ocorrido enquanto o datagrama estava em trânsito. Um serviço de segurança da camada de rede também poderia oferecer a prevenção do ataque de repetição, querendo dizer que Bob poderia detectar quaisquer datagramas duplicados que um atacante possa inserir. Veremos em breve que o IPsec provê mecanismos para todos esses serviços de segurança, ou seja, para sigilo, autenticação da fonte, integridade dos dados e prevenção do ataque de repetição.

8.6.1 IPsec e redes virtuais privadas (VPNs)

Uma instituição que se estende por diversas regiões geográficas muitas vezes deseja ter sua própria rede IP, para que seus hospedeiros e servidores possam enviar dados um ao outro de uma maneira segura e sigilosa. Para alcançar essa meta, a instituição poderia, na verdade, empregar uma rede física independente — incluindo roteadores, enlaces e uma infraestrutura DNS — que é completamente distinta da Internet pública. Essa rede disjunta, reservada a uma instituição particular, é chamada de rede privada. Como é de se esperar, uma rede privada pode ser muito cara, já que a instituição precisa comprar, instalar e manter sua própria infraestrutura de rede física.

Em vez de implementar e manter uma rede privada, hoje muitas instituições criam VPNs a partir da Internet pública existente. Com uma VPN, o tráfego interdepartamental é enviado por meio da Internet pública e não de uma rede fisicamente independente. Mas para prover sigilo, o tráfego interdepartamental é criptografado antes de entrar na Internet pública. Um exemplo simples de uma VPN é mostrado na Figura 8.28. Aqui, a instituição consiste em uma matriz, uma filial e vendedores viajantes, que normalmente acessam a Internet do seu quarto de hotel. (A figura mostra somente um vendedor.) Nesta VPN, quando dois hospedeiros dentro da matriz enviam datagramas IP um ao outro ou quando dois hospedeiros dentro de uma filial querem se comunicar, eles usam o bom e velho IPv4 (ou seja, sem os serviços IPsec.). Entretanto, quando dois dos hospedeiros da instituição se comunicam através de um caminho que cruza a Internet pública, o tráfego é codificado antes de entrar na Internet.

Para entender como a VPN funciona, vamos examinar um exemplo simples no contexto da Figura 8.28. Quando um hospedeiro na matriz envia um datagrama IP a um vendedor no hotel, o roteador de borda na matriz converte o datagrama IPv4 em um datagrama IPsec e o encaminha à Internet. Esse datagrama IPsec, na verdade, possui um cabeçalho IPv4 tradicional, de modo que os roteadores na Internet pública processam o datagrama como se ele fosse um datagrama IPv4 comum — para eles, o datagrama é perfeitamente comum. Mas, conforme ilustrado na Figura 8.28, a carga útil do datagrama IPsec inclui um cabeçalho IPsec, que é utilizado para o processamento do IPsec; além disso, a carga útil do datagrama IPsec está codificada. Quando o datagrama IPsec chegar ao laptop do vendedor, o OS no laptop decodifica a carga útil (e provê outros serviços de segurança, como

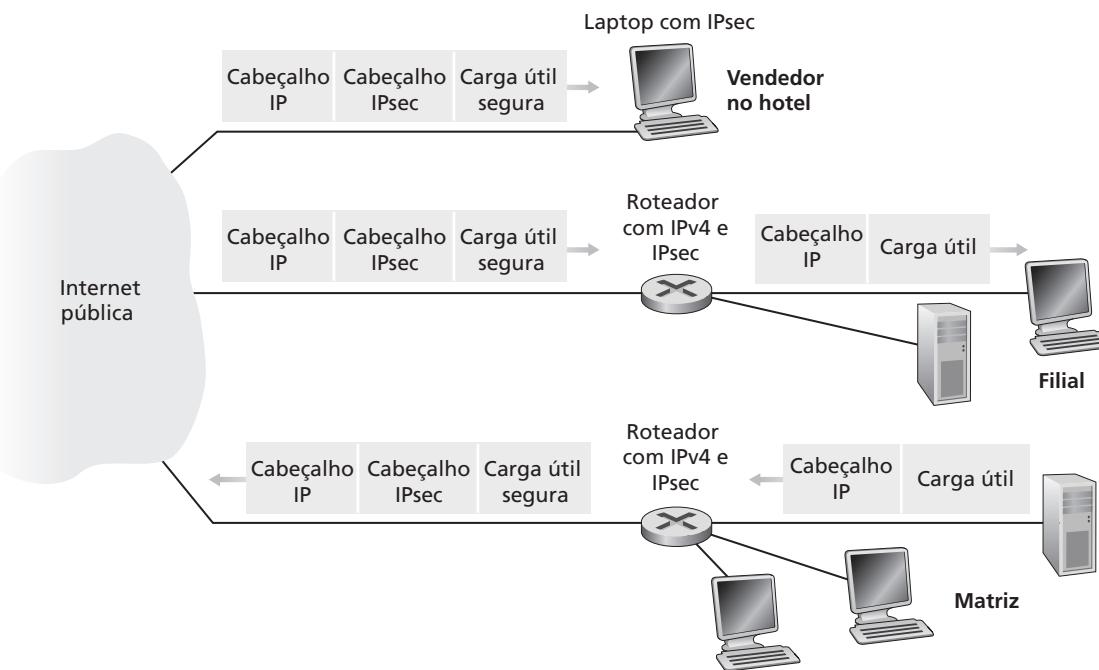


Figura 8.28 Rede Virtual Privada (VPN)

a verificação da integridade dos dados) e passa a carga útil não codificada para o protocolo da camada superior (por exemplo, para o TCP ou UDP).

Acabamos de dar uma visão geral de alto nível de como uma instituição pode implementar o IPsec para criar uma VPN. Para compreender a floresta por entre as árvores, deixamos de lado muitos detalhes importantes. Vamos agora dar uma olhada neles mais de perto.

8.6.2 Os protocolos AH e ESP

O IPsec é um material um tanto complexo — ele é definido em mais de uma dúzia de RFCs. Dois RFCs importantes são RFC 4301, o qual descreve a arquitetura de segurança IP geral, e RFC 2411, que fornece uma visão geral dos protocolos IPsec. Nossa meta neste livro, como de costume, não é simplesmente reprocessar os RFCs secos e complexos, mas fazer uma abordagem mais operacional e pedagógica para descrever os protocolos.

No conjunto dos protocolos IPsec, existem dois protocolos principais: o protocolo Cabeçalho de Autenticação (AH) e o protocolo Carga de Segurança de Encapsulamento (ESP). Quando uma entidade remetente IPsec (normalmente um hospedeiro ou um roteador) envia datagramas seguros a uma entidade destinatária (também um hospedeiro ou um roteador), ela utiliza o protocolo AH ou o protocolo ESP. O protocolo AH provê autenticação da fonte e integridade dos dados mas *não* provê sigilo. O protocolo ESP provê autenticação da fonte, integridade dos dados *e* sigilo. Em razão de o sigilo ser muitas vezes essencial às VPNs e outras aplicações IPsec, o protocolo ESP é muito mais utilizado do que o protocolo AH. Para desmistificar o IPsec e evitar suas complexidades, a partir de agora nos focaremos exclusivamente no protocolo ESP. Os leitores que desejam aprender também sobre o protocolo AH devem explorar os RFCs e outros recursos on-line.

8.6.3 Associações de segurança

Os datagramas IPsec são enviados entre pares de entidades da rede, tal como entre hospedeiros, entre dois roteadores, ou entre um hospedeiro e um roteador. Antes de enviar datagramas IPsec da entidade remetente à entidade destinatária, essas entidades criam uma conexão lógica da camada de rede, denominada associação de

segurança (SA). Uma SA é uma conexão lógica simplex; ou seja, ela é unidirecional do remetente ao destinatário. Se as duas entidades querem enviar datagramas seguros uma a outra, então duas SAs (ou seja, duas conexões lógicas) precisam ser estabelecidas, uma em cada direção. Por exemplo, considere mais uma vez uma VPN institucional na Figura 8.28. Essa instituição consiste em uma filial, uma matriz e, digamos, n vendedores viajantes. Por exemplo, vamos supor que haja tráfego IPsec bidirecional entre a matriz e a filial e tráfego IPsec bidirecional entre a matriz e os vendedores viajantes. Existem quantas SAs nessa VPN? Para responder essa questão, observe que há duas SAs entre o roteador de borda da matriz e o roteador de borda da filial (uma em cada direção); para cada laptop do vendedor, há duas SAs entre o roteador de borda da matriz e o laptop (novamente, uma em cada direção). Portanto, no total, há $(2 + 2n)$ SAs. Mantenha em mente, entretanto, que nem todo o tráfego enviado à Internet pelos roteadores de borda ou pelos laptops será de IPsec seguro. Podemos citar como exemplo um hospedeiro na matriz que quer acessar ao servidor Web (como Amazon ou Google) na Internet pública. Assim, o roteador de borda (e os laptops) emitirão na Internet os datagramas IPv4 comuns e os datagramas IPsec seguros.

Vamos agora olhar “por dentro” de uma SA. Para tornar essa discussão tangível e concreta, utilizaremos o contexto de uma SA do roteador R1 ao roteador R2 na Figura 8.29. (Você pode pensar no Roteador 1 como o roteador de borda da matriz, e o Roteador 2 como o roteador de borda da filial, conforme Figura 8.28.) O Roteador R1 manterá as informações de estado sobre essa SA, as quais incluirão:

- Um identificador de 32 bits para a SA, denominado Índice de Parâmetro de Segurança (SPI)
- A interface remetente da SA (neste caso 200.168.1.100) e a interface destinatária da SA (neste caso 193.68.2.23)
- O tipo de criptografia a ser usada (por exemplo, 3DES com CBC)
- A chave de criptografia
- O tipo de verificação de integridade (por exemplo, HMAC com MD5)
- A chave de autenticação

Quando um roteador precisa construir um datagrama IPsec para encaminhar através dessa SA, ele acessa essas informações de estado para determinar como deveria autenticar e criptografar o datagrama. De maneira semelhante, o roteador R2 manterá as mesmas informações de estado sobre essa SA e as usará para autenticar e decriptografar qualquer datagrama IPsec que chegar da SA.

Uma entidade IPsec (roteador ou hospedeiro) muitas vezes guardam essas informações de estado para muitas SAs. No exemplo sobre a VPN na Figura 8.28 com n vendedores, o roteador de borda guarda informações de estado para $(2 + 2n)$ SAs. Uma entidade IPsec armazena as informações de estado para todas as suas SAs em seu Banco de Dados de Associação de Segurança (SAD), o qual é uma estrutura de dados do núcleo do sistema operacional.

8.6.4 O datagrama IPsec

Após descrever sobre as SAs, podemos agora descrever o verdadeiro datagrama IPsec. O IPsec possui duas formas diferentes de pacotes, uma para o modo túnel e outra para o modo transporte. O modo túnel, o mais apropriado para as VPNs, é mais implementado do que o modo transporte. Para desmistificar o IPsec e evitar

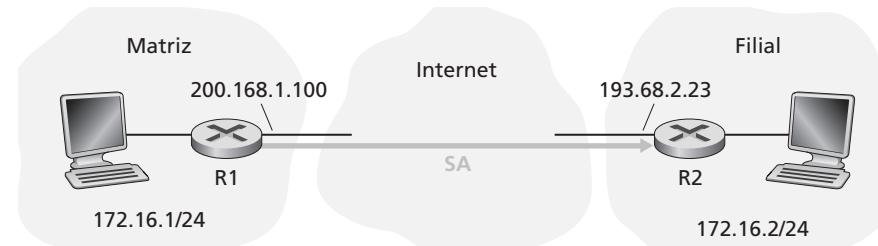


Figura 8.29 Associação de Segurança (SA) de R1 a R2

susas complexidades, a partir de agora nos focaremos exclusivamente no modo túnel. Uma vez que você tiver uma compreensão sólida do modo túnel, você poderá facilmente aprender, por sua conta, o modo transporte.

O formato do pacote do datagrama IPsec é ilustrado na Figura 8.30. Você pode pensar que os formatos do pacote são maçantes e desinteressantes, mas logo veremos que o datagrama IPsec na verdade parece e tem gosto de uma iguaria mexicana! Vamos examinar os campos do IPsec no contexto da Figura 8.29. Suponha que o roteador R1 receba um datagrama IPv4 comum do hospedeiro 172.16.1.17 (na rede da matriz) destinado ao hospedeiro 172.16.2.48 (na rede da filial). O roteador R1 utiliza a seguinte receita para converter esse “datagrama IPv4 original” em um datagrama IPsec:

- Anexa atrás do datagrama IPv4 original (que inclui os campos de cabeçalho originais) um campo “trailer ESP”
- Codifica o resultado utilizando o algoritmo e a chave especificados pela SA
- Anexa na frente dessa quantidade codificada um campo denominado “cabeçalho ESP”; o pacote resultante é denominado “enchilada”.
- Cria uma autenticação MAC por toda a enchilada usando o algoritmo e chave especificados na SA
- Anexa a MAC atrás da enchilada, formando a carga útil.
- Por fim, cria um cabeçalho IP novo com todos os campos de cabeçalho IPv4 clássicos (juntos normalmente com 20 bytes de comprimento), o qual se anexa antes da carga útil.

Observe que o datagrama IP resultante é um autêntico datagrama IPv4, com os tradicionais campos de cabeçalho IPv4 acompanhados por uma carga útil. Mas, neste caso, a carga útil contém um cabeçalho ESP, o datagrama IP original, um trailer ESP e um campo de autenticação ESP (com o datagrama original e o trailer ESP cifrados). O datagrama IP original possui o endereço IP remetente 172.16.1.17 e o endereço IP destinatário 172.16.2.48. Em razão de o datagrama IPsec incluir o datagrama IP original, esses endereços são inclusos (e cifrados) como parte da carga útil do pacote IPsec. Mas e os endereços IP remetente e destinatário que estão no novo cabeçalho IP, ou seja, o cabeçalho localizado à esquerda do datagrama IPsec? Como você pode esperar, eles são estabelecidos para as interfaces do roteador remetente e destinatário nas duas extremidades dos túneis, isto é, 200.168.1.100 e 193.68.2.23. Além disso, o número do protocolo nesse novo campo de cabeçalho IPv4 não será o número do TCP, UDP ou SMTP, mas igual a 50, determinando que esse é um datagrama IPsec que está utilizando o protocolo ESP.

Após R1 enviar um datagrama IPsec à Internet pública, ele passará por diversos roteadores antes de chegar ao R2. Cada um desses roteadores processará o datagrama como se fossem um datagrama comum — eles são completamente inconscientes do fato de que o datagrama está transportando dados cifrados pelo IPsec. Para esses roteadores da Internet pública, como o endereço IP remetente no cabeçalho externo é R2, o destino final do datagrama é R2.

Após acompanhar um exemplo de como o datagrama IPsec é construído, vamos olhar de perto os ingredientes da enchilada. Vemos na Figura 8.30 que o trailer ESP consiste em três campos: enchimento, tamanho do

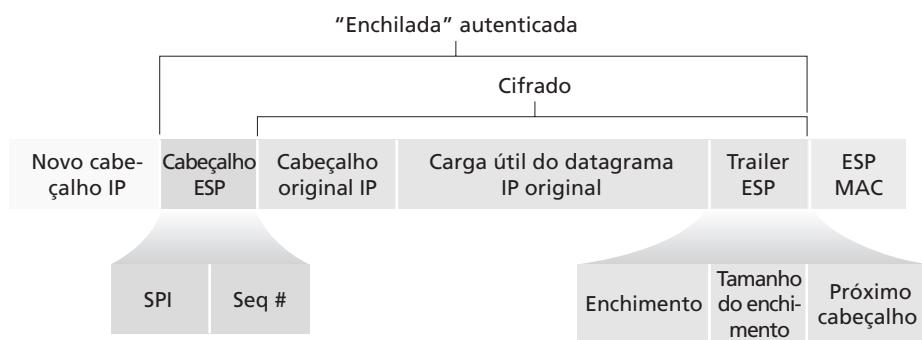


Figura 8.30 Formato do datagrama IPsec

enchimento e próximo cabeçalho. Lembre-se de que cifras de bloco exigem que a mensagem a ser cifrada seja um múltiplo inteiro do comprimento de bloco. O enchimento (que consiste de bytes insignificantes) é usado de modo que quando adicionado ao datagrama original (junto com o tamanho do enchimento e próximo cabeçalho), a “mensagem” resultante seja um número inteiro de blocos. O campo tamanho do enchimento indica à entidade destinatária quanto enchimento foi inserido (e, portanto, precisa ser removido). O próximo cabeçalho identifica o tipo (por exemplo, UDP) de dados contidos no campo de dados da carga útil. Os dados da carga útil (normalmente o datagrama IP original) e o trailer ESP são concatenados e, então, cifrados.

Anexado na frente dessa unidade cifrada está o cabeçalho ESP, o qual é enviado em aberto e consiste em dois campos: o SPI e o campo de número de sequência. O SPI indica à entidade destinatária a SA à qual o datagrama pertence; a entidade destinatária pode, então, indexar seu SAD com o SPI para determinar os algoritmos e chaves apropriados de autenticação/decriptação. O campo de número de sequência é usado para proteger ataques de repetição.

A entidade remetente também anexa uma autenticação MAC. Como afirmado anteriormente, a entidade remetente calcula um MAC por toda a enxilada (que consiste em um cabeçalho ESP, o datagrama IP original e o trailer ESP — com a criptografia do datagrama e do trailer). Lembre-se de que para calcular um MAC, o remetente anexa uma chave MAC secreta à enxilada e calcula um hash de tamanho fixo do resultado.

Quando R2 recebe o datagrama IPsec, ele observa que o endereço IP destinatário do datagrama é o próprio R2, que, então, processa o datagrama. Como o campo de protocolo (no cabeçalho IP à esquerda) é 50, R2 entende de que deve aplicar o processamento IPsec ESP ao datagrama. Primeiro, alinhando na enxilada, E2 usa o SPI para determinar à qual SA o datagrama pertence. Segundo, ele calcula o MAC da enxilada e verifica se o MAC é compatível com o valor do campo MAC ESP. Se for, ele sabe que a enxilada vem de R1 e que não foi adulterada. Terceiro, ele verifica o campo número de sequência para ver se o datagrama é novo (e não repetido). Quarto, ele decodifica a unidade cifrada utilizando o algoritmo e a chave de criptografia associados à SA. Quinto, ele remove o enchimento e extrai o datagrama IP original básico. E, finalmente, sexto, ele encaminha o datagrama original à rede da filial em direção a seu destino final. Ufa, que receita complicada, não é? Ninguém disse que era fácil preparar e desvendar uma enxilada!

Na verdade existe outra importante sutileza que precisa ser abordada. Ela se baseia na seguinte questão: Quando R1 recebe um datagrama (inseguro) de um hospedeiro na rede da matriz, e esse datagrama é destinado a algum endereço IP destinatário fora da matriz, como R1 sabe se ele deve ser convertido em um datagrama IPsec? E se ele vai ser processado por um IPsec, como R1 sabe qual SA (de muitas SAs em seu SAD) deve ser usada para construir o datagrama IPsec? O problema é resolvido da seguinte maneira. Junto com um SAD, a entidade IPsec também mantém outra estrutura de dados denominada Banco de Dados de Política de Segurança (SPD). O SPD indica que tipos de datagramas (como uma função do endereço IP remetente, endereço IP destinatário e tipo do protocolo) serão processadas pelo IPsec; e para aqueles que serão processados pelo IPsec, qual SA deve ser usada. De certa forma, as informações em um SPD indicam “o que” fazer com um datagrama que está chegando; as informações no SAD indicam “como” fazer isso.

Resumo dos serviços IPsec

Quais serviços o IPsec provê, exatamente? Vamos examinar esses serviços do ponto de vista de um atacante, digamos Trudy, que é uma mulher no meio, situada no caminho entre R1 e R2 na Figura 8.29. Suponha por toda essa discussão que Trudy não conhece as chaves de autenticação e criptografia usadas pela SA. O que Trudy pode e não pode fazer? Primeiro, Trudy não pode ver o datagrama original. Na verdade, não só os dados do datagrama original estão ocultos de Trudy como também o número de protocolo, o endereço IP remetente e o endereço IP destinatário. Em relação aos datagramas enviados através da SA, Trudy sabe somente que eles vieram de algum hospedeiro em 172.16.1.0/24 e é destinado a algum hospedeiro em 172.16.2.0/24. Ela não sabe se está carregando dados TCP, UDP ou ICMP; ela não sabe que está carregando HTTP, SMTP ou qualquer outros tipos de dados de aplicação. Esse sigilo, portanto, vai além do SSL. Segundo, suponha que Trudy tente adulterar um datagrama na SA alterando alguns de seus bits. Quando esse datagrama alterado chegar a R2, a verificação de integridade falhará (usando um MAC), impedindo a tentativa maliciosa de Trudy mais uma vez. Terceiro, suponha que Trudy

tente se passar por R1, criando um datagrama IPsec com remetente 200.168.1.100 e destinatário 193.68.2.23. O ataque de Trudy será inútil, pois a verificação da integridade do datagrama em R2 falhará novamente. Por fim, em razão de o IPsec incluir números de sequência, Trudy não poderá criar um ataque de repetição bem-sucedido. Em resumo, conforme afirmado no início desta seção, o IPsec provê — entre qualquer par de dispositivos que processam pacotes através da camada de rede — sigilo, autenticação da fonte, integridade dos dados e proteção contra ataque de repetição.

8.6.5 Gerenciamento de chave no IPsec

Quando uma VPN possui um número pequeno de pontos finais (por exemplo, apenas dois roteadores como na Figura 8.29), o administrador de rede pode inserir manualmente informações sobre a SA (algoritmos e chaves de criptografia/autenticação e os SPIs) nos SADs dos pontos de chegada. Essa “teclagem manual” é claramente impraticável para uma VPN grande, a qual pode consistir em centenas ou mesmo milhares de roteadores e hospedeiros IPsec. Implementações grandes e geograficamente distribuídas exigem um mecanismo automático para a criação das SAs. O IPsec o faz com o protocolo de Troca de Chave (IKE), especificado em RFC 4306.

O IKE possui semelhanças com a apresentação (*handshake*) em SSL (consulte Seção 8.5). Cada entidade IPsec possui um certificado, o qual inclui a chave pública da entidade. Da mesma forma que o SSL, o protocolo IKE tem os dois certificados de troca de entidades, autenticação de negociação e algoritmos de criptografia, e seguramente troca de materiais de chave para criar chaves de sessão nas SAs IPsec. Diferentemente do SSL, o IKE emprega duas fases para realizar essas tarefas.

Vamos investigar essas duas fases no contexto de dois roteadores, R1 e R2, na Figura 8.29. A primeira fase consiste em duas trocas de pares de mensagem entre R1 e R2:

Durante a primeira troca de mensagens, os dois lados usam Diffie-Hellman (consulte os Problemas) para criar um IKE SA bidirecional entre os roteadores. Para nos confundir, esse IKE SA bidirecional é totalmente diferente da SA IPsec discutida nas Seções 8.6.3 e 8.6.4. O IKE SA provê um canal cifrado e autenticado entre os dois roteadores. Durante a primeira troca de par de mensagem, as chaves são estabelecidas para a criptografia e autenticação para o IKE SA. Também é estabelecido um segredo mestre que será usado para computar chaves SA IPsec mais adiante na fase 2. Observe que durante a primeira etapa, as chaves públicas e privadas RSA não são usadas. Em particular, nem R1 nem R2 revelam sua identidade assinando uma mensagem com sua chave privada.

Durante a segunda troca de mensagens, ambos os lados revelam um ao outro sua identidade assinando suas mensagens. Entretanto, as identidades não são reveladas a um analisador passivo, visto que elas são enviadas através de um canal seguro IKE SA. Também durante essa fase, os dois lados negociam os algoritmos de autenticação e criptografia que serão empregados pelas SAs IPsec.

Na fase 2 do IKE, os dois lados criam uma SA em cada direção. Ao fim da fase 2, as chaves de sessão de autenticação e criptografia são estabelecidas em ambos os lados para as duas SAs. Os dois lados podem, então, usar as SAs para enviar datagramas seguros, como descrito na Seção 8.6.3 e 8.6.4. A principal motivação de ter duas fases no IKE é o custo computacional — visto que a segunda fase não envolve qualquer chave pública de criptografia, o IKE pode criar um grande número de SAs entre as duas entidades IPsec com um custo computacional relativamente pequeno.

8.7 Segurança em LANs sem fio

Segurança é uma preocupação importante em redes sem fio, onde as ondas de rádio carregando quadros podem propagar muito além da construção contendo os hospedeiros e as estações base sem fio. Nesta seção apresentaremos uma breve introdução em segurança sem fio. Para maiores informações, veja o altamente recomendado livro de Edney e Arbaugh [Edney, 2003].

O tema de segurança em 802.11 tem atraído muita atenção em círculos técnicos e na mídia. Apesar de discussões consideráveis terem sido feitas, poucos debates foram feitos — parece existir um entendimento universal de que a especificação 802.11 original contém uma série de falhas graves na segurança. De fato, podemos fazer o download de softwares de domínio público que tiram proveito dessas falhas, fazendo com que aqueles que usam mecanismos de segurança 802.11 comuns fiquem expostos a ataques de segurança tanto quanto aqueles que não usam nenhum tipo de atributos de segurança.

Na seção seguinte, discutiremos os mecanismos de segurança inicialmente padronizados na especificação 802.11, conhecida coletivamente como **Privacidade Equivalente Cabeada (WEP)**, do inglês *Wired Equivalent Privacy*). Como o nome sugere, a WEP tem como propósito fornecer um nível de segurança semelhante ao que é encontrado em redes cabeadas. Então, discutiremos algumas falhas de segurança da WEP e examinaremos o padrão 802.11i, uma versão fundamentalmente mais segura do que 802.11, adotado em 2004.

8.7.1 Privacidade Equivalente Cabeada (WEP)

O protocolo IEEE 802.11 WEP [IEEE 802.11 2009] fornece autenticação e criptografia de dados entre um hospedeiro e um ponto de acesso sem fio (ou seja, a estação base) usando uma abordagem de chave compartilhada simétrica. A WEP não especifica um gerenciamento de algoritmo de chave, então supomos que o hospedeiro e o ponto de acesso sem fio de alguma forma concordam sobre a chave através do método fora-de-banda. A autenticação é realizada como segue:

1. Um hospedeiro sem fio requisita uma autenticação por um ponto de acesso.
 2. O ponto de acesso responde à requisição de autenticação com um valor de 128 bytes nonce.
 3. O hospedeiro sem fio criptografa o nonce usando uma chave simétrica que compartilha com o ponto de acesso.
 4. O ponto de acesso decriptografa o nonce do hospedeiro criptografado.

Se o nonce decriptografado for compatível com o valor nonce originalmente enviado ao hospedeiro, então o hospedeiro é autenticado pelo ponto de acesso.

O algoritmo criptografado de dados WEP é ilustrado na Figura 8.31. Uma chave simétrica secreta de 40 bits, K_s , é presumidamente conhecida por ambos, hospedeiro e ponto de acesso. Além disso, um Vetor de Inicialização (IV, do inglês *Initialization Vector*) de 24 bits é anexado a uma chave de 40 bits para criar uma chave de 64 bits que serão usados para criptografar um único quadro. O IV mudará de um quadro para o outro, e, por conseguinte, cada quadro será criptografado com uma chave de 64 bits diferente.

A criptografia é efetuada como segue. Primeiramente, um valor de 4 bytes de CRC (veja a Seção 5.2) é calculado para a carga útil de dados. Então, a carga útil e o CRC de quatro bytes são criptografados usando uma cifra de fluxo RC4. Não veremos os detalhes de um RC4 aqui (veja [Schneier, 1995] e [Edney, 2003] para mais informações). Para nossos objetivos, é suficiente saber que, quando o valor de uma chave (nesse caso, a chave de 64 bits $[K_s, IV]$) é apresentado ao algoritmo RC4, este produz um fluxo de valores de chave $k_1^IV, k_2^IV, k_3^IV, \dots$ que são usados para criptografar os dados e o valor CRC em um quadro. Para propósitos práticos, podemos pensar

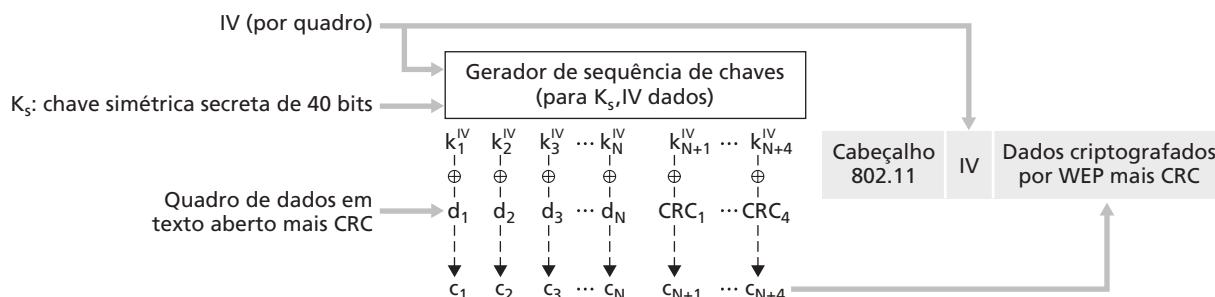


Figura 8.31 Protocolo 802.11 WEP

nessas operações como se fossem executadas em um byte por vez. A criptografia é realizada por OU-exclusivo no i -ésimo byte de dados, di, com a i -ésima chave, k_i^{IV} , no fluxo de valores de chaves produzido pelo par (K_s, IV) para produzir o i -ésimo byte do texto cifrado, c_i :

$$c_i = d_i \oplus k_i^{IV}$$

O valor de IV muda de um quadro ao próximo e é incluído no *texto aberto* do cabeçalho de cada quadro criptografado-WEP 802.11, como mostrado na Figura 8.31. O receptor aceita a chave simétrica secreta de 40 bits que compartilha com o transmissor, anexa o IV e usa a chave de 64 bits resultante (que é idêntica a chave usada pelo transmissor para executar a criptografia) para decriptografar o quadro:

$$d_i = c_i \oplus k_i^{IV}$$

O uso adequado do algoritmo RC4 necessita que o mesmo valor da chave de 64 bits *nunca* seja usado mais de uma vez. Lembre-se de que a chave WEP muda em uma base quadro a quadro. Para um determinado K_s (que muda em raras ocasiões), isso significa que existem somente 2^{24} chaves únicas. Se essas chaves são escolhidas de modo aleatório, podemos mostrar [Walker, 2000; Edney, 2003] que a probabilidade de ter escolhido o mesmo valor de IV (e, portanto, usado a mesma chave de 64 bits) é de mais de 99 por cento depois de somente 12.000 quadros. Com um quadro de 1 Kbyte e a transmissão de dados de velocidade 11 Mbps, somente alguns segundos são necessários antes que 12.000 quadros sejam transmitidos. Além do mais, já que IV é transmitido em texto aberto no quadro, um curioso saberá quando um valor duplicado for usado.

Para notar um dos vários problemas que ocorrem quando uma chave duplicada é usada, considere o seguinte ataque ao texto aberto escolhido pela Trudy à Alice. Suponha que a Trudy (usando provavelmente uma falsificação de IP) envia uma solicitação para Alice (por exemplo, uma solicitação http ou uma solicitação FTP) para transmitir um arquivo de conteúdo declarado, $d_1, d_2, d_3, d_4, \dots$. A Trudy também nota os dados criptografados $c_1, c_2, c_3, c_4, \dots$. Já que $d_i = c_i \oplus k_i^{IV}$, se OU-exclusivo c_i com cada lado dessa igualdade, teremos

$$d_i = c_i \oplus k_i^{IV}$$

Nesta relação, a Trudy pode usar os valores conhecidos de d_i e c_i para calcular k_i^{IV} . Da próxima vez que a Trudy vir o mesmo valor de VI sendo usado, ela saberá a sequência de chave $k_1^{IV}, k_2^{IV}, k_3^{IV}, \dots$ e assim será capaz de decriptografar a mensagem criptografada.

Existem outras diversas preocupações adicionais sobre a segurança da WEP. [Fluhrer, 2001] descreve um ataque aproveitando-se de uma fraqueza conhecida em RC4 quando certas chaves são escolhidas. [Stubblefield, 2002] discute maneiras eficazes de programar e tirar proveito deste ataque. Outra preocupação com a WEP envolve os bits CRC, mostrados na Figura 8.31, e transmitidos no quadro 802.11 para detectar bits alterados na carga útil. No entanto, um atacante que muda o conteúdo criptografado (por exemplo, substituindo textos ininteligíveis por dados criptografados originais), calcula o CEC através de textos ininteligíveis e coloca o CRC em um quadro WEP, pode produzir um quadro 802.11 que será aceito pelo receptor. O que é necessário nesse caso são técnicas de integridade de mensagem como aquelas estudadas na Seção 8.3 para detectar intercepções físicas de sinal ou substituições. Para maiores informações sobre segurança da WEP, veja [Edney, 2003; Walker, 2000; Weatherspoon, 2000; Segurança 802.11 2009] e as referências nestes a esse respeito.

8.7.2 IEEE 802.11i

Logo após o lançamento do IEEE 802.11 em 1999, o trabalho no desenvolvimento de uma versão nova e aprimorada do 802.11, com mecanismos de segurança mais fortes, começou. O novo padrão, conhecido como 802.11i, passou por uma ratificação final em 2004. Como veremos, enquanto a WEP fornecia uma criptografia relativamente fraca, somente um meio de executar autenticação e nenhum mecanismo de distribuição de chaves, o IEEE 802.11i fornece formas muito mais fortes de criptografia, um conjunto extenso de mecanismos de autenticação e um mecanismo de distribuição de chaves. A seguir, apresentamos uma síntese do 802.11i; um excelente panorama técnico (em fluxo de áudio) sobre 802.11i é [TechOnline, 2004].

A Figura 8.32 dá uma visão geral sobre o enquadramento 802.11i. Além do cliente sem fio e do ponto de acesso, 802.11i define um servidor de autenticação, com o qual o AP se comunica. Separar o servidor de comuni-

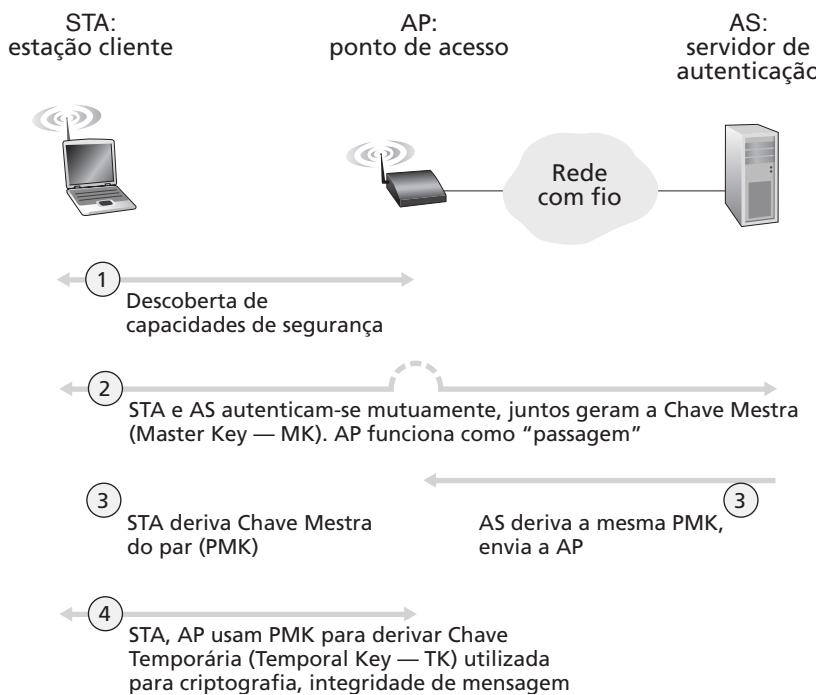


Figura 8.32 802.11i: quatro fases de operação

cação do AP permite que um servidor de autenticação sirva muitos APs, centralizando as decisões (frequentemente sensíveis) relativas à autenticação e acesso dentro de um único servidor, e deixando os custos e a complexidade do AP menores. 802.11i opera em quatro fases:

1. *Descoberta.* Na fase de descobrimento, o AP anuncia sua presença e as formas de autenticação e criptografia que podem ser fornecidas ao nó do cliente sem fio. Então, o cliente solicita as formas específicas de autenticação e criptografia que deseja. Apesar de o cliente e o AP já estarem trocando mensagens, o cliente ainda não foi autenticado nem tem uma chave criptografada, então vários passos ainda serão necessários antes que o cliente possa se comunicar com um hospedeiro arbitrário remoto através de um canal sem fio.
2. *Autenticação mútua e geração da Chave Mestra (MK).* A autenticação ocorre entre o cliente sem fio e o servidor de autenticação. Nesta fase, o ponto de acesso age essencialmente como um relé, encaminhando mensagens entre o cliente e o servidor de autenticação. O **Protocolo de Autenticação Extensível (EAP)**, do inglês *Extensible Authentication Protocol* [RFC 2284] define o formato da mensagem fim a fim usado em um simples modo de requisição/resposta de interação entre o cliente e o servidor de autenticação. Como mostrado na Figura 8.33, as mensagens EAP são encapsuladas usando um **EAPoL** (EAP através da LAN, [IEEE 802.1X]) e enviadas através de um enlace 802.11 sem fio. Então, estas mensagens EAP são desen- capsuladas no ponto de acesso, e re-encapsuladas usando um protocolo **RADIUS** para a transmissão através de UDP/IP ao servidor de autenticação. Enquanto o protocolo e o servidor RADIUS [RFC 2865] não são requisitados pelo protocolo 802.11i, eles são, na prática, componentes padrão para 802.11i. O protocolo recentemente padronizado **DIAMETER** [RFC 3588] é propenso a substituir o RADIUS em um futuro próximo.

Com o EAP, o serviço de autenticação pode escolher diversos modos para realizar a autenticação. Enquanto 802.11i não dita um método específico de autenticação, o esquema de autenticação EAP-TLS [RFC 2716] é frequentemente usado. O EAP-TLS usa técnicas de chaves públicas (incluindo a criptografia nonce e resumos de mensagens) semelhantes às que estudamos na Seção 8.3, que permitem que o cliente e o servidor de autenticação se autentiquem mutuamente, e produzam uma Chave Mestra (MK, do inglês *Master Key*) que é conhecida por ambos os participantes.

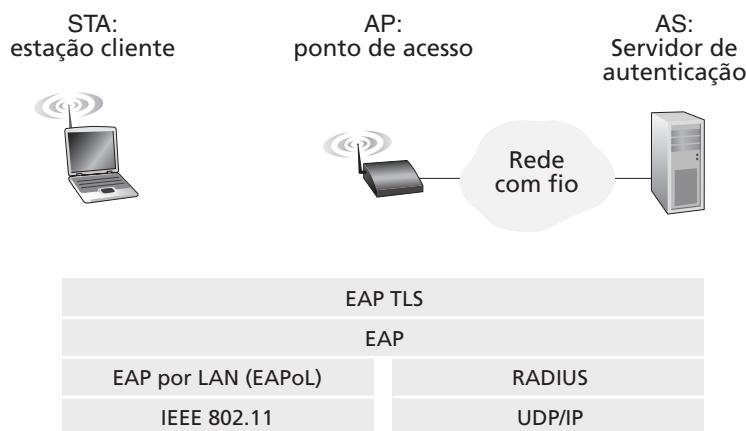


Figura 8.33 O EAP é um protocolo fim a fim. As mensagens EAP são encapsuladas usando um EAPoL através de um enlace sem fio entre o cliente e o ponto de acesso, e usando RADIUS através de UDP/IP entre o ponto de acesso e o servidor de autenticação.

3. *Geração de Chave Mestra Pareada (PMK).* A MK é compartilhada secretamente somente para o cliente e para o servidor de autenticação, sendo ela usada por estes para gerar uma segunda chave, a Chave Mestra Pareada (PMK, do inglês *Pairwise Master Key*). Então, o servidor de autenticação envia a PMK ao AP. E é aqui que queríamos chegar! O cliente e o AP têm agora uma chave compartilhada (lembre-se de que em WEP, o problema da distribuição de chaves não foi nem citado) e se autenticaram mutuamente. Agora eles estão quase prontos para entrar em ação.
4. *Geração de Chave Temporal (TK).* Com a PMK, o cliente sem fio e o AP podem agora gerar chaves adicionais que serão usadas para comunicação. De interesse particular temos a Chave Temporal (TK, do inglês *Temporal Key*), que será usada para implementar a criptografia de nível de enlace através do enlace sem fio e um hospedeiro remoto arbitrário. 802.11i fornece várias formas de criptografia, incluindo um esquema de criptografia baseada em AES e uma versão reforçada da criptografia WEP.

8.8 Segurança operacional: firewalls e sistemas de detecção de invasão

Vimos, em todo este capítulo, que a Internet não é um lugar muito seguro — nela, os delinquentes estão por toda parte, criando todo tipo de destruição. Dada a natureza hostil da Internet, vamos considerar uma rede de organização e um administrador de rede que a administra. Do ponto de vista de um administrador, o mundo está dividido claramente em dois campos — os bonzinhos (que pertencem à organização que administra a rede e que deveriam poder acessar recursos dentro da rede que ele administra de um modo relativamente livre de restrições) e os bandidos (todo o resto, cujo acesso aos recursos da rede deve ser cuidadosamente inspecionado). Em muitas organizações, que vão desde castelos medievais a modernos escritórios de empresas, há um único ponto de entrada/saída onde ambos, bonzinhos e bandidos que entram e saem da organização, passam por inspeção de segurança. Em castelos medievais, essa inspeção era feita em um portão, na extremidade de uma ponte levadiça; em escritórios empresariais, a inspeção é feita na central de segurança. Em redes de computadores, quando o tráfego que entra/sai de uma rede passa por inspeção de segurança, é registrado, descartado ou transmitido; isso é feito por mecanismos operacionais conhecidos como firewalls, sistemas de detecção de invasão (IDSs) e sistemas de prevenção de invasão (IPSs).

8.8.1 Firewalls

Um **firewall** é uma combinação de hardware e software que isola a rede interna de uma organização da Internet em geral, permitindo que alguns pacotes passem e bloqueando outros. Um firewall permite que um

administrador de rede controle o acesso entre o mundo externo e os recursos da rede que administra gerenciando o fluxo de tráfego de e para esses recursos. Um firewall possui três objetivos:

Todo o tráfego de fora para dentro, e vice-versa, passa por um firewall. A Figura 8.34 mostra um firewall, situado diretamente no limite entre a rede administrada e o resto da Internet. Enquanto grandes organizações podem usar diversos níveis de firewalls ou firewalls distribuídos [Skoudis, 2006], alocar um firewall em um único ponto de acesso à rede, conforme mostrado na Figura 8.34, facilita o gerenciamento e a execução de uma política de acesso seguro.

Somente o tráfego autorizado, como definido pela política de segurança local, poderá passar. Com todo o tráfego que entra e sai da rede institucional passando pelo firewall, este pode limitar o acesso a tráfego autorizado.

O próprio firewall é imune à penetração. O próprio firewall é um mecanismo conectado à rede. Se não projetado ou instalado adequadamente, ele pode ser comprometedor, podendo oferecer apenas uma falsa sensação de segurança (que é pior do que não ter nenhum firewall!).

Cisco e Check Point são dois dos principais fornecedores de firewall hoje. Você pode criar um firewall facilmente (filtro de pacote) a partir de um sistema Linux usando tabelas IP (software de domínio público que normalmente acompanha o Linux).

Os firewalls podem ser classificados em três categorias: filtros de pacote tradicionais, filtros de estado e gateways de aplicação. Abordaremos cada um deles nas subseções seguintes.

Filtros de pacote tradicionais

Como mostra a Figura 8.34, uma organização normalmente tem um roteador de borda que conecta sua rede interna com seu ISP (e dali com a Internet pública, mais ampla). Todo o tráfego que sai ou que entra na rede interna passa por esse roteador e é nesse roteador que ocorre a **filtragem de pacotes**. Um filtro de pacote examina cada datagrama que está sozinho, determinando se o datagrama deve passar ou ficar baseado nas regras específicas do administrador.

As decisões de filtragem são normalmente baseadas em:

- Endereço IP de origem e de destino.
- Tipo de protocolo no campo do datagrama IP: TCP, UDP, ICMP, OSPF etc.

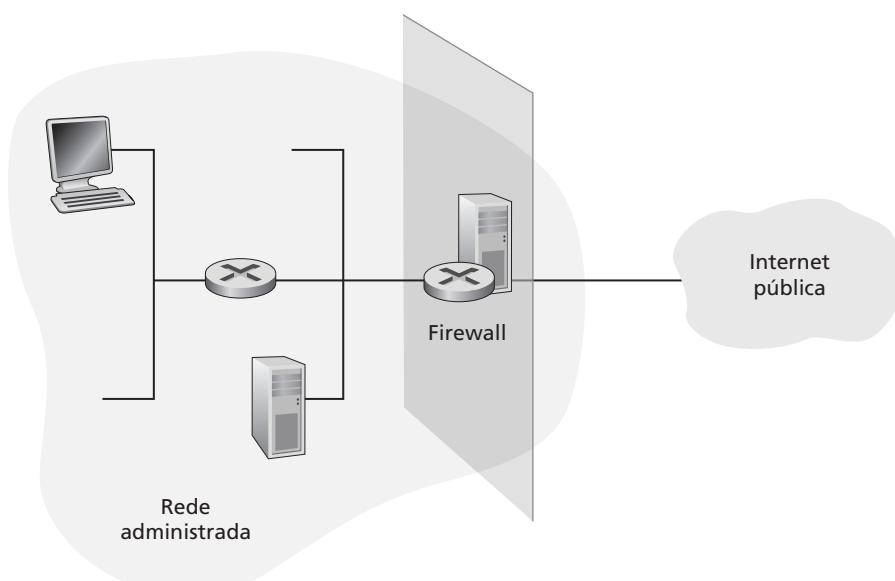


Figura 8.34 Posição do firewall entre a rede administrada e o mundo exterior

- Porta TCP ou UDP de origem e de destino.
- Flag bits do TCP: SYN, ACK etc.
- Tipo de mensagem ICMP.
- Regras diferentes para datagramas que entram e saem da rede
- Regras diferentes para interfaces do roteador diferentes

Um administrador de rede configura o firewall com base na política da organização. A política pode considerar a produtividade do usuário e o uso de largura de banda, bem como as preocupações com segurança da organização. A Tabela 8.5 lista um número de políticas possíveis que uma organização pode ter, e como elas seriam endereçadas com um filtro de pacotes. Por exemplo, se a organização não quer nenhuma conexão TCP de entrada, exceto aquelas para o servidor Web público, ela pode bloquear todos os segmentos TCP SYN, com exceção dos segmentos TCP SYN com porta de destino 80 e endereço IP destinatário correspondente ao servidor Web. Se a organização não quer que seus usuários monopolizem o acesso à largura de banda com aplicações de rádio via Internet, ela pode bloquear todo o tráfego UDP não importante (visto que o rádio via Internet é normalmente enviado por UDP). Se a organização não quer que sua rede interna seja mapeada (rastreio de rota) por um estranho, ela pode bloquear todas as mensagens ICMP TTL expiradas que saem da rede da organização.

Uma política de filtragem também pode ser baseada na combinação de endereços e números de porta. Por exemplo, o roteador de filtragem poderia bloquear todos os datagramas Telnet (os que têm número de porta 23), exceto os que estão vindo ou indo de ou para uma lista de endereços IP específicos. Essa política permite conexões Telnet de e para os hospedeiros que estão na lista. Infelizmente, basear a política em endereços externos não oferece nenhuma proteção contra datagramas cujo endereço de fonte foi falsificado.

A filtragem pode também ser baseada no bit TCP ACK ter ou não valor. Esse estratagema é bastante útil quando uma organização quer permitir que seus clientes internos se conectem com servidores externos, mas quer impedir que clientes externos se conectem com servidores internos. Lembre-se de que o primeiro segmento de todas as conexões TCP (veja a Seção 3.5) tem o bit ACK com valor 0, ao passo que todos os outros segmentos da conexão têm o bit ACK com valor 1. Assim, se uma organização quiser impedir que clientes externos iniciem uma conexão com servidores internos, ela simplesmente filtrará todos os segmentos que entram que tenham o bit ACK ajustado em 0. Essa política elimina todas as conexões TCP originadas do exterior, mas permite conexões que se originam internamente.

As regras do firewall são implementadas em roteadores com listas de controle de acesso, tendo cada interface do roteador sua própria lista. Um exemplo de uma lista de controle de acesso para uma organização 222.22/16 é ilustrado na Tabela 8.6. Essa lista é para uma interface que conecta o roteador aos ISPs externos da organização. As regras são aplicadas a cada datagrama que atravessa a interface de cima para baixo. As primeiras duas regras juntas permitem que usuários internos naveguem na Web: a primeira regra permite qualquer pacote TCP com porta de destino 80 saia da rede da organização: a segunda regra permite que qualquer pacote TCP com porta de origem 80 e o bit ACK entram na rede da organização. Observe que se uma fonte externa tenta estabelecer uma

Política	Configuração de firewall
Não há acesso exterior à Web	Abandonar todos os pacotes de saída para qualquer endereço IP, porta 80
Não há conexões TCP de entrada	Abandonar todos os pacotes TCP SYN para qualquer IP exceto 130.207.244.203, porta 80
Impedir que rádios Web comam a largura de banda disponível	Abandonar todos os pacotes UDP de entrada — exceto pacotes DNS.
Impedir que sua rede seja usada por um ataque DoS smurf	Abandonar todos os pacotes ping que estão indo para um endereço "broadcast" (por exemplo, 130.207.255.255)
Impedir que a rota de sua rede seja rastreada	Abandonar todo o tráfego de saída expirado ICMP TTL

Tabela 8.5 Políticas e regras de filtragem correspondentes para uma rede da organização 130.27/16 com servidor Web 130.207.244.203

Ação	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Flag bit
Permitir	222.22/16	Fora de 222.22/16	TCP	>1023	80	Qualquer um
Permitir	Fora de 222.22/16	222.22/16	TCP	80	>1023	ACK
Permitir	222.22/16	Fora de 222.22/16	UDP	>1023	53	—
Permitir	Fora de 222.22/16	222.22/16	UDP	53	>1023	—
Negar	Todos	Todos	Todos	Todos	Todos	Todos

Tabela 8.6 Lista de controle de acesso para uma interface do roteador

conexão TCP com um hospedeiro interno, a conexão será bloqueada, mesmo que a porta de origem ou destino seja 80. As segundas duas regras juntas permitem que pacotes DNS entrem e saiam da rede da organização. Em resumo, essa lista de controle de acesso limitada bloqueia todo o tráfego exceto o tráfego Web iniciado de dentro da organização e do tráfego DNS. [CERT Filtering, 2009] apresenta uma lista de portas/protocolos para a filtragem de pacotes para evitar diversos buracos de segurança conhecidos em aplicações de rede existentes.

Filtros de pacote com controle de estado

Em um filtro de pacote tradicional, as decisões de filtragem são feitas em cada pacote isolado. Os filtros de estado rastreiam conexões TCP e usam esse conhecimento para tomar decisões sobre filtragem.

Para entender esses filtros de estado, vamos reexaminar a lista de controle de acesso da Tabela 8.6. Embora um tanto limitada, essa lista, no entanto, permite que qualquer pacote que chegue de fora com um ACK = 1 e porta de origem 80 atravesse o filtro. Esses pacotes poderiam ser usados em tentativas de destruir o sistema interno com pacotes defeituosos, realizar ataques de recusa de serviços, ou mapear a rede interna. Uma solução ingênua é bloquear pacotes TCP ACK também, mas tal método impediria que os usuários internos da organização navegassem na Web.

Os filtros de estado resolvem esse problema rastreando todas as conexões TCP de entrada em uma tabela de conexão. Isso é possível porque o firewall pode notar o início de uma nova conexão observando uma apresentação de três vias (SYN, SYNACK e ACK); ele pode observar o fim de uma conexão ao ver um pacote FIN para a conexão. O firewall também pode (de forma conservadora) admitir que a conexão está finalizada quando não observou nenhuma atividade no decorrer da conexão, digamos, por 60 segundos. Um exemplo de tabela de conexão para um firewall é mostrado na Tabela 8.7. Essa tabela indica que, no momento, há três conexões TCP em andamento, as quais foram iniciadas dentro da organização. Ademais, o filtro de estado inclui uma nova coluna, “conexão de verificação”, em sua lista de controle de acesso, conforme mostrado na Tabela 8.8. Observe que essa tabela é idêntica à lista de controle de acesso da Tabela 8.6, exceto que ela indica que a conexão deve ser verificada para duas das regras.

Vamos analisar alguns exemplos e ver como a tabela de conexão e a lista de controle de acesso trabalham em conjunto. Suponha que um atacante tente enviar um pacote defeituoso para a rede da organização por meio de um

Endereço de origem	Endereço de destino	Porta de origem	Porta de destino
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

Tabela 8.7 Tabela de conexão para o filtro de estado

Ação	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Flag bit	Conexão de checagem
Permitir	222.22/16	Fora de 222.22/16	TCP	>1023	80	Qualquer um	
Permitir	Fora de 222.22/16	222.22/16	TCP	80	>1023	ACK	X
Permitir	222.22/16	Fora de 222.22/16	UDP	>1023	53	—	
Permitir	Fora de 222.22/16	222.22/16	UDP	53	>1023	—	X
Negar	Todos	Todos	Todos	Todos	Todos	Todos	

Tabela 8.8 Lista de controle de acesso para filtro de estado

datagrama com porta de origem TCP 80 e com o flag ACK definido. Suponha ainda que esse pacote possua um número de porta de origem 12543 e endereço IP remetente 150.23.23.155. Quando esse pacote chega ao firewall, este verifica a lista de controle de acesso da Tabela 8.8, que indica que a tabela de conexão deve também ser verificada antes de permitir que esse pacote entre na rede da organização. O firewall verifica a tabela de conexão e observa que esse pacote não faz parte de uma conexão TCP em andamento e o rejeita.

Como um segundo exemplo, suponha que um usuário interno queira navegar em um site Web externo. Como o usuário primeiro envia um segmento TCP SYN, sua conexão TCP é registrada na tabela de conexão. Quando o servidor Web envia pacotes de volta (com o bit ACK necessariamente definido), o firewall verifica a tabela e observa que uma conexão correspondente está em progresso. O firewall, então, deixará esses pacotes passarem, sem interferir na navegação do usuário interno.

Gateway de aplicação

Nos exemplos que acabamos de mostrar, vimos que a filtragem de pacotes permite que uma organização faça uma filtragem grosseira de conteúdos de cabeçalhos IP e TCP/UDP, incluindo endereços IP, números de porta e bits de reconhecimento. Vimos, por exemplo, que a filtragem baseada em uma combinação de endereços IP e números de porta pode permitir que clientes internos executem Telnet para o exterior, ao mesmo tempo que impede que clientes externos executem Telnet para o interior. Mas, e se uma organização quiser fornecer o serviço Telnet a um conjunto restrito de usuários internos (em vez de a endereços IP)? E se a organização quiser que esses usuários privilegiados se autentiquem antes de obter permissão para criar sessões Telnet com o mundo externo? Essas tarefas estão além das capacidades de um filtro. Na verdade, informações sobre a identidade de usuários internos não estão incluídas nos cabeçalhos IP/TCP/UDP; elas estão nos dados da camada de aplicação.

Para assegurar um nível mais refinado de segurança, os firewalls têm de combinar filtro de pacotes com gateways de aplicação. Gateways de aplicação fazem mais do que examinar cabeçalhos IP/TCP/UDP e tomam decisões com base em dados da aplicação. Um **gateway de aplicação** é um servidor específico de aplicação através do qual todos os dados da aplicação (que entram e que saem) devem passar. Vários gateways de aplicação podem executar no mesmo hospedeiro, mas cada gateway é um servidor separado, com seus próprios processos.

Para termos uma percepção dos gateways de aplicação, vamos projetar um firewall que permite que apenas um conjunto restrito de usuários execute Telnet para o exterior e impede que todos os clientes externos executem Telnet para o interior. Essa política pode ser aplicada pela implementação da combinação de um filtro de pacotes (em um roteador) com um gateway de aplicação de Telnet, como mostra a Figura 8.35. O filtro do roteador está configurado para bloquear todas as conexões Telnet, exceto aquelas que se originam do endereço IP do gateway de aplicação. Essa configuração de filtro força todas as conexões Telnet de saída a passarem pelo gateway de aplicação. Considere agora um usuário interno que quer executar Telnet com o mundo exterior. Em primeiro lugar, ele tem de estabelecer uma sessão Telnet com o gateway de aplicação. Uma aplicação que está executando

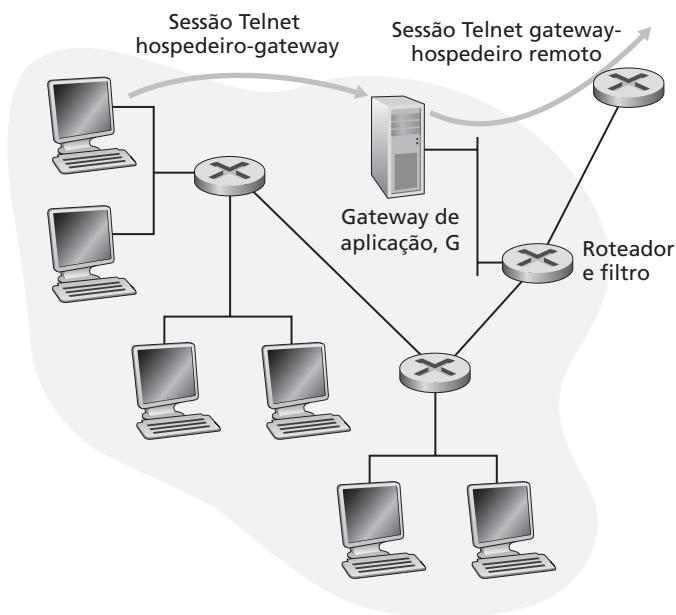


Figura 8.35 Firewall composto de um gateway de aplicação e um filtro

no gateway — e que fica à escuta de sessões Telnet que entram — solicita ao usuário sua identificação e senha. Quando o usuário fornece essas informações, o gateway de aplicação verifica se ele tem permissão para executar Telnet com o mundo exterior. Se não tiver, a conexão Telnet do usuário interno ao gateway será encerrada pelo gateway. Se o usuário tiver permissão, o gateway (1) pedirá ao usuário o nome do computador externo com o qual ele quer se conectar, (2) estabelecerá uma sessão Telnet entre o gateway e o hospedeiro externo e (3) passará ao hospedeiro externo todos os dados que chegam do usuário e ao usuário todos os dados que chegam do hospedeiro externo. Assim, o gateway de aplicação Telnet não só autoriza o usuário, mas também atua como um servidor Telnet e um cliente Telnet, passando informações entre o usuário e o servidor Telnet remoto. Note que o filtro permitirá a etapa 2, porque é o gateway que inicia a conexão Telnet com o mundo exterior.

Redes internas frequentemente têm vários gateways de aplicação, como gateways para Telnet, HTTP, FTP e e-mail. De fato, o servidor de correio (veja a Seção 2.4) e o cache Web de uma organização são gateways de aplicação.

Gateways de aplicação não estão isentos de desvantagens. Em primeiro lugar, é preciso um gateway de aplicação diferente para cada aplicação diferente. Em segundo lugar, há um preço a pagar em termos de desempenho, visto que todos os dados serão repassados por meio do gateway. Isso se torna uma preocupação particularmente quando vários usuários ou aplicações estão utilizando o mesmo gateway. Por fim, o software cliente deve saber como entrar em contato com o gateway quando o usuário fizer uma solicitação, e deve saber como dizer ao gateway de aplicação a qual servidor se conectar.

8.8.2 Sistemas de detecção de intrusos

Acabamos de ver que um filtro de pacote (tradicional e de estado) inspeciona campos de cabeçalho IP, TCP, UDP e ICMP quando está decidindo quais pacotes deixará passar através do firewall. No entanto, para detectar muitos tipos de ataque, precisamos executar uma **inspeção profunda de pacote**, ou seja, precisamos olhar através dos campos de cabeçalho e dentro dos dados da aplicação que o pacote carrega. Como vimos na Seção 8.8.1, gateways de aplicação frequentemente fazem inspeções profundas de pacote. Mas um gateway de aplicação só executa isto para uma aplicação específica.

Claramente existe espaço para mais um dispositivo — um dispositivo que não só examina os cabeçalhos de todos os pacotes ao passar por eles (como um filtro de pacote), mas também executa uma inspeção profunda de pacote (diferentemente do filtro de pacote). Quando tal dispositivo observa o pacote suspeito, ou uma série de

História

Anonimato e privacidade

Suponha que você queira visitar aquela polêmica página Web (por exemplo, o site de um ativista político) e você (1) não quer revelar seu endereço IP à página da Web, (2) não quer que o seu ISP (que pode ser o ISP de sua casa ou do escritório) saiba que você está visitando este site, e (3) não quer que o seu IP local veja os dados que você está compartilhando com o site. Se você usa a abordagem tradicional de conexão direta à página Web sem nenhuma criptografia, você falhará em seus três objetivos. Mesmo que você use SSL, você falhará nas duas primeiras questões: seu endereço IP fonte é apresentado à página Web em todo datagrama enviado; e o endereço de destino de cada pacote enviado pode facilmente ser analisado pelo seu ISP local.

Para obter anonimato e privacidade, você pode usar uma combinação de um servidor proxy confiável e SSL, como mostrado na Figura 8.36. Com essa abordagem, você faz uma conexão SSL com o proxy confiável. Então envie, nessa conexão SSL, uma solicitação HTTP para o site desejado. Quando o proxy receber essa solicitação SSL de HTTP criptografado, ele decriptografará a solicitação e encaminhará o texto claro da solicitação HTTP à página Web. A página Web responde ao proxy, que por sua vez encaminha a resposta a você através do SSL. Como a página Web só vê o endereço IP do proxy, e não o endereço do seu cliente, você está de fato obtendo um acesso anônimo a página Web. E devido ao tráfego entre você e o proxy ser criptografado, seu ISP local não pode invadir sua privacidade ao logar no site que você visitou ou gravar os dados que estavam sendo compartilhados. Atualmente, muitas empresas disponibilizam tais serviços proxy, (como a www.proxyfy.com).

É claro que, ao usar esta solução, seu proxy saberá tudo: saberá seu endereço IP e o endereço IP do site que você está visitando; pode ver todo o tráfego em texto claro compartilhado entre você e a página Web. Uma abordagem melhor, adotada pelo serviço de anonimato e privacidade TOR, é sequenciar seu tráfego através de uma série de servidores proxys que não compartilham informações entre si [TOR 2009]. Em particular, o TOR permite que indivíduos independentes contribuam com proxys para seu acervo. Quando um usuário se conecta a um servidor usando o TOR, ele escolhe aleatoriamente (de seu acervo de proxys) uma corrente de três proxys e sequencia todo o tráfego entre cliente e servidor através dessa corrente. Dessa maneira, supondo que os proxys não trocam informações entre si, ninguém percebe que ocorreu uma comunicação entre seu endereço IP e a página da Web desejada. Além disso, apesar de o texto claro ser enviado entre o último proxy e o servidor, o último proxy não sabe qual endereço IP está enviando ou recebendo o texto claro.

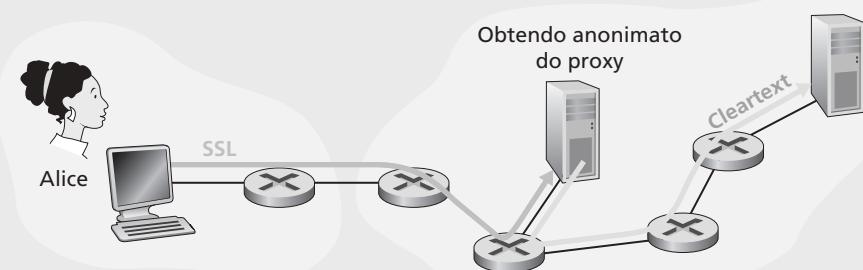


Figura 8.36 Fornecimento de anonimato e privacidade com um proxy

pacotes suspeitos, ele impede que tais pacotes entrem na rede organizacional. Ou, quando a atividade só é vista como suspeita, o dispositivo pode deixar os pacotes passarem, mas envia um alerta ao administrador de rede, que pode examinar o tráfego minuciosamente e tomar as ações necessárias. Um dispositivo que gera alertas quando observa tráfegos potencialmente mal intencionados é chamado de **sistema de detecção de intrusos (IDS)**, do inglês *intrusion detection system*). Um dispositivo que filtra o tráfego suspeito é chamado de **sistema de prevenção de intrusão (IPS**, do inglês *intrusion prevention system*). Nesta seção, estudaremos ambos os sistemas — IDS e IPS — já que o mais interessante aspecto técnico desses sistemas é como eles detectam tráfego suspeito (em vez de enviarem alertas ou abandonar pacotes). Daqui para a frente iremos nos referir ao sistema IDS e ao sistema IPS como sistema IDS.

Um IDS pode ser usado para detectar uma série de tipos de ataques, incluindo mapeamento de rede (provinho, por exemplo, de nmap), escaneamento de portas, escaneamento de pilha TCP, ataques de inundação de banda larga DoS, worms e vírus, ataques de vulnerabilidade de OS e ataques de vulnerabilidade de aplicações. (Veja a Seção 1.6 para um tutorial sobre ataques de rede). Atualmente, milhares de organizações empregam sistemas IDS. Muitos desses sistemas implementados são patenteados, comercializados pela Cisco, Check Point, e outros fornecedores de equipamentos de segurança. Mas muitos desses sistemas IDS implementados são sistemas de domínio público, como o sistema extremamente popular Snort IDS (o qual iremos discutir em breve).

Uma organização pode implementar um ou mais sensores DNS em sua rede organizacional. A Figura 8.37 mostra uma organização que tem três sensores IDS. Quando múltiplos sistemas são implementados, eles trabalham tipicamente em harmonia, enviando informações sobre atividades de tráfegos suspeitos ao processador central IDS, que coleta e integra essas informações e envia alarmes aos administradores da rede quando acharem apropriado. Na Figura 8.37, a organização dividiu sua rede em duas regiões: uma região de segurança máxima, protegida por um filtro de pacote e um gateway de aplicação e monitorada por sensores IDS; e uma região de segurança baixa — referida como **zona desmilitarizada (DMZ**, do inglês *demilitarized zone*) — protegida somente por um filtro de pacote, mas também monitorada por sensores IDS. Observe que a DMZ inclui os servidores da organização que precisam se comunicar com o mundo externo, como um servidor Web e seus servidores autoritativos. Você deve estar se perguntando, por que sensores IDS? Por que não colocar um sensor IDS atrás do filtro de pacote (ou até integrá-lo ao filtro de pacote) da Figura 8.37? Logo veremos que um IDS não só precisa fazer uma inspeção pro-

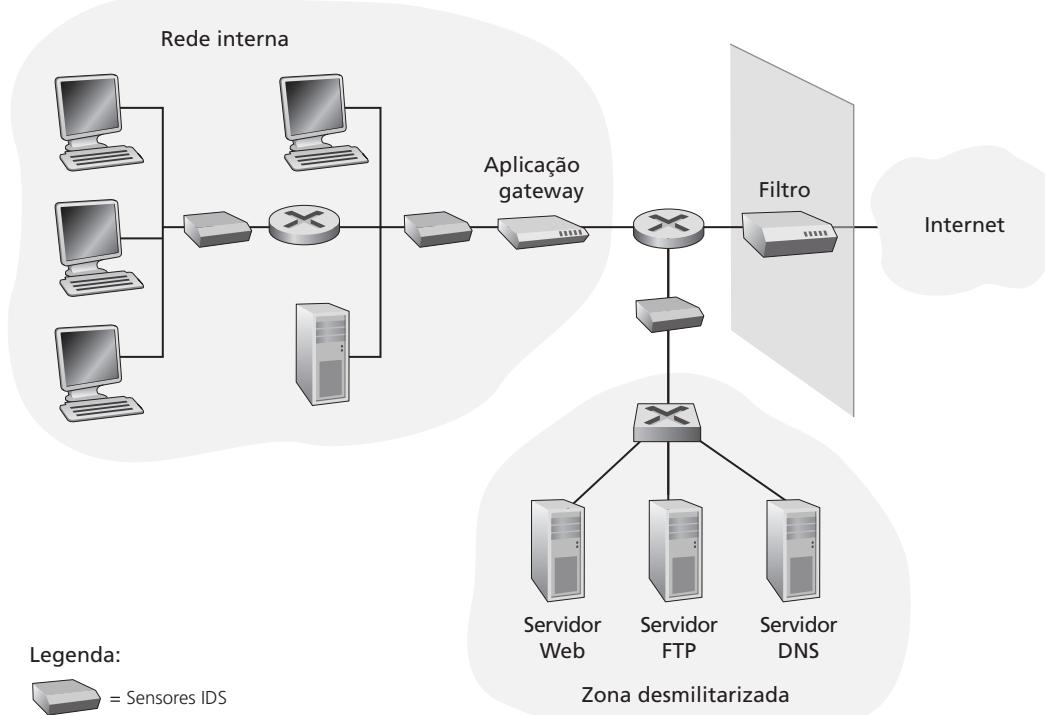


Figura 8.37 Uma organização implementando um filtro, uma aplicação gateway e sensores IDS

funda de pacote, como também compara cada pacote que passa com milhares de “assinaturas”; isso pode ser um volume de processamento significativo, particularmente se a organização recebe gigabits/seg de tráfego da Internet. Ao colocar sensores IDS além da transmissão, cada sensor só vê uma fração do tráfego da organização, e pode facilmente manter um ritmo. No entanto, sistemas IDS e IPS de alto desempenho estão disponíveis hoje e muitas organizações podem acompanhar com somente um sensor localizado próximo ao roteador de acesso.

Sistemas IDS são amplamente classificados tanto como **sistemas baseados em assinatura**, ou **sistemas baseados em anomalias**. Um IDS baseado em assinatura mantém um banco de dados extenso de ataques de assinaturas. Cada assinatura é um conjunto de regras relacionadas a uma atividade de intrusos. Uma assinatura pode ser uma lista de características sobre um único pacote (por exemplo, números de portas de origem e destino, tipo de protocolo, e uma sequência de bits em uma carga útil de um pacote), ou pode estar relacionada a uma série de pacotes. As assinaturas são normalmente criadas por engenheiros habilidosos em segurança de rede que tenham pesquisado ataques conhecidos. Um administrador de rede de uma organização pode personalizar as assinaturas ou inserir as próprias no banco de dados.

Operacionalmente, uma IDS baseada em assinatura analisa cada pacote que passa, comparando cada pacote analisado com as assinaturas no banco de dados. Se um pacote (ou uma série de pacotes) é compatível com uma assinatura no banco de dados, o IDS gera um alerta. O alerta pode ser enviado ao administrador da rede por uma mensagem de email, pode ser enviada ao sistema de administração de rede, ou pode simplesmente ser registrado para futuras inspeções.

Apesar de os sistemas IDS baseados em assinaturas serem amplamente implementados, eles têm uma série de limitações. Acima de tudo, eles requerem conhecimento prévio do ataque para gerar uma assinatura precisa. Ou seja, um IDS baseado em assinatura é completamente cego a novos ataques que ainda não foram registrados. Outra desvantagem é que mesmo se uma assinatura for compatível, pode não ser o resultado de um ataque, mas mesmo assim um alarme é gerado. Finalmente, pelo fato de cada pacote ser comparado com uma ampla coleção de assinaturas, o IDS fica pressionado com processamento e falha na detecção de muitos pacotes malignos.

Um IDS baseado em anomalias cria um perfil de tráfego enquanto observa o tráfego em operação normal. Ele procura então por cadeias de pacote que são estatisticamente incomuns, por exemplo, uma porcentagem irregular de pacotes ICMP ou um crescimento exponencial de scanners de porta e varreduras de ping. O mais interessante sobre sistemas IDS baseados em anomalias é que eles não recorrem a conhecimentos prévios de outros ataques — ou seja, eles podem detectar potencialmente novos ataques, que não foram documentados. Por outro lado, é um problema extremamente desafiador distinguir o tráfego normal de tráfegos estatisticamente incomuns. Até hoje, a maioria das implementações de IDS são primeiramente baseadas em assinaturas, apesar de algumas terem as funções de sistemas baseados em anomalias.

Snort

Snort é um código aberto IDS de domínio público com centenas de milhares de implementações existentes [Snort, 2007; Koziol, 2003]. Ele pode ser executado em plataformas Linux, UNIX e Windows. Ele usa uma interface libpcap de análise genérica, que também é usada pelo Wireshark e muitas outras ferramentas de análises de pacotes. Pode facilmente lidar com 100 Mbps de tráfego; para instalações com velocidades de tráfego de gigabit/seg, múltiplos sensores Snort serão necessários.

Para termos um vislumbre do Snort, vamos observar o exemplo de uma assinatura Snort:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any  
(msg:"ICMP PING NMAP"; dsiz: 0; itype: 8;)
```

Esta assinatura é compatível com qualquer pacote ICMP que entrar na rede da organização (\$HOME_NET) e que venha do exterior (\$EXTERNAL_NET), é do tipo 8 (ping ICMP), e tem uma carga útil vazia (dsiz=0). Já que o nmap (veja a Seção 1.6) gera pacotes ping com características específicas, essa assinatura é projetada para detectar varreduras de ping usadas pelo nmap. Quando um pacote é compatível com esta assinatura, o Snort gera um alerta que inclui a mensagem “ICMP PING NMAP”.

Talvez o que seja mais impressionante sobre o Snort é a vasta comunidade de usuários e experts em segurança que mantém sua base de dados de assinaturas. Normalmente, em algumas horas do novo ataque, a comunidade do Snort escreve e lança uma assinatura de ataque, que então é transferida via download por centenas de milhares de implementações de Snort ao redor do mundo. Além disso, ao usarem a sintaxe da assinatura do Snort, os administradores de rede podem criar suas próprias assinaturas a fim de satisfazer as necessidades da organização modificando assinaturas existentes ou criando novas.

8.9 Resumo

Neste capítulo, examinamos os diversos mecanismos que Bob e Alice, os amantes secretos, podem usar para se comunicar com segurança. Vimos que Bob e Alice estão interessados em confidencialidade (para que somente eles possam entender o conteúdo de uma mensagem transmitida), autenticação de entroncamento final (para que tenham a certeza de que estão falando um com o outro) e integridade de mensagem (para que tenham certeza de que suas mensagens não sejam alteradas em trânsito). É claro que a necessidade de comunicação segura não está limitada a amantes secretos. Na verdade, vimos nas seções 8.4 a e 8.7 que a segurança é necessária em várias camadas de uma arquitetura de rede para proteção contra bandidos que têm à mão um grande arsenal de ataques possíveis.

Na primeira parte deste capítulo, apresentamos vários princípios subjacentes à comunicação segura. Na Seção 8.2, examinamos as técnicas criptográficas para criptografar e decriptografar dados, incluindo criptografia de chaves simétricas e criptografia de chaves públicas. O DES e o RSA foram examinados como estudos de caso específicos dessas duas classes mais importantes de técnicas de criptografia em uso nas redes de hoje.

Na Seção 8.3, examinamos duas abordagens para fornecer a integridade da mensagem: códigos de autenticação de mensagem (MACs) e assinaturas digitais. As duas abordagens têm uma série de paralelos. Ambas usam funções hash criptografadas e ambas permitem que verifiquemos a fonte da mensagem, assim como a integridade da própria mensagem. Uma diferença importante é que os MACs não recorrem à criptografia, ao passo que as assinatura digitais necessitam de uma infraestrutura de chave pública. Ambas as técnicas são muito usadas na prática, como vimos na Seções 8.4 à 8.7. Além disso, as assinaturas digitais são usadas para criar certificados digitais, os quais são importantes para a verificação da validade de uma chave pública. Vimos também a autenticação de ponto final e como nonces podem ser usados para impedir ataques de repetição.

Nas Seções 8.4 a 8.7 examinamos diversos protocolos de segurança de rede que são usados extensivamente na prática. Vimos que uma criptografia de chave simétrica está no núcleo de PGP, SSL, Ipsec e segurança sem fio. Vimos que uma criptografia pública é crucial para ambos PGP e SSL. Estudamos que um PGP usa assinaturas eletrônicas para a integridade da mensagem, ao passo que SSL e Ipsec usam MACs. Agora que entendemos os princípios básicos da criptografia, e tendo estudado como esses princípios são usados, você agora deve estar pronto para projetar seus próprios protocolos de segurança de rede!

Munidos das técnicas abordadas nas seções 8.2 a 8.7, Bob e Alice podem se comunicar com segurança (esperamos que eles sejam estudantes de rede que aprenderam o que este livro ensinou e possam, dessa maneira, evitar que suas travessuras sejam descobertas por Trudy!) Porém, a confiabilidade é apenas uma pequena parte do quadro da segurança na rede. Como vimos na Seção 8.8, o foco da segurança em redes está se concentrando cada vez mais em garantir a segurança da infraestrutura da rede contra o ataque potencial dos bandidos. Assim, na última parte deste capítulo, estudamos firewalls e sistemas IDS que inspecionam pacotes que entram e saem uma organização de rede.

Este capítulo cobriu diversos fundamentos, enquanto focava nos tópicos mais importantes sobre a segurança em uma rede moderna. Os leitores que desejam mais informações podem investigar as referências citadas neste capítulo. Em particular, recomendamos [Skoudis, 2006] sobre ataques e segurança operacional, [Kaufmann, 1995] para criptografia e como aplicá-la em segurança de rede, [Rescorla, 2001] para uma explicação profunda, mas didática sobre SSL e [Edney, 2003] para uma discussão completa sobre segurança 802.11, incluindo uma investigação sobre a WEP e suas falhas. Os leitores também podem querer consultar [Ross, 2009] para um conjunto vasto de slides do Power Point (mais de 400) sobre segurança de rede e outros labs baseados em Linux.



Exercícios de fixação

Capítulo 8 Questões de revisão

Seção 8.1

1. Quais são as diferenças entre confidencialidade de mensagem e integridade de mensagem? É possível ter confidencialidade sem integridade? É possível ter integridade sem confidencialidade? Justifique sua resposta.
2. Equipamentos da Internet (roteadores, comutadores, servidores DNs, servidores da Web, usuários e sistemas etc.) frequentemente precisam se comunicar com segurança. Dê três exemplos específicos de pares de equipamentos da Internet que precisem de uma comunicação segura.

Seção 8.2

3. Da perspectiva de um serviço, qual é uma diferença importante entre uma sistema de chave simétrica e uma sistema de chave pública?
4. Suponha que um intruso tenha uma mensagem decodificado, bem como a versão decodificada dessa mensagem. Ele pode montar um ataque somente com texto cifrado, um ataque com texto aberto conhecido ou um ataque com texto aberto escolhido?
5. Considere uma cifra de 8 blocos. Quantos blocos de entrada possíveis uma cifra tem? Quantos mapeamentos possíveis podem existir? Se analisarmos cada mapeamento como uma chave, então quantas chaves essa cifra teria?
6. Suponha que N pessoas queiram se comunicar com cada uma das outras $N - 1$ pessoas usando criptografia de chaves simétricas. Todas as comunicações entre quaisquer duas pessoas, i e j , são visíveis para todas as outras pessoas desse grupo de N , e nenhuma outra pessoa desse grupo pode decodificar suas comunicações. O sistema, como um todo, requer quantas chaves? Agora, suponha que seja usada criptografia de chaves públicas. Quantas chaves serão necessárias nesse caso?
7. Suponha que $n = 10.000$, $a = 10.023$ e $b = 10.004$. Use uma identidade de aritmética modular para calcular $(a \times b) \bmod n$ de cabeça.
8. Suponha que você queira criptografar a mensagem 10101111 criptografando um número decimal que corresponda a essa mensagem. Qual seria esse número decimal?

Seção 8.3

9. De que maneira um hash fornece um melhor controle de integridade da mensagem do que uma soma de verificação (como a soma de verificação da Internet)?
10. Você pode decriptografar o hash de uma mensagem a fim de obter a mensagem original? Explique sua resposta.
11. Considere a variação de um algoritmo MAC (Figura 8.9), onde o transmissor envia $(m, H(m) + s)$, onde $H(m) + s$ é a concatenação de $H(m)$ e s . Essa variação é falha? Por que ou por que não?
12. O que significa afirmar que um documento é verificável, não falsificável e não repudiável?
13. De que modo um resumo de mensagem criptografado por chave pública proporciona uma assinatura digital melhor do que utilizar a mensagem criptografada com chave pública?
14. Suponha que o certifier.com crie um certificado para foo.com. Tipicamente, todo o certificado seria criptografado com a chave pública de certifier.com. Verdadeiro ou Falso?
15. Suponha que Alice tenha uma mensagem pronta para enviar para qualquer pessoa que perguntar. Milhares de pessoas querem ter a mensagem da Alice, mas cada uma quer ter certeza da integridade da mensagem. Nesse contexto, você acha que um esquema baseado em MAC ou baseado em assinatura digital é mais apropriado? Por quê?
16. Qual é a finalidade de um nonce em um protocolo de identificação de ponto final?
17. O que significa dizer que um nonce é um valor que só ocorre uma vez na vida? Na vida de quem?
18. O que é o ataque do homem do meio? Esse ataque pode ocorrer quando são usadas chaves simétricas?

Seções 8.4 – 8.7

19. Suponha que Bob receba uma mensagem PGP de Alice. Como o Bob sabe com certeza que Alice criou a mensagem (e não Trudy, por exemplo)? O PGP usa um MAC para integridade da mensagem?
20. Em um registro SSL, existe um campo para uma sequência de números SSL. Verdadeiro ou Falso?
21. Qual é a finalidade de um nonce em um protocolo de autenticação em uma apresentação SSL?

22. Suponha que uma sessão SSL utilize uma cifra bloqueada com CBC. Verdadeiro ou Falso: o servidor envia ao cliente o IV em aberto?
23. Suponha que Bob inicie uma conexão TCP com a Trudy, que está fingindo ser Alice. Durante a apresentação, Trudy envia para Bob um certificado da Alice. Em qual etapa do algoritmo de cumprimento SSL o Bob descobrirá que ele não está se comunicando com a Alice?
24. Considere uma cadeia de pacotes do Hospedeiro A ao Hospedeiro B usando IPsec. Tipicamente, um novo SA será estabelecido para cada pacote enviado na cadeia. Verdadeiro ou Falso?
25. Suponha que o TCP esteja sendo executado através de IPsec entre o centro de operações e a filial na Figura 8.29. Se o TCP retransmitir o mesmo pacote, então os dois pacotes correspondentes enviados por pacotes R1 terão o mesmo número sequencial no cabeçalho ESP. Verdadeiro ou Falso?
26. Um SA IKE e um SA IPsec são a mesma coisa? Verdadeiro ou Falso?
27. Considere um WEP para 802.11. Suponha que a informação seja 10101100 e a chave de fluxo seja 1111000. Qual o texto cifrado?
28. Em WEP, um IV é enviado em aberto em cada quadro. Verdadeiro ou Falso?

Seção 8.8

29. Um filtro de pacote de estado mantém duas estruturas de dados. Nomeie-as e brevemente descreva o que elas fazem.
30. Considere um filtro de pacote tradicional (sem estado). Esse filtro de pacote pode filtrar pacotes baseado em flag bits TCP assim como em outros campos de cabeçalho. Verdadeiro ou Falso?
31. Em um filtro de pacote tradicional, cada interface pode ter sua própria lista de controle de acesso. Verdadeiro ou Falso?
32. Por que uma aplicação de gateway trabalha juntamente com o filtro de roteador para ser eficaz?
33. IDSs baseados em assinaturas e IPSs inspecionam a carga útil de segmentos TCP e UDP. Verdadeiro ou Falso?



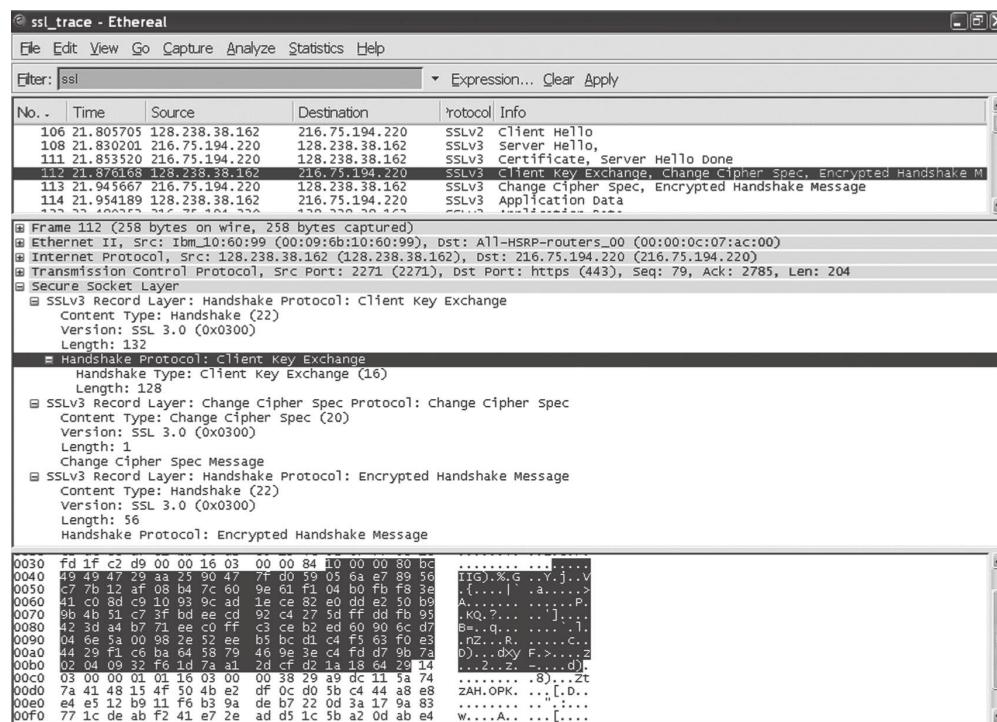
Problemas

1. Usando a cifra monoalfabética da Figura 8.3, codifique a mensagem "This is an easy problem." Decodifique a mensagem "rmij'u uamu xyj."
2. Mostre que o ataque com texto aberto conhecido de Trudy em que ela conhece os pares de tradução (texto cifrado, texto aberto) para sete letras reduz em aproximadamente 10^9 o número de possíveis substituições a verificar no exemplo apresentado na Seção 8.2.1.
3. Considere o sistema polialfabético mostrado na Figura 8.4. Um ataque com texto aberto escolhido que consiga obter a codificação da mensagem "The quick fox jumps over the lazy brown dog" é suficiente para decifrar todas as mensagens? Explique sua resposta.
4. Considere o bloco cifrado na Figura 8.5. Suponha que cada bloco cifrado T_i simplesmente invertam a ordem dos oito bits de entrada (então, por exemplo, 11110000 se torna 00001111). Além disso, suponha que o misturador de 64 bits não modifique qualquer bit (de modo que o valor de saída de m -ésimo bit seja igual ao valor de entrada do m -ésimo). (a) Sendo $n = 3$ e a entrada original de 64 bits igual a 10100000 repetidos oito vezes, qual o valor da saída? (b) Repita a parte (a), mas agora troque o último bit da entrada original de 64 bits de 0 para 1. (c) Repita as partes (a) e (b) mas agora suponha que o misturador de 64 bits inverte a ordem de 64 bits.
5. Considere o bloco cifrado da Figura 8.5. Para uma determinada "chave", Alice e Bob precisariam ter 8 tabelas, cada uma de 8 bits a 8 bits. Para a Alice (ou o Bob) armazenarem todas as oito tabelas, quantos bits de armazenamento são necessários? Como esse número se compara com o número de bits necessários para um bloco cifrado de tabela cheia de 64 bits?
6. Considere o bloco cifrado de 3 bits da Tabela 8.1. Suponha que o texto aberto seja 100100100. (a) Inicialmente suponha que o CBC não seja usado. Qual é o texto cifrado resultante? (b) Suponha que Trudy analise o texto cifrado. Supondo-se que ela saiba que um bloco cifrado de bits está sendo usado sem o CBC (mas ela não sabe a cifra específica), o que ela pode suspeitar? (c) Agora suponha que o CBC é usado com $IV = 111$. Qual o texto cifrado resultante?
7. Usando RSA, escolha $p = 3$ e $q = 11$ e codifique a palavra "dog", criptografando cada letra separadamente. (a) Aplique o algoritmo de decriptação à versão criptografada para recuperar a mensagem

- original em texto aberto. (b) Repita a parte (a), mas agora criptografe “dog” como uma mensagem m .
8. Considere RSA com $p = 5$ e $q = 11$.
 - a. Quais são n e z ?
 - b. Deixe que e seja 3. Por que esta é uma escolha aceitável para e ?
 - c. Encontre d para que $de \equiv 1 \pmod{z}$ e $d < 160$.
 - d. Criptografe a mensagem $m = 8$ usando a chave (n, e) . Deixe c denotar o texto cifrado correspondente. Mostre todo o processo. Dica: Para simplificar os cálculos, use como dados:
$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$
 9. Neste problema, exploraremos o algoritmo criptografado Diffie-Hellman (DH) de chave pública, o qual permite que duas entidades concordem em uma chave compartilhada. O algoritmo DH faz uso de um número primo grande p e outro número grande g , menor que p . Ambos, p e g , são públicos (de modo que um atacante os conheça). Em DH, Alice e Bob escolhem, independentemente, chaves secretas, S_A e S_B , respectivamente. Alice então calcula sua chave pública, T_A elevando g a S_A e tomando o mod p . Bob, similarmente, calcula sua própria chave pública T_B elevando g a S_B e tomando o mod p . Então, Alice e Bob trocam suas chaves públicas através da Internet. Alice calcula a chave secreta compartilhada S ao elevar T_B à S_A e tomando o mod p . Bob, similarmente, calcula a chave compartilhada S' ao elevar T_A a S_B e tomando o mod p .
 - a. Prove que no geral, Alice e Bob obtêm a mesma chave simétrica, ou seja, prove que $S = S'$.
 - b. Com $p = 11$ e $g = 2$, suponha que Alice e Bob escolham chaves privadas $S_A = 5$ e $S_B = 12$, respectivamente. Calcule as chaves públicas de Alice e Bob, T_A e T_B . Mostre todo o processo.
 - c. Seguindo a parte (b), calcule S como a chave simétrica compartilhada. Mostre todo o processo.
 - d. Forneça um diagrama de tempo que mostre como Diffie-Hellman pode ser atacado por um homem do meio. O diagrama de tempo deve ter três linhas verticais, uma para Alice, uma para Bob e uma para a atacante Trudy.
 10. Suponha que a Alice queira se comunicar com o Bob usando uma chave simétrica criptografada usando uma chave de sessão K_S . Na Seção 8.2, aprendemos que uma chave pública criptografada pode ser usada para distribuir a chave de sessão da Alice para o Bob. Neste problema, exploramos como uma chave de sessão pode ser distribuída — sem uma chave pública criptografada — usando um centro de distribuição de chaves (KDC, do inglês *key distribution center*). O KDC é um servidor que compartilha uma única chave simétrica secreta com cada usuário registrado. Para Alice e Bob, denote essas chaves K_{A-KDC} e K_{B-KDC} . Projete um esquema que use o KDC para distribuir K_S à Alice e Bob. Seu esquema deverá usar três mensagens para distribuir uma chave de sessão: a mensagem da Alice ao KDC; a mensagem do KDC à Alice; e finalmente a mensagem de Alice para Bob. A primeira mensagem é K_{A-KDC} (A, B). Usando a notação, $K_{A-KDC}, K_{B-KDC}, S, A$ e B responda às seguintes questões.
 - a. Qual a segunda mensagem?
 - b. Qual a terceira mensagem?
 11. Calcule uma terceira mensagem, diferente das duas mensagens na Figura 8.18, que tenha a mesma soma de verificação dessas duas mensagens na figura citada.
 12. Suponha que Alice e Bob compartilhem duas chaves secretas: uma chave de autenticação S_1 e uma chave criptografada simétrica S_2 . Aumente a Figura 8.9 para que ambas, integridade e confidencialidade, sejam fornecidas.
 13. No protocolo de distribuição BitTorrent de arquivo P2P (veja o Capítulo 2), a semente quebra o arquivo em blocos, e os pares redistribuem os blocos uns aos outros. Sem nenhuma proteção, um atacante pode facilmente causar destruição em um torrent ao se mascarar como se fosse um par benevolente e enviar blocos falsos aos outros conjuntos de pares no torrent. Esse pares que não são suspeitos redistribuem os blocos falsos aos outros pares, que distribuem os blocos falsos aos outros pares. Sendo assim, é importante para o BitTorrent ter um mecanismo que permita que os pares verifiquem a integridade de um bloco, para que não distribuam blocos falsos. Suponha que quando um par se junta a um torrent, inicialmente pega um arquivo .torrent de uma fonte inteiramente confiável. Escreva um esquema simples que permita que os pares verifiquem a integridade dos blocos.
 14. O protocolo de roteamento OSPF usa um MAC em vez de assinaturas digitais para fornecer a integridade da mensagem. Por que você acha que o MAC foi escolhido em vez de assinaturas digitais?
 15. Considere nosso protocolo de autenticação da Figura 8.16, no qual Alice se autentica para Bob, e que vimos que funciona bem (isto é, não encontramos nenhuma falha nele). Agora suponha que, enquanto Alice está se autenticando para Bob, este deve se autenticar para Alice. Apresente um cenário no qual Trudy, fazendo-se passar por Alice, agora pode se autenticar para Bob como se fosse Alice. (Dica: considere que a sequência de operações do protocolo, uma com Trudy iniciando a sequência e outra com Bob iniciando a sequência, pode ser intercalada arbitráriamente.)

- riamente. Preste particular atenção ao fato de que Bob e Alice usarão um nonce e que, caso não se tome cuidado, o mesmo nonce pode ser utilizado com más intenções.)
- 16.** No ataque do homem do meio mostrado na Figura 8.19, Alice não autenticou Bob. Se Alice tivesse solicitado a Bob que se autenticasse usando um protocolo de autenticação de chave pública, o ataque do homem do meio teria sido evitado? Exponha seu raciocínio.
- 17.** A Figura 8.20 mostra as operações que a Alice deve realizar com PGP para fornecer confidencialidade, autenticação e integridade. Faça um diagrama das operações correspondentes que Bob precisa executar no pacote recebido por Alice.
- 18.** Suponha que Alice queira enviar um e-mail a Bob. Bob tem um par de chave pública-privada (K_B^P, K_B^S), e a Alice tem o certificado do Bob. Mas a Alice não tem um par de chave pública, privada. Alice e Bob (e o mundo inteiro) compartilham a mesma função hash $H(\bullet)$.
- (a) Nessa situação, é possível projetar um esquema para que o Bob possa verificar que a Alice criou a mensagem? Se sim, mostre como, com um diagrama de bloco para Alice e Bob.
- (b) É possível projetar um esquema que forneça confidencialidade para enviar a mensagem da Alice a Bob? Se sim, mostre como com um diagrama de bloco para Alice e Bob.
- 19.** Considere a saída Wireshark abaixo para uma porção de sessão SSL.

- a. O pacote Wireshark 112 é enviado pelo cliente ou pelo servidor?
- b. Qual o número IP do servidor e seu número de porta?
- c. Considerando que nenhuma perda ou retransmissão ocorreram, qual será a sequência de números do próximo segmento TCP enviado pelo cliente?
- d. Quantos registros SSL existem no pacote Wireshark 112?
- e. O pacote 112 contém um Segredo Mestre ou uma Segredo Mestre Criptografado, ou nenhum?
- f. Supondo que o campo do tipo na apresentação é de 1 byte e cada campo de comprimento é de 3 bytes, quais são os valores do primeiro e do último bytes do Segredo Mestre (ou do Segredo Mestre Criptografado)?
- g. A mensagem de apresentação criptografado do cliente leva em conta o número de registros SSL?
- h. A mensagem de apresentação criptografada do servidor leva em conta o número de registros SSL?
- 20.** Na Seção 8.5.1 mostramos que sem os números de sequência, a Trudy (a mulher do meio) pode causar destruição em uma sessão SSL ao trocar os segmentos TCP. A Trudy pode fazer algo similar ao deletar um segmento TCP? O que ela precisa fazer para ter sucesso em seu ataque de apagamento? Quais serão os seus efeitos?



- 21.** Suponha que a Alice e o Bob estão se comunicando através de uma sessão SSL. Suponha que um atacante, que não tem nenhuma das chaves compartilhadas, insira um segmento TCP falso em fluxo de pacote com a correta soma de verificação TCP e números sequenciais (e corretos endereços IP e números de porta). A SSL do lado receptor aceitará o pacote falso e passará a carga útil à aplicação receptora? Por que ou por que não?
- 22.** As seguintes questões de Verdadeiro/Falso se referem à Figura 8.29.
- Quando um hospedeiro em 172.16.1/24 envia um datagrama ao servidor Amazon.com, o roteador R1 vai criptografar o datagrama usando IPsec.
 - Quando um hospedeiro em 172.16.1/24 envia um datagrama a um servidor 172.16.2/24, o roteador R1 mudará os endereços de origem e destino do datagrama IP.
 - Suponha que um hospedeiro 172.16.1/24 inicie uma conexão TCP com um servidor Web em 172.16.2/24. Como parte dessa conexão, todos os datagramas enviados pelo R1 terão o número de protocolo 50 no campo esquerdo do cabeçalho IPv4.
 - Considere o envio de um segmento TCP de um hospedeiro em 172.16.1/24 a um hospedeiro em 172.16.2/24. Suponha que o reconhecimento desse segmento se perde, então um TCP reenvia o segmento.
- 23.** Considere o exemplo na Figura 8.29. Suponha que Trudy é a mulher do meio, que insere datagramas no fluxo de datagramas indo de R1 a R2. Como parte do ataque de repetição, Trudy envia uma cópia duplicada de um dos datagramas enviados de R1 à R2. O R2 decriptografará o datagrama duplicado e o encaminhará à rede da filial? Se não, descreva em detalhes como R2 detecta o datagrama duplicado.
- 24.** Considere o seguinte pseudoprotocolo WEP. A chave é de 4 bits e o IV, de 2 bits. O IV é anexado ao final da chave quando está gerando o fluxo de chaves. Suponha que uma chave secreta compartilhada é 1010. O fluxo de chaves para quatro entradas possíveis são:
- 101000: 00101011010101001011010100100...
 101001: 1010011011001010110100100101101...
 101010: 0001101000111100010100101001111...
 101011: 111110101000000101010100010111...
- Suponha que todas as mensagens tenham 8 bits. Suponha que o ICV (controle de integridade) seja de 4 bits e calculado por OU-exclusivo nos primeiros 4 bits de dados com os últimos 4 bits de dados. Suponha que o pseudo-pacote WEP consista em três campos: primeiro o campo IV, depois o campo de mensagem e por último o campo ICV, com alguns desses campos criptografados.
- Queremos enviar a mensagem $m = 10100000$ usando $IV = 11$ e WEP. Qual serão os valores nos três campos WEP?
 - Mostre que quando o receptor decriptografa o pacote WEP, ele recupera a mensagem e o ICV.
 - Suponha que Trudy intercepte um pacote WEP (não necessariamente com $IV = 11$) e quer modificá-lo antes de encaminhá-lo ao receptor. Suponha que a Trudy gire o primeiro bit do ICV. Admitindo que a Trudy não sabe como o fluxo de chaves para quaisquer IVs, que outros bits a Trudy precisa gerar também, para que os pacotes recebidos passem pelo controle ICV?
 - Justifique sua resposta modificando os bits do pacote WEP da parte (a), decriptografando o pacote resultante e verificando o controle de integridade.
- 25.** Forneça uma tabela de filtro e uma tabela de conexão para um firewall de estado que seja tão restrito quanto possível, mas que efetue o seguinte:
- Permite que todos os usuários internos estabeleçam sessões Telnet com hospedeiros externos.
 - Permite que usuários externos naveguem pela página Web da empresa em 222.22.0.12.
 - Mas, em qualquer outro caso, bloqueia todo o tráfego interno e externo.
- A rede interna é 222.22/16. Na sua solução, suponha que a tabela de conexão esteja normalmente ocultando três conexões, de dentro para fora. Você precisará inventar um endereço IP e número de portas apropriados.
- 26.** Suponha que Alice queira visitar a página Web activist.com usando um serviço do tipo TOR. Esse serviço usa dois servidores proxy colaboradores, Proxy 1 e Proxy 2. Alice primeiro obtém os certificados (cada um contendo uma chave pública) para Proxy 1 e Proxy 2 de um servidor central. Denote $K_1^+()$, $K_1^-()$, $K_2^+()$ e $K_2^-()$ para a criptografia/decriptografia com chaves publica e privada RSA.
- Usando um diagrama de tempo, forneça um protocolo (o mais simples possível) que permita que Alice estabeleça uma sessão compartilhada de chave S_1 com Proxy 1. Denote $S_1(m)$ para a criptografia/decriptografia do dado m com a chave compartilhada S_1 .
 - Usando um diagrama de tempo, forneça um protocolo (o mais simples possível) que permita que Alice estabeleça uma sessão compartilhada da chave S_2 com Proxy 2, sem revelar seu endereço Ip ao Proxy 2.

- c. Suponha que as chaves compartilhadas S_1 e S_2 estejam determinadas. Usando um diagrama de tempo, forneça um protocolo (o mais simples possível e não usando uma chave criptografada pública) que permita que Alice requisite uma

página html de activist.com sem revelar seu endereço IP ao Proxy 2 e sem revelar à Proxy 1 qual site ela está visitando. Seu diagrama deve terminar com uma requisição HTTP chegando a activist.com.



Questões dissertativas

1. Suponha que um intruso consiga inserir e remover mensagens DNS da rede. Monte três cenários demonstrando os problemas que esse intruso poderia causar.
2. O que é Kerberos? Como funciona? Como ele pode ser relacionado ao Problema 10 deste capítulo?
3. Se o IPsec oferece segurança na camada de rede, por que ainda são necessários mecanismos de segurança em camadas acima do IP?
4. Faça uma pesquisa sobre os botnets. Que protocolos e sistemas os atacantes usam para controlar e atualizar os botnets hoje em dia?



Wireshark Lab

Neste lab (disponível na página Web), investigamos os protocolos de Camada Soquete Seguro (SSL, do inglês *Secure Socket Layer*). Lembre-se de que na Seção 8.5 o SSL é usado para a segurança de uma conexão TCP, e é extensivamente usado na prática para a segurança de transações pela Internet. Neste lab, focaremos os registros SSL enviados

através de uma conexão TCP. Tentaremos delinear e classificar cada um desses registros, focando no entendimento do porquê e como de cada registro. Investigamos os vários tipos de registro SSL assim como os campos nas mensagens SSL, analisando os passos dos registros SSL enviados entre seu hospedeiro e um servidor *e-commerce*.



IPsec Lab

Neste lab (disponível na página Web), exploraremos como criar SAs IPsec entre caixas Linux. Você pode fazer a primeira parte com duas caixas Linux simples, cada uma com um adaptador Ethernet. Mas na segunda parte do lab, você precisará de quatro ca-

xas Linux, duas tendo dois adaptadores Ethernet. Na segunda metade do lab, você criará SAs IPsec usando protocolos ESP no modo túnel. Primeiro você criará as SAs manualmente e depois fazendo o IKE criar as SAs.

Entrevista

Steven M. Bellovin

Steven M. Bellovin ingressou no corpo docente da Universidade de Columbia depois de muitos anos no AT&T (AT&T Fellow) do Network Services Research Lab nos AT&T Labs Research, em Florham Park, New Jersey. Ele trabalha especificamente com redes e segurança, e com as causas que as tornam incompatíveis. Em 1995, Steven recebeu o Usenix Lifetime Achievement Award por seu trabalho na criação da Usenet, a primeira rede de troca de informações que ligava dois ou mais computadores e permitia que os usuários compartilhassem informações e discutissem em conjunto. Steve também foi eleito membro da National Academy of Engineering. Fez mestrado na Columbia University e doutorado na University of North Carolina, em Chapel Hill.



O que o fez se decidir pela especialização na área de segurança de redes?

O que vou dizer parecerá estranho, mas a resposta é simples. Estudei programação de sistemas e administração de sistemas, o que levou naturalmente à segurança. E sempre me interessei por comunicações, desde a época em que trabalhava em tempo parcial em programação de sistemas, quando ainda estava na universidade.

Meu trabalho na área de segurança continua a ser motivado por duas coisas — um desejo de manter a utilidade dos computadores, o que significa impedir que sua função seja corrompida por atacantes, e um desejo de proteger a privacidade.

Qual era sua visão sobre a Usenet na época em que o senhor a estava desenvolvendo? Qual é sua visão agora?

No início considerávamos a Usenet como um meio para discutir ciência da computação e programação de computadores em todo o país e para utilizar em questões administrativas locais, como recurso de vendas, e assim por diante. Na verdade, minha previsão original era de uma ou duas mensagens por dia, de 50 a 100 sites no máximo. Mas o crescimento real ocorreu em tópicos relacionados a pessoas, incluindo — mas não se limitando a — interações de seres humanos com computadores. Meus grupos de bate-papo preferidos foram, durante muitos anos, coisas como rec.woodworking (marcenaria), bem como sci.crypt (criptografia).

Até certo ponto, as redes de notícias foram desbancadas pela Web. Se eu fosse iniciar o projeto hoje, ele seria muito diferente. Mas ainda é um excelente meio para alcançar um público muito amplo interessado no assunto, sem ter de passar por sites Web particulares.

Quais pessoas o inspiraram profissionalmente? De que modo?

O professor Fred Brooks — fundador e primeiro chefe do departamento de ciência da computação da University of North Carolina, em Chapel Hill, gerente da equipe que desenvolveu o IBM S/360 e o OS/360, e autor do livro *The Mythical Man-Month* — exerceu uma tremenda influência sobre minha carreira. Acima de tudo, ele me ensinou observação e compromissos — como considerar problemas no contexto do mundo real (e como o mundo real era muito mais confuso do que qualquer teórico gostaria que fosse), e como equilibrar interesses conflitantes ao elaborar uma solução. Grande parte do trabalho com computadores é engenharia — a arte de fazer os compromissos certos de modo a satisfazer muitos objetivos contraditórios.



Em sua opinião, qual é o futuro das redes e da segurança?

Até agora, grande parte da segurança que temos vem do isolamento. Um firewall, por exemplo, funciona impedindo o acesso a certas máquinas e serviços. Mas vivemos em uma época na qual a conectividade é cada vez maior — ficou mais difícil isolar coisas. Pior ainda, nossos sistemas de produção requerem um número muito maior de componentes isolados, interconectados por redes. Garantir a segurança de tudo isso é um de nossos maiores desafios.

Em que projetos especiais o senhor está trabalhando agora?

Estou me concentrando muito em questões operacionais. Não basta que um sistema seja seguro; ele também tem de ser utilizável. Isso significa, por sua vez, que operadores de redes têm de poder monitorar coisas. Mas monitorar pode conflitar com segurança — a segurança tende a ocultar e proteger computadores e informações, mas operadores precisam ver certas coisas. Além disso, sistemas têm de continuar funcionando mesmo que componentes cruciais falhem. Como compartilhar informações com os interessados corretos sem permitir a entrada dos bandidos? Essa é a pergunta que estou tentando responder agora.

Na sua opinião, têm havido grandes avanços na área de segurança? Até onde teremos de ir?

Ao menos do ponto de vista científico, sabemos como fazer criptografia. E isso é uma grande ajuda. Mas a maioria dos problemas de segurança se deve a códigos defeituosos e este é um problema muito mais sério. Na verdade, é o problema mais antigo da ciência da computação que ainda não foi resolvido — e acho que continuará sendo. O desafio é descobrir como manter a segurança de sistemas quando temos de construí-los com componentes insecurinhos. Hoje já podemos fazer isso no que tange às falhas de hardware; mas podemos fazer o mesmo em relação à segurança?

O senhor pode dar algum conselho aos estudantes sobre a Internet e segurança em redes?

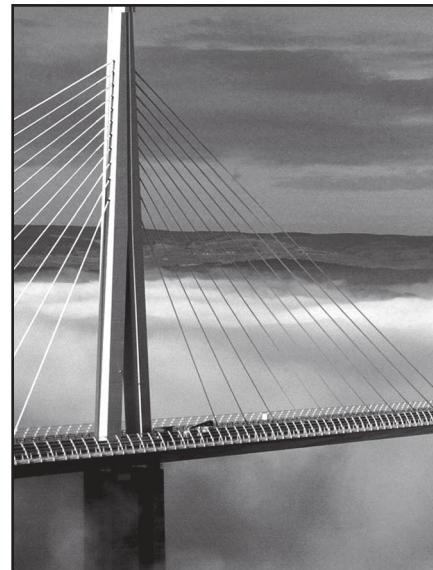
Aprender os mecanismos é a parte fácil. Aprender a “pensar como um paranoico” é mais difícil. Você tem de lembrar que as distribuições de probabilidade não se aplicam — os atacantes podem encontrar e encontrará condições improváveis. E os detalhes são importantes — e muito!





Capítulo 9

Gerenciamento de rede



Após termos percorrido nosso caminho pelos oito primeiros capítulos deste livro, estamos agora conscientes de que uma rede consiste em muitas peças complexas de hardware e software que interagem umas com as outras — desde os enlaces, comutadores, roteadores, hospedeiros e outros dispositivos, que são os componentes físicos da rede, até os muitos protocolos (tanto em hardware quanto em software) que controlam e coordenam esses componentes. Quando centenas ou milhares desses componentes são montados em conjunto por alguma organização para formar uma rede, não é nada surpreendente que ocasionalmente eles apresentem defeitos, que elementos da rede sejam mal configurados, que recursos da rede sejam utilizados excessivamente ou que componentes da rede simplesmente ‘quebrem’ (por exemplo, um cabo pode ser cortado, uma latinha de refrigerante pode ser derramada sobre um roteador). O administrador de rede, cuja tarefa é mantê-la ‘viva e atuante’, deve estar habilitado a reagir a esses contratemplos (e, melhor ainda, a evitá-los). Com potencialmente milhares de componentes de rede espalhados por uma grande área, ele, em sua central de operações (*network operations center* — NOC), evidentemente necessita de ferramentas que o auxiliem a monitorar, administrar e controlar a rede. Neste capítulo, examinaremos a arquitetura, os protocolos e as bases de informação que um administrador de rede utiliza para realizar seu trabalho.

9.1 O que é gerenciamento de rede?

Antes de discutir o gerenciamento de redes em si, vamos considerar alguns cenários ilustrativos do ‘mundo real’ que não são de redes, mas nos quais um sistema complexo com muitos componentes em interação deve ser monitorado, gerenciado e controlado por um administrador. As usinas de geração de energia elétrica têm uma sala de controle, onde mostradores, medidores e lâmpadas monitoram o estado (temperatura, pressão, vazão) de válvulas, tubulações, vasos e outros componentes remotos da instalação industrial. Esses dispositivos permitem que o operador monitore os muitos componentes da planta e possam alertá-lo (o famoso pisca-pisca vermelho de alerta) caso haja algum problema iminente. O operador responsável executa certas ações para controlar esses componentes. De maneira semelhante, a cabine de um avião é equipada com instrumentos para que o piloto possa monitorar e controlar os muitos componentes de uma aeronave. Nesses dois exemplos, o ‘administrador’ monitora equipamentos remotos e *analisa* os dados para garantir que os equipamentos estejam funcionando e operando dentro dos limites especificados (por exemplo, que a fusão do núcleo de uma usina nuclear não esteja na iminência de acontecer ou que o combustível do avião não esteja prestes a acabar), *controla reativamente* o

sistema fazendo ajustes de acordo com as modificações ocorridas no sistema ou em seu ambiente e gerencia proativamente o sistema (por exemplo, detectando tendências ou comportamentos anômalos que permitem executar uma ação antes que surjam problemas sérios). Nesse mesmo sentido, o administrador de rede vai monitorar, gerenciar e controlar ativamente o sistema do qual está encarregado.

Nos primórdios das redes de computadores, quando elas ainda eram artefatos de pesquisa, e não uma infraestrutura usada por milhões de pessoas por dia, ‘gerenciamento de rede’ era algo de que nunca se tinha ouvido falar. Se alguém descobrisse um problema na rede, poderia realizar alguns testes, como o *ping*, para localizar a fonte do problema e, em seguida, modificar os ajustes do sistema, reiniciar o software ou o hardware ou chamar um colega para fazer isso. (O RFC 789 traz uma discussão de fácil leitura sobre a primeira grande ‘queda’ da ARPAnet em 27 de outubro de 1980, muito antes da existência de ferramentas de gerenciamento de rede, e sobre os esforços realizados para entender os motivos dessa queda e recuperar a rede.) Como a Internet pública e as intranets privadas cresceram e se transformaram de pequenas redes em grandes infraestruturas globais, a necessidade de gerenciar mais sistematicamente a enorme quantidade de componentes de hardware e software dentro dessas redes também se tornou mais importante.

Com o intuito de motivar nosso estudo de gerenciamento de redes, vamos começar com um exemplo simples. A Figura 9.1 ilustra uma pequena rede constituída de três roteadores e alguns hospedeiros e servidores. Mesmo para uma rede tão simples, há muitos cenários em que o administrador de rede muito se beneficiará por ter à mão as ferramentas de gerenciamento adequadas:

Detecção de falha em uma placa de interface em um hospedeiro ou roteador. Com ferramentas de gerenciamento apropriadas, uma entidade de rede (por exemplo, o roteador A) pode indicar ao administrador de rede que uma de suas interfaces não está funcionando. (Isso certamente é melhor do que receber um telefonema, no NOC (centro de operações), de um usuário zangado dizendo que a conexão com a rede caiu!) Um administrador de rede que monitora e analisa de maneira ativa o tráfego da rede pode até impressionar o usuário (aquele, o zangado) detectando problemas na interface bem antes e substituindo a placa de interface antes que ela caia. Isso poderá ser feito, por exemplo, se o administrador notar um aumento de erros de somas de verificação em quadros que estão sendo enviados por uma placa de interface que está prestes a falhar.

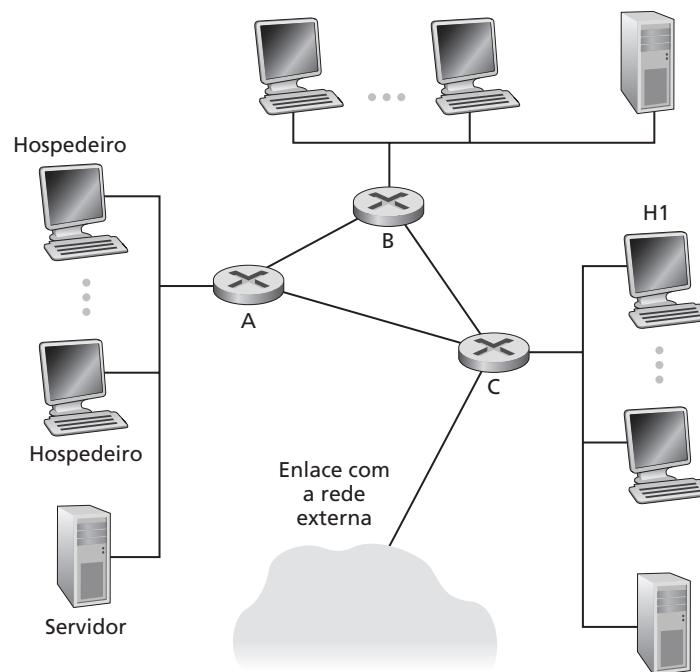


Figura 9.1 Um cenário simples que ilustra a utilização do gerenciamento de rede

Monitoração de hospedeiro. O administrador de rede pode verificar periodicamente se todos os hospedeiros da rede estão ativos e operacionais. Mais uma vez, ele pode, de fato, impressionar um usuário da rede, reagindo proativamente a um problema (falha em um hospedeiro) antes de o defeito ser apontado pelo usuário.

Monitoração de tráfego para auxiliar o oferecimento de recursos. Um administrador de rede pode monitorar padrões de tráfego entre fontes e destinos e notar, por exemplo, que, comutando servidores entre segmentos de LAN, o total de tráfego que passa por várias LANs poderia ser reduzido de maneira significativa. Imagine a felicidade geral com um desempenho melhor sem o custo de novos equipamentos. De modo similar, monitorando a utilização de um enlace, um administrador de rede pode determinar que um segmento de LAN ou o enlace com o mundo externo está sobrecarregado e que um enlace de maior largura de banda deve ser providenciado (ainda que com um custo mais alto). Ele também poderia ser alertado automaticamente quando o nível de congestionamento de um enlace ultrapassasse determinado limite, para providenciar um enlace de maior largura de banda antes que o congestionamento se tornasse sério.

Detecção de mudanças rápidas em tabelas de roteamento. A alternância de rotas — mudanças frequentes em tabelas de roteamento — pode indicar instabilidades no roteamento ou um roteador mal configurado. Evidentemente, um administrador de rede que configurou um roteador de modo inapropriado prefere ele mesmo descobrir o erro antes que a rede caia.

Monitoração de SLAs. Acordos de Nível de Serviços (Service Level Agreements — SLAs) são contratos que definem parâmetros específicos de medida e níveis aceitáveis de desempenho do provedor de rede em relação a essas medidas [Huston, 1999a]. Verizon e Sprint são apenas dois dos muitos provedores de rede que garantem SLAs [Verizon, 2009; Sprint, 2009] a seus usuários. Alguns desses SLAs são: disponibilidade de serviço (interrupção de serviços), latência, vazão e requisitos para notificação da ocorrência de serviço interrompido. É claro que, se um contrato especificar critérios de desempenho de prestação de serviços entre um provedor de rede e seus usuários, então a medição e o gerenciamento do desempenho do sistema também serão de grande importância para o administrador de rede.

Detecção de intrusos. Um administrador de rede provavelmente vai querer ser avisado quando chegar tráfego de uma fonte suspeita ou quando se destinar tráfego a ela (por exemplo, hospedeiro ou número de porta). Da mesma maneira, ele pode querer detectar (e, em muitos casos, filtrar) a existência de certos tipos de tráfego (por exemplo, pacotes roteados pela fonte ou um grande número de pacotes SYN dirigidos a um dado hospedeiro) que são característicos de determinados tipos de ataque à segurança que consideramos no Capítulo 8.

A International Organization for Standardization (ISO) criou um modelo de gerenciamento de rede que é útil para situar os cenários apresentados em um quadro mais estruturado. São definidas cinco áreas de gerenciamento de rede:

Gerenciamento de desempenho. A meta do gerenciamento de desempenho é quantificar, medir, informar, analisar e controlar o desempenho (por exemplo, utilização e vazão) de diferentes componentes da rede. Entre esses componentes estão dispositivos individuais (por exemplo, enlaces, roteadores e hospedeiros), bem como abstrações fim a fim, como um trajeto pela rede. Veremos, em breve, que padrões de protocolo como o SNMP (Simple Network Management Protocol — Protocolo Simples de Gerenciamento de Rede) [RFC 3410] desempenham um papel fundamental no gerenciamento de desempenho da Internet.

Gerenciamento de falhas. O objetivo do gerenciamento de falhas é registrar, detectar e reagir às condições de falha da rede. A linha divisória entre gerenciamento de falha e gerenciamento de desempenho é bastante indefinida. Podemos considerar o gerenciamento de falhas como o tratamento imediato de falhas transitórias da rede (por exemplo, interrupção de serviço em enlaces, hospedeiros, ou em hardware e software de roteadores), enquanto o gerenciamento de desempenho adota uma abordagem de longo prazo em relação ao desempenho da rede em face de demandas variáveis de tráfego e falhas.

ocasionais na rede. Como acontece no gerenciamento de desempenho, o SNMP tem um papel fundamental no gerenciamento de falhas.

Gerenciamento de configuração. O gerenciamento de configuração permite que um administrador de rede saiba quais dispositivos fazem parte da rede administrada e quais são suas configurações de hardware e software. O [RFC 3139] oferece uma visão geral de gerenciamento e requisitos de configuração para redes IP.

Gerenciamento de contabilização. O gerenciamento de contabilização permite que o administrador da rede especifique, registre e controle o acesso de usuários e dispositivos aos recursos da rede. Quotas de utilização, cobrança por utilização e alocação de acesso privilegiado a recursos fazem parte do gerenciamento de contabilização.

Gerenciamento de segurança. A meta do gerenciamento de segurança é controlar o acesso aos recursos da rede de acordo com alguma política definida. As centrais de distribuição de chaves e as autoridades certificadoras que estudamos na Seção 8.3 são componentes do gerenciamento de segurança. O uso de firewalls para monitorar e controlar pontos externos de acesso à rede, um tópico que estudamos na Seção 8.9, é outro componente crucial.

Neste capítulo, abordaremos apenas os rudimentos do gerenciamento de rede. Nossa foco é propositalmente limitado — examinaremos somente a infraestrutura do gerenciamento de rede: a arquitetura geral, os protocolos de gerenciamento de rede e a base de informações que servem para um administrador de rede mantê-la em pé e em funcionamento. Não abordaremos os processos de tomada de decisão do administrador de rede, que é quem deve planejar, analisar e reagir às informações gerenciais dirigidas à NOC. Nessa área estão alguns tópicos, como identificação e gerenciamento de falhas [Katzela, 1995; Medhi, 1997; Steinder, 2002], detecção proativa de anomalias [Thottan, 1998], correlação entre alarmes [Jakobson, 1993] e outros mais. Tampouco abordaremos o tópico mais amplo do gerenciamento de serviços [Saydam, 1996; RFC 3053; AT&T SLM, 2006] — o fornecimento de recursos, como largura de banda, capacidade do servidor e outros recursos computacionais e de comunicação necessários para cumprir os requisitos de serviço específicos da missão de uma empresa. Nessa última área, padrões como o TMN [Glitho, 1995; Sidor, 1998] e o TINA [Hamada, 1997], que são mais amplos, mais abrangentes (e alguns consideravelmente muito mais pesados), abordam essa questão mais geral.

“O que é gerenciamento de rede?” Essa pergunta é feita com frequência. Nossa discussão anterior expôs e ilustrou a necessidade de alguns usos do gerenciamento de rede. Concluiremos esta seção com uma definição de gerenciamento de rede em uma única sentença (embora um tanto longa e encadeada), dada por [Saydam, 1996]:

“Gerenciamento de rede inclui o oferecimento, a integração e a coordenação de elementos de hardware, software e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede, e de elementos, para satisfazer às exigências operacionais, de desempenho e de qualidade de serviço em tempo real a um custo razoável.”

É uma sentença de perder o fôlego, mas é uma definição viável. Nas seções seguintes, acrescentaremos um pouco de recheio a essa definição bastante enxuta de gerenciamento de rede.

9.2 A infraestrutura do gerenciamento de rede

Vimos, na seção anterior, que o gerenciamento de rede exige a capacidade de “monitorar, testar, consultar, configurar (...) e controlar” os componentes de hardware e software de uma rede. Como os dispositivos da rede são distribuídos, fazer isso exigirá, no mínimo, que o administrador possa coletar dados (por exemplo, para a finalidade de monitoração) de uma entidade remota e efetuar mudanças nessa entidade (por exemplo, controle). Nesta seção, uma analogia humana será útil para entender a infraestrutura necessária para gerenciamento de rede.

Imagine que você seja o chefe de uma grande organização que tem filiais em todo o mundo. É tarefa sua assegurar que as diversas partes de sua organização operem sem percalços. Como você o fará? No mínimo, você coletará dados periodicamente de suas filiais por meio de relatórios e de várias medições quantitativas de ativi-

dade, produtividade e orçamento. De vez em quando (mas não sempre), você será explicitamente notificado da existência de um problema em uma das filiais; o gerente da filial que quer galgar a escada corporativa (e talvez ficar com seu cargo) pode enviar relatórios não solicitados mostrando que as coisas estão correndo muito bem na filial a seu cargo. Você examinará os relatórios que receber esperando encontrar operações regulares em todos os lugares, mas, sem dúvida, encontrará problemas que precisarão de sua atenção. Você poderá iniciar um diálogo pessoal com uma de suas filiais problemáticas, coletar mais dados para entender o problema e, então, passar uma ordem executiva (“Faça essa mudança!”) ao gerente da filial.

Implícita nesse cenário humano muito comum está uma infraestrutura para controlar a organização — o chefe (você), os locais remotos que estão sendo controlados (as filiais), seus agentes remotos (os gerentes das filiais), protocolos de comunicação (para transmitir relatórios e dados padronizados e para diálogos pessoais) e dados (o conteúdo dos relatórios e as medições quantitativas de atividade, produtividade e orçamento). Cada um desses componentes do gerenciamento organizacional humano tem uma contrapartida no gerenciamento de rede.

A arquitetura de um sistema de gerenciamento de rede é conceitualmente idêntica a essa analogia simples com uma organização humana. O campo do gerenciamento de rede tem sua terminologia específica própria para os vários componentes de uma arquitetura de gerenciamento de rede; portanto, adotamos aqui essa terminologia. Como mostra a Figura 9.2, há três componentes principais em uma arquitetura de gerenciamento de rede: uma entidade gerenciadora (o chefe, na analogia apresentada), os dispositivos gerenciados (as filiais) e um protocolo de gerenciamento de rede.

A **entidade gerenciadora** é uma aplicação que em geral tem um ser humano no circuito e que é executada em uma estação central de gerência de rede na NOC. Ela é o lócus da atividade de gerenciamento de rede; ela controla a coleta, o processamento, a análise e/ou a apresentação de informações de gerenciamento de rede. É nela que são iniciadas ações para controlar o comportamento da rede e é aqui que o administrador humano interage com os dispositivos da rede.

Um **dispositivo gerenciado** é um equipamento de rede (incluindo seu software) que reside em uma rede gerenciada. Ele corresponde à filial de nossa analogia humana. Um dispositivo gerenciado pode ser um hospedeiro, um roteador, uma ponte, um hub, uma impressora ou um modem. No interior de um dispositivo gerenciado pode haver diversos **objetos gerenciados**. Estes são, na verdade, as peças de hardware propriamente ditas que estão dentro do dispositivo gerenciado (por exemplo, uma placa de interface de rede) e os conjuntos de parâmetros de configuração para as peças de hardware e software (por exemplo, um protocolo de roteamento intradomínio, como o RIP). Em nossa analogia humana, os objetos gerenciados podem ser os departamentos existentes na filial. Esses objetos gerenciados têm informações associadas a eles que são coletadas dentro de uma **Base de Informações de Gerenciamento** (Management Information Base — MIB); veremos que os valores dessas informações estão disponíveis para a entidade gerenciadora (e, em muitos casos, podem ser ajustados por ela). Em nossa analogia humana, a MIB corresponde aos dados quantitativos (medições de atividade, produtividade e orçamento, podendo este último ser estabelecido pela entidade gerenciadora!) que são trocados entre o escritório central e a filial. Estudaremos as MIBs detalhadamente na Seção 9.3. Por fim, reside também em cada dispositivo gerenciado um **agente de gerenciamento de rede**, um processo que é executado no dispositivo gerenciado, que se comunica com a entidade gerenciadora e que executa ações locais nos dispositivos gerenciados sob o comando e o controle da entidade gerenciadora. O agente de gerenciamento de rede é o gerente da filial em nossa analogia humana.

O terceiro componente de uma arquitetura de gerenciamento de rede é o **protocolo de gerenciamento de rede**. Esse protocolo é executado entre a entidade gerenciadora e o agente de gerenciamento de rede dos dispositivos gerenciados, o que permite que a entidade gerenciadora investigue o estado dos dispositivos gerenciados e, indiretamente, execute ações sobre eles mediante seus agentes. Agentes podem usar o protocolo de gerenciamento de rede para informar à entidade gerenciadora a ocorrência de eventos excepcionais (por exemplo, falhas de componentes ou violação de patamares de desempenho). É importante notar que o protocolo de gerenciamento de rede em si não gerencia a rede. Em vez disso, ele fornece uma ferramenta com a qual o administrador de rede pode gerenciar (“monitorar, testar, consultar, configurar, analisar, avaliar e controlar”) a rede. Essa é uma distinção sutil, mas importante.

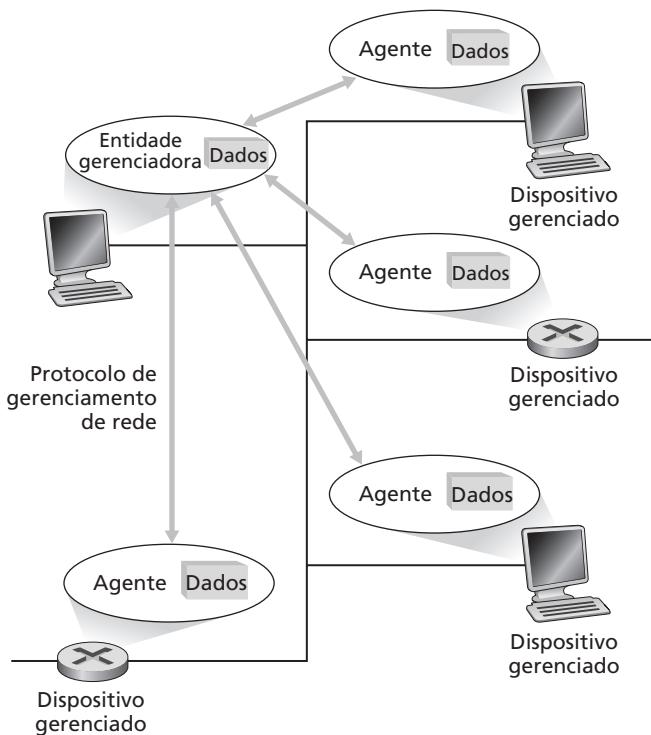


Figura 9.2 Principais componentes de uma arquitetura de gerenciamento de rede

Embora a infraestrutura do gerenciamento de rede seja conceitualmente simples, podemos nos atrapalhar com o vocabulário especial de gerenciamento de rede, como os termos ‘entidade gerenciadora’, ‘dispositivo gerenciado’, ‘agente de gerenciamento’ e ‘base de informações de gerenciamento’. Na terminologia do gerenciamento de rede, em nosso cenário simples de monitoração de hospedeiros, ‘agentes de gerenciamento’ localizados em ‘dispositivos gerenciados’ são periodicamente examinados pela ‘entidade gerenciadora’ — uma ideia simples, mas um problema linguístico! Mas, com um pouco de sorte, lembrar sempre da analogia com uma organização humana e seus óbvios paralelos com o gerenciamento de rede nos ajudará neste capítulo.

Nossa discussão anterior sobre arquitetura de gerenciamento de rede foi genérica e se aplica, em geral, a vários padrões e esforços de gerenciamento de rede que vêm sendo propostos há anos. Os padrões de gerenciamento de rede começaram a amadurecer no final da década de 1980, sendo que o OSI CMISE/CMIP (Common Management Service Element/Common Management Information Protocol — **Elemento de Serviço de Gerenciamento Comum/Protocolo de Informação de Gerenciamento Comum**) [Piscatello, 1993; Stallings, 1993; Glitho, 1998] e o SNMP (Simple Network Management Protocol — **Protocolo Simples de Gerenciamento de Rede**) da Internet [RFC 3410; Stallings, 1999; Rose, 1996] emergiram como os dois padrões mais importantes [Miller, 1997; Subramanian, 2000]. Ambos foram projetados para ser independentes de produtos ou de redes de fabricantes específicos. Como o SNMP foi projetado e oferecido rapidamente em uma época em que a necessidade de gerenciamento de rede começava a ficar premente, ele encontrou uma ampla aceitação. Hoje, esse protocolo é a estrutura de gerenciamento de rede mais amplamente usada e disseminada. Abordaremos o SNMP em detalhes na seção seguinte.



Princípios na prática

A central de operações de rede da SprintLink

Há redes de todos os tamanhos e formatos. Desde a menor das redes residenciais até o maior dos ISPs de nível 1, cabe ao operador (ou aos operadores) da rede garantir que sua operação seja tranquila. Mas o que acontece em uma central de operações de rede (NOC) e o que um operador de rede faz, na realidade?

Com uma rede que atravessa o mundo, a Sprint opera uma das maiores redes IP Tier 1 do mundo. Conhecida como SprintLink (mais informações em www.sprint.com), a rede possui mais de 70 pontos de presença (POPs) são locais que contêm roteadores IP SprintLink onde clientes se interconectam com a rede) e mais de 800 roteadores. É muita largura de banda! A principal NOC da SprintLink localiza-se em Reston, VA, e NOCs de suporte localizam-se na Flórida, Geórgia e Kansas City. A Sprint também mantém NOCs para sua rede de caixas eletrônicos, rede frame-relay e rede de transporte de fibra óptica subjacente. A qualquer tempo um grupo de quatro centrais operacionais está monitorando e gerenciando os equipamentos e o núcleo da rede IP SprintLink. Outro grupo está lidando com registros de clientes e respondendo às suas solicitações.

A automação — em monitoração (correlação de alarmes, identificação de falhas e restauração de serviços), gerenciamento de configuração e relatório de problemas dos clientes — possibilita que esse grupo extraordinariamente reduzido de operadores gerencie essa rede tão grande e complexa.

Quando ocorre um problema, a preocupação principal do operador da SprintLink é manter, ou restaurar rapidamente, o serviço oferecido aos clientes. Os operadores da NOC fazem triagem, diagnóstico e etapas de restauração seguindo um conjunto definido de procedimentos correspondentes a um conjunto conhecido de problemas. Problemas que não podem ser diagnosticados imediatamente, ou que não podem ser resolvidos por operadores dentro de um período de tempo específico que depende do nível de seriedade do problema (por exemplo, 15 minutos), são passados para membros da National Technical Assistance Center (NTAC) da Sprint, que oferecem suporte. Os técnicos da NTAC dedicam-se a estudar a fundo as causas dos problemas; eles também redigem procedimentos para a NOC e, quando necessário, trabalham com fornecedores de equipamentos (por exemplo, fabricantes de roteadores) para diagnosticar e resolver problemas relacionados a equipamentos específicos. Aproximadamente 90% dos problemas são tratados diretamente por técnicos e engenheiros da NOC. O pessoal da NOC e da NTAC também interage com outros grupos, incluindo parceiros NOCs (internos e externos) e grupos de campo da Sprint que são os 'olhos, as mãos e os ouvidos' dos POPs da Sprint.



Técnicos da Sprint monitoram a saúde da rede em centros de operação como o mostrado acima.

Como discutimos anteriormente neste capítulo, o ‘gerenciamento de rede’ na SprintLink (bem como em outros ISPs) evoluiu do gerenciamento de falhas para o gerenciamento de desempenho para o gerenciamento de serviços, com ênfase cada vez maior nas necessidades do cliente. Embora deem a máxima atenção às necessidades dos clientes, os operadores da Sprint também se orgulham muito de sua excelência operacional e do papel que desempenham na manutenção e na segurança da infraestrutura da rede global de um dos maiores ISPs do mundo.

9.3 A estrutura de gerenciamento padrão da Internet

Ao contrário do que o nome SNMP (protocolo simples de gerenciamento de rede) possa sugerir, o gerenciamento de rede na Internet é muito mais do que apenas um protocolo para transportar dados de gerenciamento entre uma entidade gerenciadora e seus agentes, e o SNMP passou a ser muito mais complexo do que a palavra ‘simples’ (o ‘S’) possa sugerir. As raízes da atual estrutura de gerenciamento padrão da Internet (Internet Standard Management Framework) remontam ao SGMP (Simple Gateway Monitoring Protocol — protocolo de monitoramento do gateway simples) [RFC 1028]. O SGMP foi projetado por um grupo de pesquisadores, usuários e administradores universitários de rede, cuja experiência com esse protocolo permitiu que eles projetassem, implementassem e oferecessem o SNMP em poucos meses [Lynch, 1993] — um feito muito distante dos processos de padronização atuais, que são bastante prolongados. Desde então, o SNMP evoluiu do SNMPv1 para o SNMPv2 e chegou à sua versão mais recente, o SNMPv3 [RFC 3410], lançada em abril de 1999 e atualizada em dezembro de 2002.

Na descrição de qualquer estrutura para gerenciamento de rede, certas questões devem inevitavelmente ser abordadas:

- O que está sendo monitorado (de um ponto de vista semântico)? E que tipo de controle pode ser exercido pelo administrador de rede?
- Qual é o modelo específico das informações que serão relatadas e/ou trocadas?
- Qual é o protocolo de comunicação para trocar essas informações?

Lembre-se de nossa analogia humana apresentada na seção anterior. O chefe e os gerentes das filiais precisarão acertar entre eles as medições de atividade, produtividade e orçamento que serão usadas para relatar a situação das filiais. De maneira semelhante, eles terão de concordar sobre que tipo de ações o chefe poderá realizar (por exemplo, cortar o orçamento, ordenar que o gerente da filial modifique alguma característica da operação do escritório ou demitir o pessoal e fechar a filial). Em um nível de maior detalhamento, eles precisarão concordar sobre o modo como esses dados serão relatados. Por exemplo, em que moeda (dólares, reais?) será apresentado o relatório de orçamento? Em que unidades será medida a produtividade? Embora talvez pareçam triviais, esses detalhes terão de ser acertados. Por fim, a maneira pela qual a informação trafegará entre o escritório central e as filiais (isto é, o protocolo de comunicação) deve ser especificada.

A estrutura de gerenciamento padrão da Internet aborda essas questões. A estrutura é constituída de quatro partes:

- Definições dos *objetos de gerenciamento de rede*, conhecidos como objetos MIB. Na Estrutura de Gerenciamento de Padrão da Internet, as informações de gerenciamento são representadas como uma coletânea de objetos gerenciados que, juntos, formam um banco virtual de informações virtuais conhecido como MIB. Um objeto MIB pode ser um contador, tal como o número de datagramas IP descartados em um roteador devido a erros em cabeçalhos de datagramas IP ou o número de erros de detecção de portadora em uma placa de interface Ethernet; um conjunto de informações descritivas, como a versão do software que está sendo executado em um servidor DNS; informações de estado, como se um determinado dispositivo está funcionando corretamente; ou informações específicas sobre protocolos, como um caminho de roteamento até um destino. Assim, os objetos MIB definem as informações de gerenciamento mantidas por um dispositivo gerenciado. Objetos MIB relacionados são reunidos em **módulos MIB**. Em nossa analogia com uma organização humana, a MIB define a informação transportada entre a filial e a sede.

Uma linguagem de definição de dados, conhecida como SMI (Structure of Management Information — Estrutura de Informação de Gerenciamento), que define os tipos de dados, um modelo de objeto e regras para escrever e revisar informações de gerenciamento. Objetos MIB são especificados nessa linguagem de definição de dados. Em nossa analogia humana, a SMI é usada para definir os detalhes do formato das informações que serão trocadas.

Um protocolo, SNMP. O SNMP é usado para transmitir informações e comandos entre uma entidade gerenciadora e um agente que os executa em nome da entidade dentro de um dispositivo de rede gerenciado.

Capacidades de segurança e de administração. A adição dessas capacidades representa o aprimoramento mais importante do SNMPv3 em comparação com o SNMPv2.

Assim, a arquitetura de gerenciamento da Internet é modular por projeto, com uma linguagem de definição de dados e de MIB independente de protocolo e um protocolo independente de MIB. O interessante é que essa arquitetura modular foi primeiramente criada para facilitar a transição de um gerenciamento de rede baseado em SNMP para uma estrutura de gerenciamento de rede que estava sendo desenvolvida pela ISO, que era a arquitetura de gerenciamento que concordaria com o SNMP quando este foi projetado — uma transição que nunca aconteceu. Com o tempo, contudo, a modularidade do projeto do SNMP permitiu que ele evoluísse mediante três importantes revisões, com cada uma das quatro partes do SNMP discutidas anteriormente se desenvolvendo independentemente. Ficou bem claro que a decisão pela modularidade foi correta, ainda que tenha sido tomada pela razão errada!

Nas subseções seguintes, examinaremos com mais detalhes os quatro componentes mais importantes da estrutura de gerenciamento padrão da Internet.

9.3.1 SMI (Estrutura de Informações de Gerenciamento)

A **Estrutura de Informações de Gerenciamento** (Structure of Management Information — SMI) (um componente da estrutura de gerenciamento de rede cujo nome é bastante estranho e não dá nenhuma pista quanto à sua funcionalidade) é a linguagem usada para definir as informações de gerenciamento que residem em uma entidade gerenciada de rede. Essa linguagem de definição é necessária para assegurar que a sintaxe e a semântica dos dados de gerenciamento de rede sejam bem definidas e não apresentem ambiguidade. Note que a SMI não define uma instância específica para os dados em uma entidade gerenciada de rede, mas a linguagem na qual a informação está especificada. Os documentos que descrevem a SMI para o SNMPv3 (confusamente denominada SMIv2) são [RFC 2578; RFC 2579; RFC 2580]. Vamos examinar a SMI de baixo para cima, começando com seus tipos de dados básicos. Em seguida, examinaremos como os objetos gerenciados são descritos em SMI e, então, como os objetos gerenciados relacionados entre si são agrupados em módulos.

Tipos de dados básicos da SMI

O RFC 2578 especifica os tipos de dados básicos da linguagem SMI de definição de módulos MIB. Embora a SMI seja baseada na linguagem de definição de objetos ASN.1 (Abstract Syntax Notation One — notação de sintaxe abstrata 1) [ISO, 1987; ISO X.680, 2002], tantos foram os tipos de dados específicos da SMI acrescentados que esta deve ser considerada uma linguagem de definição de dados por direito próprio. Os 11 tipos de dados básicos definidos no RFC 2578 são mostrados na Tabela 9.1. Além desses objetos escalares, também é possível impor uma estrutura tabular sobre um conjunto ordenado de objetos MIB usando a construção SEQUENCE OF; consulte o RFC 2578 para mais detalhes. Grande parte dos tipos de dados da Tabela 9.1 será familiar (ou auto-explicativa) para a maioria dos leitores. O único tipo de dado que discutiremos com mais detalhes será o OBJECT IDENTIFIER, que é usado para dar nome a um objeto.

Construções SMI de nível mais alto

Além dos tipos de dados básicos, a linguagem de definição de dados SMI também fornece construções de linguagem de nível mais alto.

Tipo de dado	Descrição
INTEGER	Número inteiro de 32 bits, como definido em ASN.1, com valor entre -2^{31} e $2^{31} - 1$, inclusive, ou um valor de uma lista de valores constantes possíveis, nomeados.
Integer32	Número inteiro de 32 bits, com valor entre -2^{31} e $2^{31} - 1$, inclusive.
Unsigned32	Número inteiro de 32 bits sem sinal na faixa de 0 a $2^{32} - 1$, inclusive.
OCTET STRING	Cadeia de bytes de formato ASN.1 que representa dados binários arbitrários ou de texto de até 65.535 bytes de comprimento.
OBJECT IDENTIFIER	Formato ASN.1 atribuído administrativamente (nome estruturado); veja a Seção 9.3.2.
Endereço IP	Endereço Internet de 32 bits, na ordem de bytes de rede.
Counter32	Contador de 32 bits que cresce de 0 a $2^{32} - 1$ e volta a 0.
Counter64	Contador de 64 bits.
Gauge32	Número inteiro de 32 bits que não faz contagens além de $2^{32} - 1$ nem diminui para menos do que 0.
TimeTicks	Tempo, medido em centésimos de segundo, transcorrido a partir de algum evento.
Opaque	Cadeia ASN.1 não interpretada, necessária para compatibilidade com versões anteriores.

Tabela 9.1 Tipos de dados básicos da SMI

A construção OBJECT-TYPE é utilizada para especificar o tipo de dado, o status e a semântica de um objeto gerenciado. Esses objetos gerenciados contêm, coletivamente, os dados de gerenciamento que estão no cerne do gerenciamento de rede. Há mais de dez mil objetos definidos em diversos RFCs da Internet [RFC 3410]. A construção OBJECT-TYPE tem quatro cláusulas. A cláusula SYNTAX de uma definição OBJECT-TYPE especifica o tipo de dado básico associado ao objeto. A cláusula MAX-ACCESS especifica se o objeto gerenciado pode ser lido, escrito, criado ou ter seu valor incluído em uma notificação. A cláusula STATUS indica se a definição do objeto é atual e válida, obsoleta (caso em que não deve ser implementada, pois sua definição está incluída por motivos históricos apenas) ou depreciada (obsoleta, mas implementável por causa de sua interoperabilidade com implementações mais antigas). A cláusula DESCRIPTION contém uma definição textual e legível do objeto; ela ‘documenta’ a finalidade do objeto gerenciado e deve fornecer todas as informações semânticas necessárias para implementá-lo.

Como exemplo de uma construção OBJECT-TYPE, considere a definição do tipo de objeto ipSystemStatsInDelivers do RFC 4293. Esse objeto define um contador de 32 bits que monitora o número de datagramas IP que foram recebidos no dispositivo gerenciado e entregues com sucesso a um protocolo de camada superior. A última linha dessa definição diz respeito ao nome desse objeto, um tópico que consideraremos na seção seguinte.

```
ipSystemStatsInDelivers OBJECT-TYPE
  SYNTAX          Counter32
  MAX-ACCESS     read-only
  STATUS          current
  DESCRIPTION    "The total number of input datagrams
                  successfully delivered to IPuser-protocols
                  (including ICMP)."
```

When tracking interface statistics, the counter of the interface to which these datagrams were addressed is incremented. This interface might not be the same as the input interface for some of the datagrams.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ipSystemStatsDiscontinuityTime.”

```
 ::= { ipSystemStatsEntry 18 }
```

A construção MODULE-IDENTITY permite que objetos relacionados entre si sejam agrupados, como conjunto, dentro de um ‘módulo’. Por exemplo, o RFC 4293 especifica o módulo MIB que define objetos gerenciados (incluindo ipSystemStatsInDelivers) para gerenciar implementações do IP e de seu protocolo associado, o ICMP. O RFC 4022 especifica o módulo MIB para TCP, e o RFC 4133 especifica o módulo MIB para UDP. O RFC 4502 define o módulo MIB para monitoração remota, RMON. Além de conter as definições OBJECT-TYPE dos objetos gerenciados dentro do módulo, a construção MODULE-IDENTITY contém cláusulas para documentar informações de contato do autor do módulo, a data da última atualização, um histórico de revisões e uma descrição textual do módulo. Como exemplo, considere o módulo de definição para gerenciamento do protocolo IP:

```
ipMIB MODULE-IDENTITY
LAST-UPDATED "200602020000Z"
ORGANIZATION "IETF IPv6 MIB Revision Team"
CONTACT-INFO
    "Editor:
     Shawn A. Routhier
     Interworking Labs
     108 Whispering Pines Dr. Suite 235
     Scotts Valley, CA 95066
     USA
     Email: <sar@iwl.com>"

DESCRIPTION:
    "The MIB module for managing IP and ICMP
     implementations, but excluding their
     management of IP routes."

Copyright © The Internet Society (2006).
This version of this MIB module is part of RFC 4293; see the
RFC itself for full legal notices.

REVISION          "200602020000Z"
DESCRIPTION
    "The IP version neutral revision with added IPv6 objects for
     ND, default routers, and router advertisements. As well as
     being the successor to RFC 2011, this MIB is also the successor
     to RFCs 2465 e 2466. Published as RFC 4293."

REVISION          "199411010000Z"
DESCRIPTION
    "A separate MIB module (IP-MIB) for IP and ICMP management
     objects. Published as RFC 2011."

REVISION          "199103310000Z"
DESCRIPTION
    "The initial revision of this MIB module was part of MIB-II,
     which was published as RFC 1213."
::= { mib-2 48}
```

A construção NOTIFICATION-TYPE é usada para especificar informações referentes a mensagens ‘SNMPv2-Trap’ e ‘Information Request’ (solicitação de informações) geradas por um agente ou por uma entidade gerenciadora; veja a Seção 9.3.3. Entre essas informações estão um texto de descrição (DESCRIPTION) que especifica quando tais mensagens devem ser enviadas, bem como uma lista de valores que devem ser incluídos na mensagem gerada; veja o RFC 2578 para mais detalhes. A construção MODULE-COMPLIANCE define o conjunto de objetos gerenciados dentro de um módulo que um agente deve implementar. A construção AGENT-CAPABILITIES especifica as capacidades dos agentes relativas às definições de notificação de objetos e de eventos.

9.3.2 Base de informações de gerenciamento: MIB

Com dissemos anteriormente, a **Base de Informações de Gerenciamento** (Management Information Base — MIB) pode ser imaginada como um banco virtual de informações que guarda objetos gerenciados cujos valores, coletivamente, refletem o ‘estado’ atual da rede. Esses valores podem ser consultados e/ou definidos por uma entidade gerenciadora por meio do envio de mensagens SNMP ao agente que está rodando em um dispositivo gerenciado em nome da entidade gerenciadora. Objetos gerenciados são especificados utilizando a construção OBJECT-TYPE da SMI discutida anteriormente e agrupados em **módulos MIB** utilizando a construção MODULE-IDENTITY.

A IETF tem estado muito atarefada com a padronização de módulos MIB associados a roteadores, hospedeiros e outros equipamentos de rede, o que inclui dados básicos de identificação sobre um determinado componente de hardware e informações de gerenciamento sobre as interfaces e os protocolos de dispositivos da rede. Até 2006 havia mais de duzentos módulos MIB baseados em padrões e um número ainda maior de módulos MIB especificados por fabricantes privados. Com todos esses padrões, a IETF precisava encontrar um modo de identificar e dar nome aos módulos padronizados, bem como aos objetos gerenciados específicos dentro de um módulo. Em vez de começar do nada, a IETF adotou uma estrutura padronizada de identificação de objetos (nomeação) que já tinha sido publicada pela ISO. Como acontece com muitas entidades dedicadas à padronização, a ISO tinha ‘grandes planos’ para sua estrutura padronizada de identificação de objetos — identificar todo e qualquer objeto padronizado possível (por exemplo, formato de dados, protocolo ou informação) em qualquer rede, independentemente das organizações dedicadas à padronização das redes (por exemplo, IETF, ISO, IEEE ou ANSI), do fabricante do equipamento ou do proprietário da rede. Um objetivo bem grandioso mesmo! A estrutura de identificação de objeto adotada pela ISO é parte da linguagem de definição de objetos ASN.1 [ISO X.680, 2002], que discutiremos na Seção 9.4. Os módulos MIB padronizados já têm seu próprio cantinho confortável dentro dessa estrutura de nomeação vastamente abrangente, como veremos a seguir.

Como mostra a Figura 9.3, pela estrutura de nomeação da ISO, os objetos são nomeados hierarquicamente. Note que cada ponto de ramo da árvore tem um nome e um número (entre parênteses); assim, qualquer ponto da árvore pode ser identificado pela sequência de nomes ou números que especificam o trajeto da raiz até aquele ponto da árvore de identificação. Um programa baseado na Web — divertido, mas incompleto e não oficial — que percorre parte da árvore de identificação de objetos (usando informações sobre os ramos oferecidas por voluntários) pode ser encontrado em [Alvestrand, 1997] e [France Telecom, 2006].

No topo da hierarquia estão a ISO e o ITU-T, as duas principais entidades de padronização que tratam da ASN.1, bem como um ramo para o esforço conjunto realizado por essas duas organizações. No ramo ISO da árvore, encontramos registros para todos os padrões ISO (1.0) e para os padrões emitidos por entidades padronizadoras de vários países membros da ISO (1.2). Embora não apareça na Figura 9.3, logo abaixo desse ramo da árvore (ISO/países membros, também conhecido como 1.2), encontrariam os Estados Unidos (1.2.840), embalhados nos quais encontrariam um número para os padrões IEEE e ANSI e para os padrões específicos de empresas privadas. Entre as empresas privadas, estão a RSA (1.2.840.11359) e a Microsoft (1.2.840.113556), sob a qual encontrariam os Microsoft File Formats (1.2.840.113556.4) para vários produtos da Microsoft, como o Word (1.2.840.113556.4.2). Mas estamos interessados em redes (e não nos arquivos do Microsoft Word), portanto, vamos voltar nossa atenção ao ramo denominado 1.3, os padrões emitidos por entidades reconhecidas pela ISO. Entre estas estão o Departamento de Defesa dos Estados Unidos (6) (sob o qual encontraremos os padrões

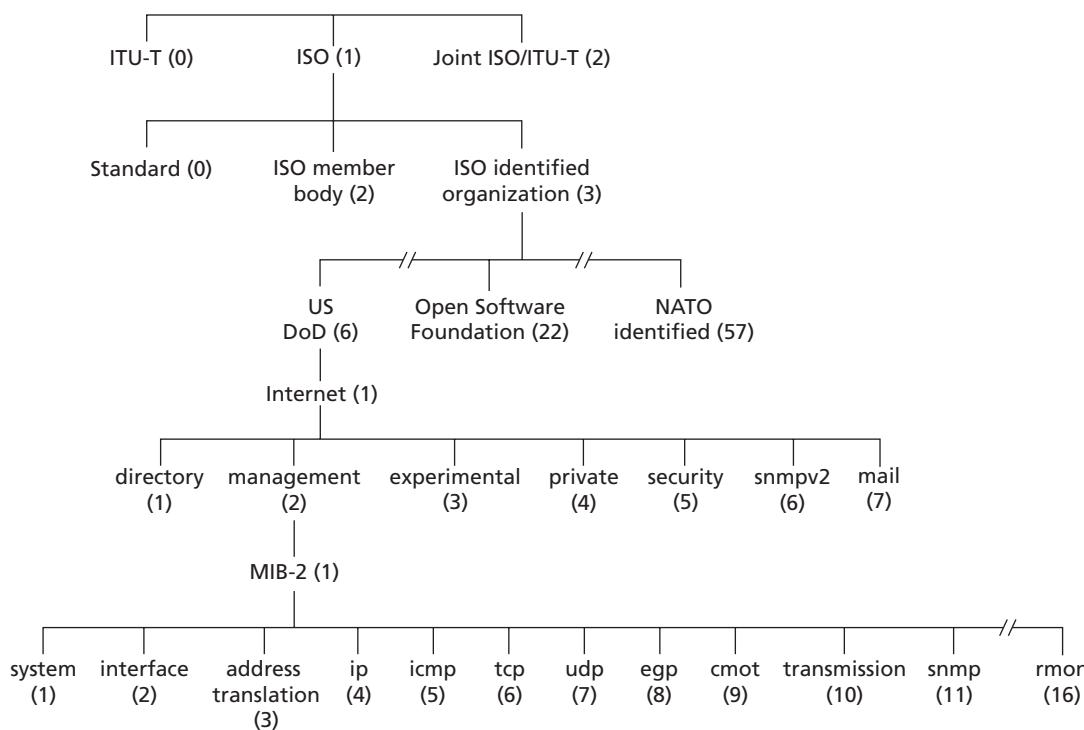


Figura 9.3 Árvore de identificadores de objetos ASN.1

da Internet), a Open Software Foundation (22), a associação de empresas aéreas SITA (69) e as entidades identificadas pela OTAN (57), bem como muitas outras organizações.

Sob o ramo Internet da árvore (1.3.6.1), há sete categorias. Sob o ramo private (1.3.6.1.4), encontramos uma lista [IANA, 2009b] dos nomes e códigos de empresas privadas para as mais de quatro mil empresas privadas que se registraram na Internet Assigned Numbers Authority (IANA) [IANA, 2009a]. Sob o ramo management (1.3.6.1.2) e MIB-2 (1.3.6.1.2.1) da árvore, encontramos a definição dos módulos MIB padronizados. Puxa! É uma longa jornada até chegarmos ao nosso cantinho no espaço de nomes da ISO!

Módulos MIB padronizados

O ramo mais baixo da árvore da Figura 9.3 mostra alguns dos módulos MIB orientados para hardware importantes (system e interface), bem como módulos associados a alguns dos protocolos mais importantes da Internet. O RFC 5000 relaciona todos os módulos MIB padronizados. Embora os RFCs referentes às MIBs sejam uma leitura tediosa e difícil, é muito instrutivo (isto é, assim como comer verduras, “é bom para você”) considerar algumas definições de módulos MIB para ter uma ideia do tipo de informação que existe dentro de um módulo.

Os objetos gerenciados que ficam sob o título system contêm informações gerais sobre o dispositivo que está sendo gerenciado; todos os dispositivos gerenciados devem suportar os objetos MIB do grupo system. A Tabela 9.2 define os objetos gerenciados no grupo system, de acordo com o RFC 1213. A Tabela 9.3 define os objetos gerenciados no módulo MIB para o protocolo UDP em uma entidade gerenciada.

9.3.3 Operações do protocolo SNMP e mapeamentos de transporte

O SNMPv2 [RFC 3416] é usado para transportar informações da MIB entre entidades gerenciadoras e agentes, executando em nome das entidades gerenciadoras. A utilização mais comum do SNMP é em um **modo comando-resposta**, no qual a entidade gerenciadora SNMPv2 envia uma requisição a um agente SNMPv2, que a

Identificador de objeto	Nome	Tipo	Descrição (segundo o RFC 1213)
1.3.6.1.2.1.1.1	sysDescr	OCTET STRING	"Nome completo e identificação da versão do tipo de hardware do sistema, do sistema operacional do software e do software de rede."
1.3.6.1.2.1.1.2	sysObjectID	OBJECT IDENTIFIER	ID atribuído pelo fabricante do objeto que "fornecce um meio fácil e não ambíguo para determinar 'que tipo de caixa' está sendo gerenciada."
1.3.6.1.2.1.1.3	sysUpTime	TimeTicks	"O tempo (em centésimos de segundo) desde que a porção de gerenciamento de rede do sistema foi reinicializada pela última vez."
1.3.6.1.2.1.1.4	sysContact	OCTET STRING	"A pessoa de contato para esse nó gerenciado, juntamente com a informação sobre como contatá-la."
1.3.6.1.2.1.1.5	sysName	OCTET STRING	"Um nome atribuído administrativamente para esse nó gerenciado. Por convenção, esse é o nome de domínio totalmente qualificado do nó."
1.3.6.1.2.1.1.6	sysLocation	OCTET STRING	"A localização física do nó."
1.3.6.1.2.1.1.7	sysServices	Integer32	Um valor codificado que indica o conjunto de serviços disponível no nó: aplicações físicas (por exemplo, um repetidor), de enlace/sub-rede (por exemplo, ponte), de Internet (por exemplo, gateway IP), fim a fim (por exemplo, hospedeiro).

Tabela 9.2 Objetos gerenciados no grupo system da MIB-2

recebe, realiza alguma ação e envia uma resposta à requisição. Em geral, uma requisição é usada para consultar (recuperar) ou modificar (definir) valores de objetos MIB associados a um dispositivo gerenciado. Um segundo uso comum do SNMP é para um agente enviar uma mensagem não solicitada, conhecida como **mensagem trap**, à entidade gerenciadora. As mensagens trap são usadas para notificar uma entidade gerenciadora de uma situação excepcional que resultou em mudança nos valores dos objetos MIB. Vimos anteriormente, na Seção 9.1, que o administrador de rede pode querer receber uma mensagem trap, por exemplo, quando uma interface cai, quando o congestionamento atinge um nível predefinido em um enlace ou quando ocorre qualquer outro evento notável. Observe que há uma série de compromissos importantes entre consulta de objetos (interação comando-resposta) e envio assíncrono de mensagens não solucionadas; veja os exercícios ao final deste capítulo.

O SNMPv2 define sete tipos de mensagens, conhecidas genericamente como PDUs (*protocol data units*), como mostra a Tabela 9.4. O formato da PDU é mostrado na Figura 9.4.

As PDUs GetRequest, GetNextRequest e GetBulkRequest são enviadas de uma entidade gerenciadora a um agente para requisitar o valor de um ou mais objetos MIB no dispositivo gerenciado do agente. Os identificadores de objeto dos objetos MIB cujos valores estão sendo requisitados são especificados na porção de vinculação de variáveis da PDU. GetRequest, GetNextRequest e GetBulkRequest diferem

Identificador de objeto	Nome	Tipo	Descrição (segundo o RFC 4113)
1.3.6.1.2.1.7.1	udpInDatagrams	Counter32	"número total de datagramas UDP entregues a usuários UDP"
1.3.6.1.2.1.7.2	udpNoPorts	Counter32	"número total de datagramas UDP recebidos para os quais não havia nenhuma aplicação na porta de destino"
1.3.6.1.2.1.7.3	udpInErrors	Counter32	"número de datagramas UDP recebidos que não puderam ser entregues por outras razões que não a falta de uma aplicação na porta de destino"
1.3.6.1.2.1.7.4	udpOutDatagrams	Counter32	"número total de datagramas UDP enviados dessa entidade"

Tabela 9.3 Objetos gerenciados no módulo MIB-2 udp

Tipo de SNMPv2-PDU	Remetente-receptor	Descrição
GetRequest	gerente a agente	pega o valor de uma ou mais instâncias de objetos MIB
GetNextRequest	gerente a agente	pega o valor da próxima instância de objeto MIB na lista ou tabela
GetBulkRequest	gerente a agente	pega valores em grandes blocos de dados, por exemplo, valores em uma grande tabela
InformRequest	gerente a gerente	informa à entidade gerenciadora remota valores da MIB que são remotos para seu acesso
SetRequest	gerente a agente	define valores de uma ou mais instâncias de objetos MIB
Response	agente a gerente ou gerente a gerente	gerada em resposta a GetRequest, GetNextRequest, GetBulkRequest, SetRequest PDU ou InformRequest
SNMPv2-Trap	agente a gerente	informa ao gerente um evento excepcional

Tabela 9.4 Tipos de SNMPv2-PDUs

no grau de especificidade de seus pedidos de dados. GetRequest pode requisitar um conjunto arbitrário de valores MIB; múltiplas GetNextRequest podem ser usadas para percorrer a sequência de uma lista ou tabela de objetos MIB, e GetBulkRequest permite que um grande bloco de dados seja devolvido, evitando a sobrecarga incorrida quando tiverem de ser enviadas múltiplas mensagens GetRequest ou GetNextRequest. Em todos os três casos, o agente responde com uma PDU Response que contém os identificadores de objetos e seus valores associados.

A PDU SetRequest é usada por uma entidade gerenciadora para estabelecer o valor de um ou mais objetos MIB em um dispositivo gerenciado. Um agente responde com uma PDU Response que contém uma mensagem Error Status ‘noError’ para confirmar que o valor na verdade foi estabelecido.

A PDU InformRequest é usada por uma entidade gerenciadora para comunicar a outra entidade gerenciadora informações MIB remotas à entidade receptora. A entidade receptora responde com uma PDU Response com a mensagem Error Status ‘noError’ para reconhecer o recebimento da PDU InformRequest.

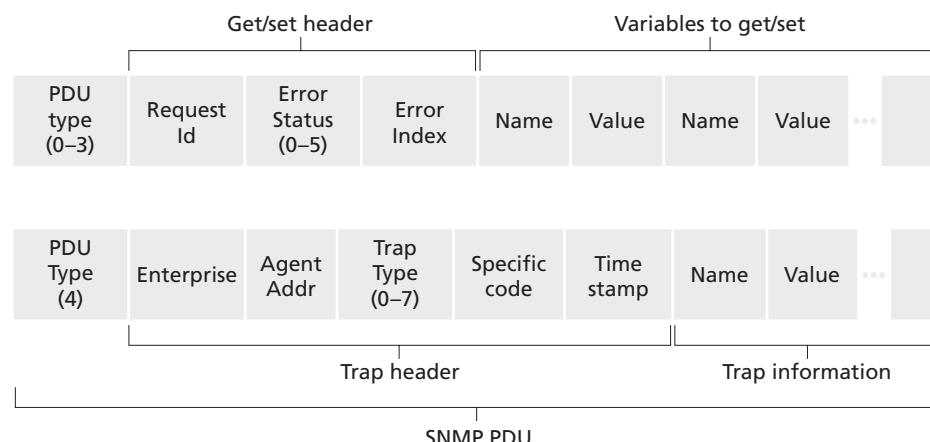


Figura 9.4 Formato da SNMP-PDU

O tipo final de SNMPv2-PDU é a mensagem trap. Mensagens trap são geradas assincronamente, isto é, não são geradas em resposta a uma requisição recebida, mas em resposta a um evento para o qual a entidade gerenciadora requer notificação. O RFC 3418 define tipos conhecidos de trap que incluem uma partida a frio ou a quente realizada por um dispositivo, a ativação ou interrupção de um enlace, a perda de um vizinho ou um evento de falha de autenticação. Uma requisição de trap recebida não exige resposta de uma entidade gerenciadora.

Dada a natureza comando-resposta do SNMPv2, convém observar que, embora as SNMP-PDUs possam ser transportadas por muitos protocolos de transporte diferentes, elas são tipicamente transportadas na carga útil de um datagrama UDP. Na verdade, o RFC 3417 estabelece que o UDP é o ‘mapeamento de transporte preferencial’. Uma vez que o UDP é um protocolo de transporte não confiável, não há garantia de que um comando ou sua resposta será recebido no destino pretendido. O campo Request Id da PDU é usado pela entidade gerenciadora para numerar as requisições que faz a um agente; a resposta de um agente adota a Request ID do comando recebida. Assim, o campo Request Id pode ser usado pela entidade gerenciadora para detectar comandos ou respostas perdidos. A entidade gerenciadora é quem decidirá se retransmitirá um comando se nenhuma resposta correspondente for recebida após um dado período de tempo. Em particular, o padrão SNMP não impõe nenhum procedimento específico de retransmissão, nem mesmo diz que o comando deve ser enviado, em primeiro lugar. Ele requer apenas que a entidade gerenciadora “aja com responsabilidade em relação à frequência e à duração das retransmissões”. Isso, é claro, nos leva a pensar como deve agir um protocolo ‘responsável’!

9.3.4 Segurança e administração

Os projetistas do SNMPv3 têm dito que o “SNMPv3 pode ser imaginado como um SNMPv2 com capacidades adicionais de segurança e de administração” [RFC 3410]. Certamente, há mudanças no SNMPv3 em relação ao SNMPv2, mas em nenhum lugar essas mudanças são mais evidentes do que nas áreas da administração e da segurança. O papel central da segurança no SNMPv3 era particularmente importante, já que a falta de segurança adequada resultava no uso do SNMP primordialmente para monitorar, em vez de controlar (por exemplo, SetRequest é raramente usada no SNMPv1).

À medida que o SNMP amadurecia, passando por três versões, sua funcionalidade crescia, mas infelizmente crescia também o número de documentos de padronização relacionados a ele. Isso fica evidenciado pelo fato de que há agora um RFC [RFC 3411] que “descreve uma arquitetura para descrever os Ambientes de Gerenciamento do SNMP”. Embora a ideia de uma ‘arquitetura’ para ‘descrever um ambiente’ possa ser um pouco excessiva para nossa cabeça, o objetivo do RFC 3411 é admirável — introduzir uma linguagem comum para descrever a funcionalidade e as ações executadas por um agente ou entidade gerenciadora SNMPv3. A arquitetura de uma entidade SNMPv3 é direta, e viajar por ela servirá para solidificar nosso entendimento do SNMP.

As denominadas **aplicações SNMP** consistem em um gerador de comandos, um receptor de notificações e um transmissor proxy (todos normalmente encontrados em uma entidade gerenciadora); um elemento respondedor de comandos e um originador de notificações (ambos tipicamente encontrados em um agente), e na possibilidade de outras aplicações. O gerador de comandos gera as PDUs GetRequest, GetNextRequest, GetBulkRequest e SetRequest, que examinamos na Seção 9.3.3, e processa as respostas recebidas dessas PDUs. O elemento respondedor de comandos executa em um agente e recebe, processa e responde (usando a mensagem Response) às PDUs GetRequest, GetNextRequest, GetBulkRequest e SetRequest recebidas. A aplicação originadora de notificações de um agente gera PDUs Trap; essas PDUs podem ser recebidas e processadas em uma aplicação receptora de notificações em uma entidade gerenciadora. A aplicação do transmissor proxy repassa as PDUs de requisição, notificação e resposta.

Uma PDU enviada por uma aplicação SNMP passa, em seguida, por um ‘processador’ SNMP, antes de ser enviada via protocolo de transporte apropriado. A Figura 9.5 mostra como uma PDU gerada pela aplicação geradora de comandos entra primeiramente no módulo de despacho, onde é determinada a versão do SNMP. A PDU é então processada no sistema de processamento de mensagens, no qual é envelopada em um cabeçalho de mensagem que contém o número da versão do SNMP, uma mensagem Id e informações sobre o tamanho da mensagem.

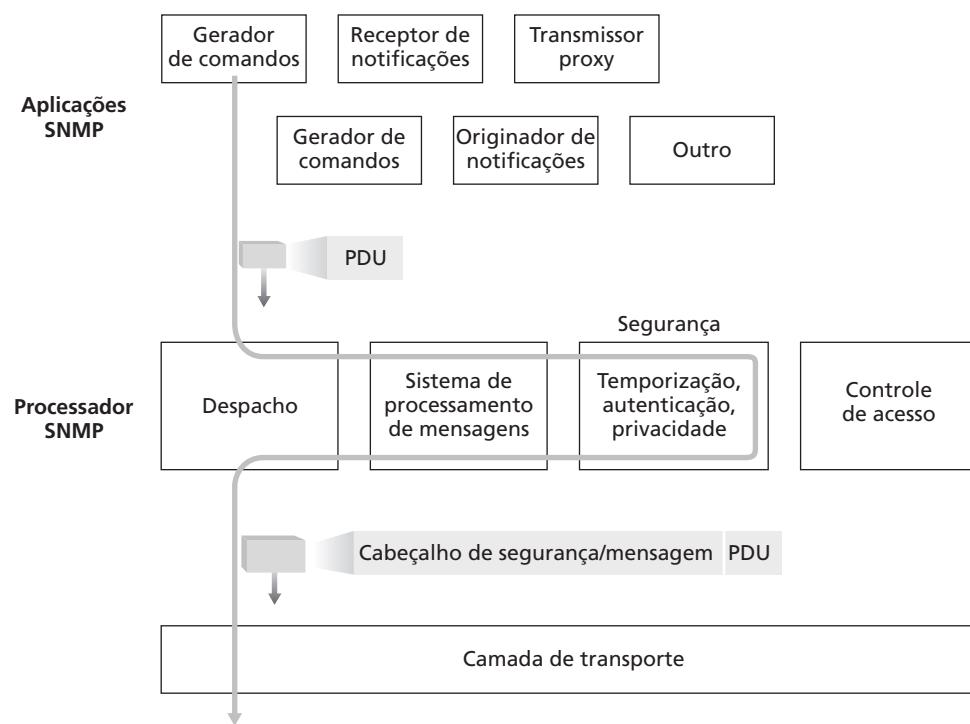


Figura 9.5 Processador e aplicações do SNMPv3

Caso seja necessária criptografia ou autenticação, são incluídos também os campos de cabeçalho apropriados para essas informações; veja o RFC 3411 para mais detalhes. Por fim, a mensagem SNMP (a PDU gerada pela aplicação e mais as informações do cabeçalho de mensagem) é passada ao protocolo de transporte apropriado. O protocolo de transporte preferencial para transportar mensagens SNMP é o UDP (isto é, as mensagens SNMP são transportadas como carga útil de um datagrama UDP), e o número de porta preferencial para o SNMP é a porta 161. A porta 162 é usada para mensagens trap.

Vimos anteriormente que as mensagens SNMP são usadas não somente para monitorar, mas também para controlar (por exemplo, por meio do comando SetRequest) elementos da rede. É claro que um intruso que conseguisse interceptar mensagens SNMP e/ou gerar seus próprios pacotes SNMP na infraestrutura de gerenciamento poderia criar um grande tumulto na rede. Assim, é crucial que mensagens SNMP sejam transmitidas com segurança. Surpreendentemente, foi somente na versão mais recente do SNMP que a segurança recebeu a atenção merecida. Sua segurança é conhecida como **segurança baseada no usuário** [RFC 3414], pois utiliza o conceito tradicional de um usuário, identificado por um nome de usuário, ao qual as informações de segurança — uma senha, um valor de chave ou acessos privilegiados — são associadas. O SNMPv3 fornece criptografia, autenticação, proteção contra ataques de reprodução (veja a seção 8.3) e controle de acesso.

Criptografia. As SNMP-PDUs podem ser criptografadas usando o DES no modo Encadeamento de Blocos de Cifras. Note que, uma vez que o DES é um sistema de chaves compartilhadas, o valor da chave secreta dos dados de codificação do usuário deve ser conhecido pela entidade receptora que deve decriptar os dados.

Autenticação. O SNMP usa a técnica de MAC que estudamos na Seção 8.3.1 para prover proteção e autenticação contra adulteração [RFC 2401]. Lembre-se de que um MAC necessita que um emissor e um receptor conheçam uma chave secreta comum.

Proteção contra ataques de reprodução. Lembre-se de que podem ser usados nonces (veja o Capítulo 8) para proteção contra ataques de reprodução. O SNMPv3 adota uma abordagem relacionada. Para garantir que uma mensagem recebida não seja uma reprodução de alguma mensagem anterior, o receptor exige que o remetente inclua em cada mensagem um valor baseado em um contador no *receptor*.

Esse contador, que funciona como um nonce, reflete o período de tempo decorrido entre a última reinicialização do software de gerenciamento de rede do remetente e o número total de reinicializações desde a última vez que o software de gerenciamento de rede do receptor foi configurado. Contanto que o contador em uma mensagem recebida esteja dentro de uma determinada margem de erro em relação ao próprio valor do receptor, a mensagem é aceita como uma mensagem original e a partir daí pode ser autenticada e/ou decriptada. Consulte o RFC 3414 para mais detalhes.

Controle de acesso. O SNMPv3 fornece um controle de acesso baseado em visões [RFC 3415] que controla quais das informações de gerenciamento de rede podem ser consultadas e/ou definidas por quais usuários. Uma entidade SNMP armazena informações sobre direitos de acesso e políticas em um banco de dados de configuração local (*Local Configuration Datastore* — LCD). Partes do LCD são acessíveis elas próprias como objetos gerenciados, definidos na MIB de Configuração do Modelo de Controle de Acesso Baseado em Visões (View-Based Access Control Model Configuration) [RFC 3415] e, portanto, podem ser gerenciados e manipulados remotamente via SNMP.

9.4 ASN.1

Neste livro, examinamos uma série de assuntos interessantes sobre redes de computadores. Esta seção sobre a ASN.1, contudo, pode não fazer parte da lista dos dez tópicos mais interessantes. Assim como as verduras, conhecer a ASN.1 e a questão mais ampla de serviços de apresentação é algo que “é bom para você”. A ASN.1 é um padrão originado na ISO, usado em uma série de protocolos relacionados à Internet, particularmente na área de gerenciamento de rede. Por exemplo, vimos na Seção 9.3 que as variáveis MIB do SNMP estão inextricavelmente ligadas à ASN.1. Portanto, embora o material sobre a ASN.1 apresentado nesta seção seja bastante árido, esperamos que o leitor nos dê um voto de confiança e acredite que o material é importante.

Para motivar nossa discussão, faça o seguinte exercício mental: suponha que fosse possível copiar dados, de maneira confiável, da memória de um computador diretamente para a memória de outro computador. Se isso fosse possível, o problema da comunicação estaria ‘resolvido’? A resposta a essa pergunta depende de como se define ‘problema de comunicação’. Com certeza, uma cópia perfeita, memória a memória, comunicaria com exatidão os bits e os bytes de uma máquina para outra. Mas essa cópia exata de bits e bytes significa que, quando o software que estiver sendo executado no computador receptor acessar esses dados, ele verá os mesmos valores que estavam armazenados na memória do computador remetente? A resposta a essa pergunta é “não necessariamente”! O nó da questão é que diferentes arquiteturas de computadores, diferentes sistemas operacionais e compiladores têm diferentes convenções de armazenamento e representação de dados. Quando se trata de comunicar e armazenar dados entre vários computadores (como acontece em todas as redes de comunicação), fica evidente que esse problema da representação de dados tem de ser resolvido.

Como exemplo desse problema, considere o fragmento simples em linguagem C a seguir. Como essa estrutura deveria ser disposta na memória?

```
struct {
    char code;
    int x;
} test;
test.x = 259;
test.code = 'a';
```

O lado esquerdo da Figura 9.6 mostra uma disposição possível para esses dados em uma arquitetura hipotética: há um único byte de memória que contém o caractere ‘a’, seguido de uma palavra de 16 bits que contém o valor inteiro 259, armazenado com o byte mais significativo antes. A disposição na memória de um outro computador é mostrada no lado direito da Figura 9.6. O caractere ‘a’ é seguido pelo valor inteiro armazenado com o byte menos significativo antes e com o inteiro de 16 bits alinhado para começar em uma fronteira de palavra de 16 bits. Certamente, se quiséssemos executar uma cópia ao pé da letra entre as memórias desses dois computa-



Princípios na prática

Há centenas (se não milhares) de produtos de gerenciamento de rede à disposição atualmente — todos incorporando até certo ponto os fundamentos de estrutura de gerenciamento de rede e SNMP que estudamos nesta seção. Um levantamento desses produtos está bem além do escopo deste livro e (sem dúvida) do âmbito da atenção do leitor. Assim, fornecemos aqui indicações de alguns dos produtos mais proeminentes. Um bom ponto de partida para se ter uma visão geral da abrangência das ferramentas de gerenciamento de rede é o Capítulo 12 do livro de [Subramanian, 2000].

As ferramentas de gerenciamento de rede podem ser divididas em duas classes amplas: as ferramentas dos fabricantes de equipamentos de rede, que são destinadas ao gerenciamento dos equipamentos desses fabricantes, e as ferramentas que visam ao gerenciamento de redes com equipamentos heterogêneos. Entre as ofertas específicas de fabricantes está o software de ferramentas de gerenciamento de rede Network Application Performance Analysis (NAPA) da Cisco criado em um dispositivo-base da Cisco [Cisco NAPA, 2009]. A Juniper oferece sistemas de suporte operacional (OSSs) para fornecimento de rede e suporte SLA/QoS [Juniper, 2009].

dores e usar a mesma definição de estrutura para acessar os valores armazenados, veríamos resultados diferentes nos dois computadores!

O fato de diferentes arquiteturas terem diferentes formatos para os dados internos é um problema real e universal. O problema específico do armazenamento de inteiros em diferentes formatos é tão comum que tem até um nome. A ordem de armazenamento de inteiros ‘big-endian’ (extremidade maior) armazena primeiramente os bytes mais significativos (nos endereços de armazenamento mais baixos). A ordem ‘little-endian’ (extremidade menor) armazena primeiramente os bytes menos significativos. Os processadores Sparc da Sun e os da Motorola são do tipo ‘big-endian’, ao passo que os processadores da Intel e da DEC/Compac Alpha são do tipo ‘little-endian’. Como curiosidade, os termos ‘big-endian’ e ‘little-endian’ vêm do livro *Viagens de Gulliver*, de Jonathan Swift, no qual dois grupos de pessoas insistem, dogmaticamente, em fazer uma coisa simples de duas maneiras diferentes (esperamos que a analogia com a comunidade da arquitetura de computadores fique clara). Um grupo da Terra de Lilliput insiste em quebrar os ovos pela extremidade maior (os ‘big-endians’), enquanto o outro grupo insiste em quebrá-los pela extremidade menor. Essa diferença foi a causa de um grande conflito e de uma rebelião civil.

Visto que diferentes computadores armazenam e representam dados de modo diferente, como os protocolos de rede devem enfrentar o problema? Por exemplo, se um agente SNMP estiver prestes a enviar uma mensagem Response que contém um número inteiro que representa a contagem do número de datagramas UDP recebidos, como ele deveria representar o valor inteiro a ser enviado à entidade gerenciadora — na ordem ‘big-endian’ ou na ordem ‘little-endian’? Uma alternativa seria o agente enviar os bytes do inteiro na mesma ordem em que serão armazenados na entidade gerenciadora. Outra alternativa seria o agente enviar o inteiro em sua própria ordem de armazenamento, deixando à entidade gerenciadora a responsabilidade de reordenar os bytes quando necessário. Qualquer uma das alternativas exigiria que o remetente ou o receptor soubesse qual o formato de representação de inteiros do outro.

Uma terceira alternativa é ter um método independente de máquina, de sistema operacional e de linguagem para descrever números inteiros e outros tipos de dados (isto é, uma linguagem de descrição de dados) e regras que estabeleçam a maneira como cada um desses tipos de dados deve ser transmitido pela rede. Quando forem recebidos dados de um determinado tipo, eles estarão em um formato conhecido e, assim, poderão ser armazena-

test.code	a	test.code	a
test.x	00000001	test.x	00000011
	00000011		00000001

Figura 9.6 Duas organizações diferentes de dados em duas arquiteturas diferentes.



Figura 9.7 O problema da apresentação

dos em qualquer formato específico que um dado computador exija. Tanto a SMI, que estudamos na Seção 9.3, quanto a ASN.1 adotam essa terceira alternativa. No jargão da ISO, esses dois padrões descrevem um **serviço de apresentação** — o serviço de transmitir e traduzir informações de um formato específico de uma máquina para outro. A Figura 9.7 ilustra um problema de apresentação no mundo real; nenhum dos receptores entende a ideia essencial que está sendo comunicada — que o interlocutor gosta de algo. Como mostrado na Figura 9.8, um serviço de apresentação pode resolver esse problema traduzindo a ideia para uma linguagem inteligível (pelo serviço de apresentação), independentemente de quem fala, enviando essa informação ao receptor e, em seguida, traduzindo-a para uma linguagem que o receptor entende.

A Tabela 9.5 mostra alguns tipos de dados definidos pela ASN.1. Lembre-se de que encontramos os tipos de dados INTEGER, OCTET STRING e OBJECT IDENTIFIER em nosso estudo anterior da SMI. Como nossa meta aqui (felizmente) não é fornecer uma introdução completa à ASN.1, remetemos o leitor aos padrões ou ao livro impresso e on-line de [Larmouth, 1996], para a descrição de tipos e construtores ASN.1, como SEQUENCE e SET, que permitem definir os tipos de dados estruturados.

Além de fornecer uma linguagem de descrição de dados, a ASN.1 oferece **Regras Básicas de Codificação** (Basic Encoding Rules — BERs), que especificam como instâncias de objetos que foram definidas usando a linguagem de descrição de dados ASN.1 devem ser enviadas pela rede. A BER adota a abordagem **TLV** (Type, Length, Value — **Tipo, Comprimento, Valor**) para a codificação de dados para transmissão. Para cada item de dados a ser remetido, são enviados o tipo dos dados, o comprimento do item de dados e o valor do item de dados, nessa ordem. Com essa simples convenção, os dados recebidos se autoidentificam.

A Figura 9.9 mostra como os dois itens de dados de um exemplo simples seriam enviados. Nesse exemplo, o remetente quer enviar a cadeia de caracteres ‘smith’ seguida de um valor decimal 259 (que é igual a 00000001 00000011 em linguagem binária ou a um valor de byte 1 seguido de um valor de byte 3) adotando a ordem

Tag	Tipo	Descrição
1	BOOLEAN	valor é ‘verdadeiro’ ou ‘falso’
2	INTEGER	pode ser arbitrariamente grande
3	BITSTRING	lista de um ou mais bits
4	OCTET STRING	lista de um ou mais bytes
5	NULL	sem valor
6	OBJECT IDENTIFIER	nome, na árvore de nomeação padrão ASN.1; veja a Seção 9.2.2
9	REAL	ponto flutuante

Tabela 9.5 Tipos de dados ASN.1 selecionados

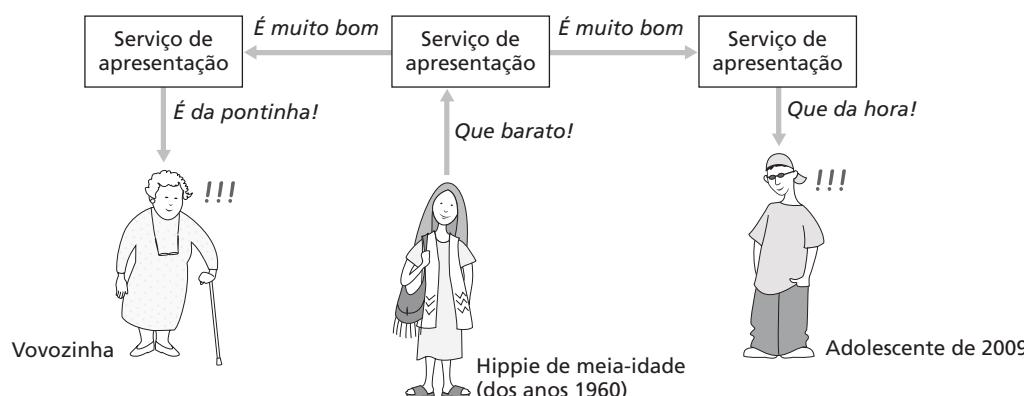


Figura 9.8 Resolução do problema da apresentação

'big-endian'. O primeiro byte da cadeia transmitida tem o valor 4, que indica que o tipo do item de dado seguinte é um OCTET STRING; esse é o 'T' do código TLV. O segundo byte da cadeia contém o comprimento do OCTET STRING, nesse caso, 5. O terceiro byte da cadeia transmitida inicia o OCTET STRING de comprimento 5; ele contém a representação ASCII da letra 's'. Os valores T, L e V dos dados seguintes são 2 (o tag do tipo INTEGER), 2 (isto é, um inteiro de comprimento de 2 bytes) e a representação 'big-endian' de 2 bytes do valor decimal 259.

Em nossa discussão, abordamos apenas um subconjunto pequeno e simples da ASN.1. Entre os recursos para aprender mais sobre a ASN.1 estão os documentos padronizados da ASN.1 [ISO X.680, 2002], o livro on-line de [Larmouth, 1996], relativo ao modelo OSI, e os sites Web relativo ao ASN.1 e [OSS, 2007] e [France Telecom 2006].

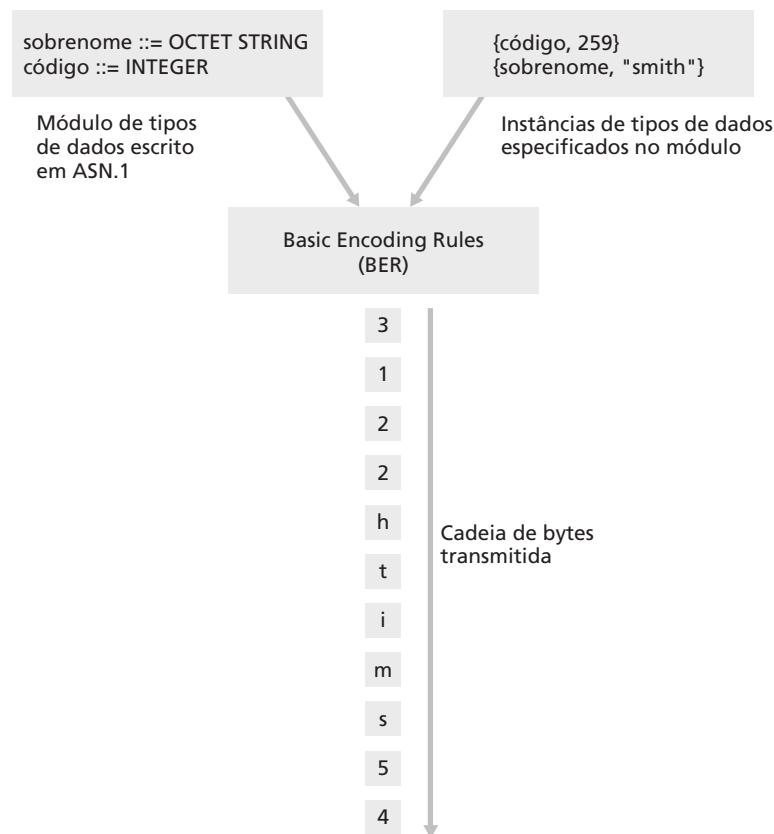


Figura 9.9 Exemplo de codificação BER

9.5 Conclusão

Nosso estudo sobre o gerenciamento de redes — e, na verdade, sobre toda a rede — agora está completo!

Neste capítulo final sobre gerenciamento de redes, começamos apresentando os motivos da necessidade de fornecer ferramentas adequadas ao administrador de rede — a pessoa que tem a tarefa de manter a rede ‘ligada e funcionando’ — para monitorar, testar, consultar, configurar, analisar, avaliar e controlar a operação da rede. Nossas analogias com a administração de sistemas complexos, como usinas elétricas, aviões e organizações humanas, ajudaram-nos a fornecer motivos para essa necessidade. Vimos que a arquitetura do sistema de gerenciamento de rede gira em torno de cinco componentes fundamentais: (1) um gerenciador de rede, (2) um conjunto de dispositivos gerenciados remotamente (pelo gerenciador de rede), (3) as bases de informações de gerenciamento (MIBs) existentes nesses dispositivos, contendo dados sobre seu estado e sua operação, (4) os agentes remotos que reportam informação das MIBs e executam ações sob o controle do gerenciador de rede e (5) um protocolo para a comunicação entre o gerenciador de rede e os dispositivos remotos.

Em seguida, examinamos em detalhes a estrutura de gerenciamento de rede da Internet e, em particular, o protocolo SNMP. Vimos como o SNMP apresenta os cinco componentes fundamentais de uma arquitetura de gerenciamento de padrão da Internet e gastamos um tempo considerável examinando objetos MIB, a SMI — a linguagem de definição de dados para especificação das MIBs — e o protocolo SNMP em si. Observamos que a SMI e a ASN.1 estão inextricavelmente interligadas e que a ASN.1 desempenha um papel fundamental na camada de apresentação do modelo de referência de sete camadas ISO/OSI, e então fizemos um estudo rápido da ASN.1. Talvez mais importante do que os detalhes da ASN.1 em si foi a necessidade percebida de fornecer a tradução entre formatos de dados específicos de cada computador de uma rede. Enquanto algumas arquiteturas de rede reconhecem explicitamente a importância desse serviço por terem uma camada de apresentação, essa camada não existe na pilha de protocolos da Internet.

Convém também observar que há muitos tópicos no gerenciamento de rede que preferimos *não* abordar — tópicos como as falhas de identificação e gerenciamento, a detecção proativa de anomalias, a correlação entre os alarmes e os itens mais amplos do gerenciamento de serviço (por exemplo, como oposto do gerenciamento de rede). Embora sejam importantes, esses tópicos merecem um livro dedicado a eles. O leitor pode consultar as referências apresentadas na Seção 9.1.



Exercícios de fixação

Capítulo 9 Questões de revisão

Seção 9.1

- Por que um administrador de rede necessita de ferramentas de gerenciamento de rede? Descreva cinco cenários.
- Quais são as cinco áreas de gerenciamento de rede definidas pela ISO?
- Qual a diferença entre gerenciamento de rede e gerenciamento de serviço?

Seção 9.2

- Defina os seguintes termos: entidade gerenciadora, dispositivo gerenciado, agente de gerenciamento de rede, MIB e protocolo de gerenciamento de rede.

Seção 9.3

- Qual é o papel da SMI no gerenciamento de rede?

- Cite uma diferença importante entre uma mensagem de comando-resposta e uma mensagem trap no SNMP.
- Quais são os sete tipos de mensagens usados no SNMP?
- O que significa um ‘processador do SNMP’?

Seção 9.4

- Qual é a finalidade da árvore de identificadores de objetos ASN.1?
- Qual é o papel da ASN.1 na camada de apresentação nos modelos de referência ISO/OSI?
- A Internet tem uma camada de apresentação? Se não tiver, como são tratadas as questões de diferenças entre arquiteturas de máquinas — por exemplo, a representação diferente de números inteiros em máquinas diferentes?
- O que significa codificação TLV?



Problemas

1. Considere as duas maneiras pelas quais ocorrem as comunicações entre uma entidade gerenciadora e um dispositivo gerenciado: modo comando-resposta e trapping. Quais são os prós e os contras dessas duas abordagens, em termos de (1) sobrecarga, (2) tempo de notificação quando ocorrem eventos excepcionais e (3) robustez quanto às mensagens perdidas entre a entidade gerenciadora e o dispositivo gerenciado?
2. Na Seção 9.3 vimos que era preferível transportar mensagens SNMP em datagramas UDP não confiáveis. Na sua opinião, por que os projetistas do SNMP preferiram o UDP ao TCP como protocolo de transporte para o SNMP?
3. Qual é o identificador de objeto ASN.1 para o protocolo ICMP (veja a Figura 9.3)?
4. Suponha que você trabalhe para uma empresa com sede nos Estados Unidos que quer desenvolver sua própria MIB para o gerenciamento de uma linha de produtos. Em que lugar da árvore de identificadores de objetos (veja a Figura 9.3) esse produto seria registrado? (Dica: você deve recorrer a alguns RFCs e a outros documentos similares para responder a essa pergunta.)
5. Lembre-se da Seção 9.3.2, que uma empresa privada (empreendimento) pode criar suas próprias variáveis MIB sob o ramo privado 1.3.6.1.4. Suponha que a IBM quisesse criar uma MIB para seu software do servidor Web. qual seria o próximo qualificador OID após 1.3.6.1.4? (Para responder a essa questão, você precisará consultar [IANA, 2009b]. Pesquise na Web para tentar descobrir se essa MIB existe para um servidor da IBM.
6. Na sua opinião, porque o comprimento precede o valor em uma codificação TLV (ao invés de vir após o valor)?
7. Considere a Figura 9.9. Qual seria a codificação BER para {peso, 276} {sobrenome, "Marco"}?
8. Considere a Figura 9.9. Qual seria a codificação BER para {peso, 160} {sobrenome, "Dario"}?



Questões dissertativas

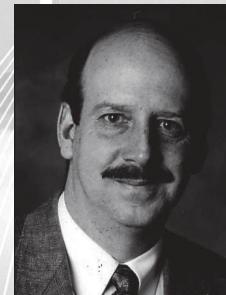
1. Além das analogias com uma usina elétrica e uma cabine de avião, cite uma analogia para um sistema complexo distribuído que precisa ser controlado.
2. Considere o cenário de motivação da Seção 9.1. Na sua opinião, que outras atividades um administrador de rede gostaria de monitorar? Por quê?
3. Leia o RFC 789. Como a queda da ARPAnet ocorrida em 1980 poderia ter sido evitada (ou sua recuperação simplificada) se seus administradores tivessem as ferramentas de gerenciamento de rede existentes hoje?



Entrevista

Jeff Case

Jeff Case é o fundador e diretor técnico da SNMP Research, Inc., empresa líder na produção de padrões da Internet e de produtos para gerenciamento de rede baseados em padrões. Ele é bacharel em educação industrial e engenharia elétrica pela Universidade de Purdue. É também doutor em educação técnica pela Universidade de Illinois, em Urbana-Champaign.



O que o fez se decidir pela especialização em tecnologia de rede?

Sempre fui fascinado por interligar coisas, desde quando era criança e enfiava grampos de cabelo nas tomadas. Essa fascinação se desenvolveu e se transformou em interesse por equipamentos de áudio durante a adolescência, quando eu montava sistemas de som para bandas de rock, cuja potência parecia ser suficiente para pulverizar concreto. Enquanto estudava e trabalhava como técnico de TV e rádio (em geral, equipamentos de áudio), fui mordido pelo bicho do computador e comecei a me interessar por tudo o que era digital, incluindo hardware e software para computadores. Comecei a me interessar pela construção de interfaces entre um componente esquisito e outro componente esquisito. Primeiro montei uma interface entre periféricos e processadores. Mais tarde, uma interface entre dois sistemas. A rede é a interface definitiva. E, até agora, a Internet é a rede definitiva.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Passei a maior parte de meus primeiros anos como profissional na Universidade de Purdue. Em determinados anos, dava aula para quase todos os cursos do currículo de graduação em eletricidade e em tecnologia da computação. Isso incluía criar novos cursos para os então emergentes tópicos de hardware e software para microprocessadores. Durante um semestre, a classe projetava um computador a partir de chips e o construía na parte laboratorial do curso, com uma equipe trabalhando na CPU, outra no subsistema de memória, outra no subsistema de E/S e assim por diante. No semestre seguinte, escrevíamos o software para o hardware que tínhamos construído.

Foi nessa mesma época que chefiei a área de computação do campus, reportando-me ao reitor como diretor de serviços acadêmicos de computação aos usuários.

Qual parte de seu trabalho lhe apresenta mais desafios?

Ficar atualizado com todas as mudanças, tanto técnicas quanto empresariais. Sou um administrador muito técnico, e está ficando cada vez mais difícil ficar atualizado com os avanços tecnológicos de nosso setor. Meu papel também exige que eu me informe sobre as mudanças no espaço empresarial, como fusões e aquisições.

Em sua opinião, qual é o futuro das redes e da Internet?

Mais, mais, mais. Mais velocidade. Mais ubiquidade. Mais conteúdo. Mais tensão entre anarquia e governança. Mais spam. Mais movimentos antispam. Mais problemas de segurança. Mais soluções de segurança. E, por fim, devemos esperar o inesperado.



Quais pessoas o inspiraram profissionalmente?

Meu falecido pai, que era um homem de negócios bem-sucedido; Dilbert; dr. Vint Cerf, dr. Jon Postel, dr. Marshall Rose e Chuck Davin, que são figuras muito conhecidas no setor da Internet; Bill Seifert, agora um sócio vice-presidente da empresa; dr. Rupert Evans, meu orientador; minha esposa, que trabalha comigo na empresa, e, finalmente, mas não menos importante, Jesus.

Li que o senhor tem uma coletânea notável de ‘provérbios’. Quando era professor de ciência da computação, quais deles costumava dizer a seus alunos?

“Um exemplo vale dois livros” (acho que é de Gauss).

“Às vezes, há uma lacuna entre teoria e prática. A lacuna entre teoria e prática não é tão grande teoricamente quanto a lacuna entre teoria e prática na prática” (não tenho ideia de onde veio esse provérbio).

Quais têm sido os maiores obstáculos na criação de padrões para a Internet?

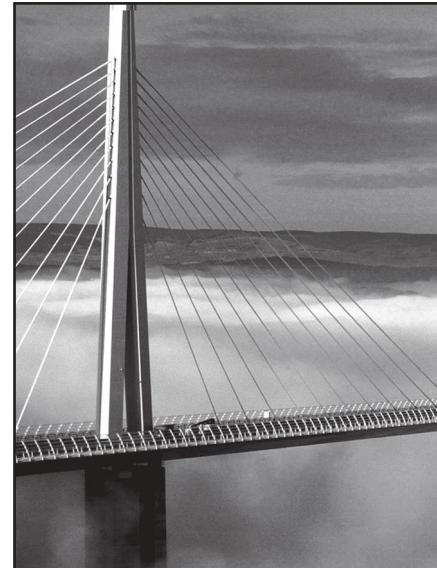
Dinheiro. Política. Egos. Fracassos de liderança.

Qual tem sido o uso mais surpreendente da tecnologia SNMP?

Todos. Na realidade, eu me envolvi no gerenciamento da Internet por causa das minhas necessidades de sobrevivência. Precisava ter algumas ferramentas adequadas para gerenciar a infraestrutura de rede da minha organização. O sucesso estrondoso que aconteceu foi obra do acaso, porque muitas outras pessoas precisavam resolver problemas semelhantes; foi também sorte, e muito trabalho árduo. O importante é que conseguimos montar a arquitetura correta logo de início.



Referências



Sempre que possível, informamos um URL como referência. Todos os URLs foram validados (e testados) em abril de 2004. Infelizmente, os URLs poderão estar desatualizados. Por favor, consulte a bibliografia atualizada na versão on-line deste livro em inglês (<http://www.awl.com/kurose-ross>).

Uma nota sobre os pedidos de comentários (*request for comments* — RFCs) da Internet: há uma cópia dos RFCs em diversos sites. Todos os URLs de RFCs a seguir levam ao arquivo RFC do Information Sciences Institute (ISI) mantido pelo Editor de RFC (RFC Editor) da Internet Society (a organização que supervisão os RFCs). Outros sites de RFC são: <http://www.faqs.org/rfcs> e <http://www.csl.sony.co.jp/fc/> (localizado no Japão).

Os RFCs da Internet podem ficar desatualizados com a publicação de outros RFCs. Aconselhamos que você acesse os sites relacionados anteriormente para obter informações mais atualizadas. A funcionalidade de busca dos RFCs no ISI (<http://www.rfc-editor.org/rfc.html>) permitirá que você procure um RFC e apresentará suas atualizações.

[3Com Addressing 2009] 3Com Corp., “White paper: Understanding IP addressing: Everything you ever wanted to know”, http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf.

[3GPP 2004] Third Generation Partnership Project homepage, <http://www.3gpp.org/>.

[802.11 Security 2009] The Unofficial 802.11 Security Web Page, <http://www.drizzle.com/~aboba/IEEE/>.

[Albitz 1993] P. Albitz e C. Liu. *DNS and BIND*. Petaluma: O'Reilly & Associates, Petaluma, CA, 1993.

[Abramson 1970] N. Abramson. “The Aloha System — Another Alternative for Computer Communications”, *Proc. 1970 Fall Joint Computer Conference, AFIPS Conference*, p. 37. 1970.

[Abramson 1985] N. Abramson. “Development of the Alohanet”, *IEEE Transactions on Information Theory*, vol. IT-31, n. 3 (mar. 1985), p. 119-123.

[Ahn 1995] J. S. Ahn, P. B. Danzig, Z. Liu e Y. Yan. “Experience with TCP Vegas: Emulation and Experiment”, *Proc. 1995 ACM SIGCOMM* (Boston, MA, ago. 1995), p. 185-195.

[Akamai 2009] Akamai homepage, <http://www.akamai.com>.

[Akella, 2003] A. Akella, S. Seshan, A. Shaikh. “An empirical Evaluation of Wide-area Internet Bottlenecks”, *Proc. 2003 ACM Internet Measurement Conf.* (Miami Flo, nov. 2003).

[Alvestrand 1997] H. Alvestrand. “Object Identifier Registry.” <http://www.alvestrand.no/objectid/>.

- [Anderson 1995]** J. B. Andersen, T. S. Rappaport, S. Yoshida. "Propagation Measurements and Models for Wireless Communication Channels", *IEEE Communications Magazine*, (jan. 1995), p. 42-49.
- [Aperjis 2008]** C. Aperjis, M. J. Freedman, R. Johari. "Peer-assisted Content Distribution with Prices", *Proc. ACM CoNEXT'08*, (Madri, dez. 2008).
- [Appenzeller 2004]** G. Appenzeller, I. Keslassy, N. Mc Keown. "Sizing Router Buffers", *Proc. 2004 ACM SIGCOMM* (Portland, OR, ago. 2004).
- [ARIN 1996]** ARIN. "IP allocation report", ftp://rs.arin.net/netinfo/ip_network_allocations.
- [Ash 1998]** G.R. Ash. *Dynamic Routing in Telecommunications Networks*, McGraw Hill: Nova York, 1998.
- [ASO-ICANN 2004]** The Address Supporting Organization home page, <http://www.aso.icann.org>.
- [AT&T SLM 2008]** AT&T Business. "AT&T Enterprise Hosting Services Service Guide", http://www.att.com/abs/serviceguide/docs/eh_sg.pdf.
- [Atheros 2009]** Atheros Communications Inc. "Atheros AR5006 WLAN Chipset Product Bulletins", <http://www.atheros.com/pt/AR5006Bulletins.htm>.
- [Ayanoglu 1995]** E. Ayanoglu, S. Paul, T. F. La Porta, K. K. Sabnani e R. D. Gitlin. "AIRMAIL: A Link-Layer Protocol for Wireless Networks", *ACM ACM/Baltzer Wireless Networks Journal*, 1:47-60, fev. 1995.
- [Bakre 1995]** A. Bakre e B. R. Badrinath. "I-TCP: Indirect TCP for Mobile Hosts", *Proc 1995. Int. Conf. on Distributed Computing Systems (ICDCS)*, maio 1995, p. 136-143.
- [Balakrishnan 1997]** H. Balakrishnan, V. Padmanabhan, S. Seshan e R. Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Transactions on Networking* 5, n. 6, dez. 1997.
- [Baran 1964]** P. Baran. "On Distributed Communication Networks", *IEEE Transactions on Communications Systems*, mar. 1964. Rand Corporation Technical report with the same title (Memorandum RM-3420-PR, 1964). <http://www.rand.org/publications/RM/RM3420/>.
- [Bardwell 2004]** J. Bardwell. "You Believe You Understand What You Think I Said... The Truth About 802.11 Signal and Noise Metrics: A Discussion Clarifying Often-Misused 802.11 WLAN Terminologies", http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf.
- [Baset 2006]** S. A. Basset e H. Schulzrinne. "An analysis of the Skype peer-to-peer Internet Telephony Protocol", *Proc. 2006 IEEE Infocom*. Barcelona, abr. 2006.
- [BBC 2001]** BBC news online. "A Small Slice of Design", abr. 2001. <http://news.bbc.co.uk/2/hi/science/nature/1264205.stm>.
- [BBC Multicast 2009]** BB, "BBC Multicast Trial" <http://support.bbc.co.uk/multicast/>.
- [Beheshti 2008]** N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, G. Salmon. "Experimental Study of Router Buffer Sizing", *Proc. ACM Internet Measurement Conference*, out. 2008, Vouliagmeni.
- [Bender 2000]** P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A. Viterbi. "CDMA/HDR: A bandwidth-efficient high-speed wireless data service for nomadic users", *IEEE Commun. Mag.*, vol. 38, n. 7, jul. 2000, p. 70-77.
- [Berners-Lee 1989]** T. Berners-Lee, CERN. "Information Management: A Proposal", mar. 1989, maio 1990. <http://www.w3.org/History/1989/proposal.html>
- [Berners-Lee 1994]** T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen e A. Secret. "The World-Wide Web", *Communications of the ACM*, vol. 37, n. 8, ago. 1994, p. 76-82.
- [Bernstein 2009]** D. Bernstein. "SYN Cookies", <http://cr.yp.to/syncookies.html>.
- [Bertsekas 1991]** D. Bertsekas e R. Gallagher. *Data Networks*, 2 ed. Englewood Cliffs: Prentice Hall, 1991.
- [Bhimani 1996]** Anish Bhimani. "Securing the Commercial Internet", *Communications of the ACM*, vol. 39, n. 6, mar. 1996, p. 29-35.
- [Biddle 2003]** P. Biddle, P. England, M. Peinado e B. Willman. "The Darknet and the Future of Content Distribution", *2002 ACM Workshop on Digital Rights Management*, nov. 2002, Washington, D.C. <http://crypto.stanford.edu/DRM2002/darknet5.doc>.

- [Biersack 1992]** E.W. Biersack. "Performance evaluation of forward error correction in ATM networks", *Proc. 1999 ACM SIGCOMM*. Baltimore, ago. 1992, p. 248-257.
- [BIND 2009]** Internet Software Consortium page on BIND, <http://www.isc.org/bind.html>.
- [Bisdikian 2001]** C. Bisdikian. "An Overview of the Bluetooth Wireless Technology", *IEEE Communications Magazine*, n. 12, dez. 2001, p. 86-94.
- [Bishop 2003]** M. Bishop. *Computer Security: Art and Science*. Boston: Addison Wesley, 2003.
- [BitTorrent 2009]** BitTorrent.org homepage, <http://www.bittorrent.org>.
- [Black 1995]** U. Black. *ATM Volume I: Foundation for Broadband Networks*. Prentice Hall, 1995.
- [Blumenthal 2001]** M. Blumenthal, D. Clark. "Rethinking the Design of the Internet: the End-to-end Arguments vs. the Brave New World", *ACM Transactions on Internet Technology*, vol. 1, n.1, ago. 2001, p. 70-109.
- [Bochman 1984]** G. V. Bochmann e C. A. Sunshine. "Formal methods in communication protocol design", *IEEE Transactions on Communications*, vol. 28, n. 4, abril 1980, p. 624-631.
- [Bolot 1994]** J-C. Bolot e T. Turletti. "A rate control scheme for packet video in the Internet", *Proc. 1994 IEEE Infocom*, p. 1216-1223.
- [Bolot 1996]** J-C. Bolot e Andreas Vega-Garcia. "Control Mechanisms for Packet Audio in the Internet", *Proc. 1996 IEEE Infocom*, 1996, p. 232-239.
- [Bradner 1996]** S. Bradner e A. Mankin. *IPng: Internet Protocol Next Generation*. Reading: Addison Wesley, 1996.
- [Brakmo 1995]** L. Brakmo e L. Peterson. "TCP Vegas: End to End Congestion Avoidance on a Global Internet", *IEEE Journal of Selected Areas in Communications*, vol. 13, n. 8, out. 1995, p. 1465-1480.
- [Breslau 2000]** L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang. "Endpoint Admission Control: Architectural Issues and Performance", *Proc. 2000 ACM SIGCOMM*, Estocolmo, ago. 2000.
- [Brodnik 1997]** A. Brodnik, S. Carlsson, M. Degemmark e S. Pink. "Small Forwarding Tables for Fast Routing Lookups", *Proceedings of ACM SIGCOMM '97*. Cannes, out. 1997, p. 3-15.
- [Bush 1945]** V. Bush. "As We May Think", *The Atlantic Monthly*, jul. 1945. <http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm>.
- [Byers 1998]** J. Byers, M. Luby, M. Mitzenmacher e A. Rege. "A digital fountain approach to reliable distribution of bulk data", *Proc. 1998 ACM SIGCOMM*, Vancouver, ago. 1998, p. 56-67.
- [Cablelabs 2004]** CableLabs homepage. <http://www.cablelabs.com>.
- [CacheLogic 2009]** CacheLogic homepage, <http://www.cacheologic.com>.
- [Caesar 2005]** M. Caesar, J. Rexford. "BGP Routing Policies in ISP Networks", *IEEE Network Magazine*, vol. 19, n. 6, nov. 2005.
- [Caldwell 2009]** C. Caldwell. The Prime Pages, <http://www.utm.edu/research/primes/prove>.
- [Cardwell 2000]** N. Cardwell, S. Savage e T. Anderson. "Modeling TCP Latency", *Proc. 2000 IEEE Infocom Conference*. Tel-Aviv, mar. 2000.
- [CASA 2009]** Center for Collaborative Adaptive Sensing of the Atmosphere. <http://www.casa.umass.edu>.
- [Casado 2007]** M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, "Ethane: Taking Control of the Enterprise", *Proc. ACM SIGCOMM'07*, ago. 2007, Kyoto, Japão.
- [Casner 1992]** S. Casner e S. Deering. "First IETF Internet Audiocast", *ACM SIGCOMM Computer Communications Review*, vol. 22, n. 3, jul. 1992, p. 92-97.
- [Ceiva 2009]** Ceiva homepage, <http://www.ceiva.com/>.
- [CENS 2009]** Center for Embedded Network Sensing, <http://www.cens.ucla.edu/>.
- [Cerf 1974]** V. Cerf e R. Kahn. "A Protocol for Packet Network Interconnection", *IEEE Transactions on Communications Technology*, vol. COM-22, n. 5, p. 627-641.

- [CERT 2001-09]** CERT. "Advisory 2001-09: Statistical Weaknesses in TCP/IP Initial Sequence Numbers", <http://www.cert.org/advisories/CA-2001-09.html>.
- [CERT 2003-04]** CERT. "CERT Advisory CA-2003-04 MS-SQL Server Worm", <http://www.cert.org/advisories/CA-2003-04.html>.
- [CERT 2009]** CERT Coordination Center. <http://www.cert.org/advisories>.
- [CERT Filtering 2002]** CERT. "Packet Filtering for Firewall Systems", http://www.cert.org/tech_tips/packet_filtering.html.
- [Cert SYN 1996]** CERT. "Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks", <http://www.cert.org/advisories/CA-1998-01.html>.
- [Chao 2001]** H. J. Chao, C. Lam e E. Oki. *Broadband Packet Switching Technologies — A Practical Guide to ATM Switches and IP Routers*. John Wiley & Sons, 2001.
- [Chen 2000]** G. Chen e D. Kotz. "A Survey of Context-Aware Mobile Computing Research", *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, nov. 2000. <http://www.cs.dartmouth.edu/reports/TR2000-381.pdf>.
- [Chen 2006]** K. -T. Chen, C. -Y. Huang, P. Huang, C.-L. Lei. "Quantifying Skype User Satisfaction", *Proc. 2006 ACM SIGCOMM*, Pisa, Itália, set. 2006.
- [Cheswick 2000]** B. Cheswick, H. Burch e S. Branigan. "Mapping and Visualizing the Internet", *Proc. 2000 Usenix Conference*. San Diego, jun. 2000.
- [Chiu 1989]** D. Chiu e R. Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks and ISDN Systems*, vol. 17, n. 1, p. 1-14. http://www.cs.wustl.edu/~jain/papers/cong_av.htm.
- [Christiansen 2001]** M. Christiansen, K. Jeffay, D. Ott e F. D. Smith. "Tuning Red for Web Traffic", *IEEE/ACM Transactions on Networking*, vol. 9, n. 3, jun. 2001, p. 249-264.
- [Chuang 2005]** S. Chuang, S. Iyer, N. McKeown. "Practical Algorithms for Performance Guarantees in Buffered Crossbars", *Proc. 2005 IEEE Infocom*.
- [Cicconetti 2006]** C. Cicconetti, L. Lenzini, A. Mingozi, K. Eklund. "Quality of Service Support in 802.16 Networks", *IEEE Network Magazine*, mar/abr. 2006, p. 50-55.
- [Cisco 12000 2009]** Cisco Systems Inc. "Cisco XR 12000 Series and Cisco 12000 Series Routers", <http://www.cisco.com/en/US/products/ps6342/index.html>.
- [Cisco 8500 2009]** Cisco Systems Inc. "Catalyst 8500 Campus Switch Router Architecture", http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/lcfg8500.pdf.
- [Cisco NAT 2009]** Cisco Systems Inc. "How NAT Works", http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml.
- [Cisco NAPA 2009]** Cisco Systems Inc. "Cisco Network Application Performance Analysis (NAPA) Solution", <http://www.cisco.com/en/US/products/sw/netmgtsw/index.html>.
- [Cisco QoS 2009]** Cisco Systems Inc. "Advanced QoS Services for the Intelligent Internet", http://www.cisco.com/warp/public/cc/pd/iosw/ioft/iotqo/tech/qos_wp.htm.
- [Cisco Queue 2009]** Cisco Systems Inc. "Congestio Management Overview", http://www.cisco.com/en/US/doc/ios/12_2/qos/configuration/guide/qcfconmg.html.
- [Cisco Security 2009]** Cisco Systems Inc. "Why You Need a Firewall", http://www.cisco.com/en/US/products/sw/secursw/ps743/products_user_guide_chapter09186a008007f303.html.
- [Cisco Switches 2009]** Cisco Systems Inc. "Multiservices Switches", <http://www.cisco.com/warp/public/cc/pd/si/index.shtml>.
- [Cisco SYN 2009]** Cisco Systems Inc. "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks", http://www.cisco.com/en/US/tech/tk828/technologies_tech_note09186a00900f67d5.shtml.

- [Clark 1988]** D. Clark. "The Design Philosophy of the DARPA Internet Protocols", *Proc 1988 ACM SIGCOMM*. Stanford, CA, ago. 1988.
- [Clarke 2002]** I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg e B. Wiley. "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, jan./fev. 2002, p. 40-49.
- [Cohen 1977]** D. Cohen. "Issues in Transnet Packetized Voice Communication", *Proc Fifth Data Communications Symposium*. Snowbird, set. 1977, p. 6-13.
- [Cohen 2003]** B. Cohen. "Incentives to Build Robustness in BitTorrent", *First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, jun. 2003.
- [Cookie Central 2009]** Cookie Central homepage, http://www.cookiecentral.com/n_cookie_faq.htm.
- [CoolStreaming 2005]** X. Zhang, J. Liu, B. Li e T.-S P. Yum. "CoolStreamingDONet/: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", *Proc. IEEE INFOCOM*, mar. 2005, Miami.
- [Cormen 2001]** T. H. Cormen. *Introduction to Algorithms*, 2 ed. Cambridge: MIT Press, 2001.
- [Crow 1997]** B. Crow, I. Widjaja, J. Kim e P. Sakai. "IEEE 802.11 Wireless Local Area Networks", *IEEE Communications Magazine*, set. 1997, p. 116-126.
- [Crowcroft 1995]** J. Crowcroft, Z. Wang, A. Smith e J. Adams. "A Comparison of the IETF and ATM Service Models", *IEEE Communications Magazine*, nov./dez. 1995, p. 12-16.
- [Crowcroft 1999]** J. Crowcroft, M. Handley e I. Wakeman. *Internetworking Multimedia*. São Francisco: Morgan-Kaufman, 1999.
- [Cusumano 1998]** M. A. Cusumano, D. B. Yoffie. *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft*. Nova York: Free Press, 1998.
- [Daigle 1991]** J. N. Daigle. *Queueing Theory for Telecommunications*. Reading, MA: Addison Wesley, 1991.
- [Dalal 1978]** Y. Dalal e R. Metcalfe. "Reverse Path Forwarding of Broadcast Packets", *Communications of the ACM*, vol. 21, n. 12, dez. 1978, p. 1040-1048.
- [Davie 2000]** B. Davie e Y. Rekhter. *MPLS: Technology and Applications*, Morgan Kaufmann Series in Networking, 2000.
- [Davies 2004]** [G. Davies, F. Kelly. "Network Dimensioning, Service Costing, and Pricing in a Packet-switched Environment", *Telecommunications Policy*, vol. 28, n. 4, p. 391-412.
- [DEC 1990]** Digital Equipment Corporation. "In Memoriam: J. C. R. Licklider 1915–1990", *SRC Research Report* 61, ago. 1990. <http://www.memex.org/licklider.pdf>.
- [DeClercq 2002]** J. DeClercq e O. Paridaens. "Scalability Implications of Virtual Private Networks", *IEEE Communications Magazine*, vol. 40, n. 5, maio 2002, p. 151-157.
- [Demers 1990]** A. Demers, S. Keshav e S. Shenker. "Analysis and Simulation of a Fair Queueing Algorithm", *Internetworking: Research and Experience*, vol. 1, n. 1, 1990, p. 3-26.
- [Denning 1997]** D. Denning (editor) e P. Denning (prefácio). *Internet Besieged: Countering Cyberspace Scofflaws*. Reading: Addison Wesley, 1997.
- [dhc 2009]** IETF Dynamic Host Configuration working group, <http://www.ietf.org/html.charters/dhc-charter.html>.
- [Diggavi 2004]** S. N. Diggavo, N. Al-Dahir, A. Stamoulis, R. Calderbank. "Great Expectations: The Value of Spatial Diversity in Wireless Networks", *Proceedings of the IEEE*, vol. 92, n. 2, fev. 2004.
- [Diot 2000]** C. Diot, B. N. Levine, B. Lyles, H. Kassem e D. Balensiefen. "Deployment Issues for the IP Multicast Service and Architecture", *IEEE Network*, vol. 14, n. 1, jan./fev. 2000, p. 78-88.
- [Dimitropoulos 2007]** X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, KC Claffy, G. Riley. "AS Relationships: Inference and Validation", *ACM Computer Communication Review*, jan. 2007.
- [Dodge 2009]** M. Dodge. "An Atlas of Cyberspaces", http://www.cybergeography.org/atlas/isp_maps.html.

- [Donahoo 2001]** M. Donahoo, K. Calvert. *TCP/IP Sockets in C: Practical Guide for Programmers*. Morgan Kaufman, 2000.
- [Douceur 2002]** J. R. Douceur. "The Sybil Attack", *First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, mar. 2002.
- [DSL 2009]** DSL Forum homepage, <http://www.dslforum.org/>.
- [Dhungel 2008]** P. Dhungel, D. Wu, B. Schonhorst, K. W. Ross. "A Measurement Study of Attacks on BitTorrent Leechers", *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)*, Tampa Bay, fev. 2008.
- [Droms 2002]** R. Droms, T. Lemon. *The DHCP Handbook*, 2 ed., SAMS Publishing, 2002.
- [Edney 2003]** J. Edney e W.A. Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*, Addison-Wesley Professional, 2003.
- [Eklund 2002]** K. Eklund, R. Marks, K. Stanswood, S. Wang. "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access", *IEEE Communications Magazine*, jun. 2002, p. 98-107.
- [Ellis 1987]** H. Ellis. "The Story of Non-Secret Encryption", <http://jya.com/ellisdoc.htm>
- [Ericsson 2009]** Ericsson. "The Evolution of the Edge", http://www.ericsson.com/technology/whitepapers/broadband/evolution_of_EDGE.shtml.
- [ESM 2009]** End System Multicast homepage, <http://esm.cs.cmu.edu/>.
- [Estrin 1997]** D. Estrin, M. Handley, A. Helmy, P. Huang e D. Thaler. "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing", *Proceedings of IEEE Infocom '98*, Nova York, abr. 1998.
- [Falkner 2007]** J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, T. Anderson. "Profiling a Million Sser DHT", *Proc. ACM Internet Measurement Conference*, nov. 2007.
- [Faloutsos 1999]** C. Faloutsos, M. Faloutsos e P. Faloutsos. "What Does the Internet Look Like? Empirical Laws of the Internet Topology", *Proc. 1999 ACM SIGCOMM*, Boston, ago. 1999.
- [Feamster 2004]** N. Feamster, J. Winick e J. Rexford. "A Model for BGP Routing for Network Engineering", *Proc. 2004 ACM SIGMETRICS*, Nova York, jun. 2004.
- [Feldman 2005]** M. Feldman J. Chuang. "Overcoming Free-riding Behavior in Peer-to-Peer Systems", *ACM SIGecom Exchanges*, jul. 2005.
- [Feldmeier 1988]** D. Feldmeier. "Improving Gateway Performance with a Routing Table Cache", *Proc. 1988 IEEE Infocom*, Nova Orleans, mar. 1988.
- [Feldmeier 1995]** D. Feldmeier. "Fast Software Implementation of Error Detection Codes", *IEEE/ACM Transactions on Networking*, vol. 3, n. 6, dez. 1995, p. 640-652.
- [FIPS 1995]** Federal Information Processing Standard. "Secure Hash Standard", *FIPS Publication 180-1*. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- [Floyd 1999]** S. Floyd, M. Handley, J. Padhye, J. Widmer. "Equation-Based Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, vol. 6, n. 5, out. 1998, p. 458-472.
- [Floyd 2000]** S. Floyd, M. Handley, J. Padhye e J. Widmer. "Equation-Based Congestion Control for Unicast Applications", *Proc. 2000 ACM Sigcomm Conference*, Estocolmo, ago. 2000.
- [Floyd 2001]** S. Floyd. "A Report on Some Recent Developments in TCP Congestion Control", *IEEE Communications Magazine*, abr. 2001.
- [Floyd 2009]** S. Floyd. "References on RED (Random Early Detection) Queue Management", <http://www.icir.org/floyd/red.html>.
- [Floyd Synchronization 1994]** S. Floyd e V. Jacobson. "Synchronization of Periodic Routing Messages", *IEEE/ACM Transactions on Networking*, vol. 2, n. 2, abr. 1997, p. 122-136.
- [Floyd TCP 1994]** S. Floyd. "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, vol. 24, n. 5, out. 1994, p. 10-23.

- [Fluhrer 2001]** S. Fluhrer, I. Mantin e A. Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4", *Eighth Annual Workshop on Selected Areas in Cryptography*, Toronto, ago. 2002.
- [Fortz 2000]** B. Fortz e M. Thorup. "Internet Traffic Engineering by Optimizing OSPF Weights", *Proc. 2000 IEEE Infocom*, Tel Aviv, abr. 2000.
- [Fortz 2002]** B. Fortz, J. Rexford e M. Thorup. "Traffic Engineering with Traditional IP Routing Protocols", *IEEE Communication Magazine*, out. 2002.
- [Foster 2002]** I. Foster. "The Grid: A New Infrastructure for 21st Century Science", *Physics Today*, vol. 55, n. 2, 2002, p. 42-47.
- [Fraleigh 2003]** C. Fraleigh, T. Tobagi, C. Diot. "Provisioning IP backbone Networks to Support Latency Sensitive Traffic", *Proc. IEEE Infocom Conference*, San Francisco, mar. 2003.
- [France Telecom 2009]** Object Identifier (OID) repository, <http://asn1.elibel.tm.fr/oid/>.
- [Freedman 2004]** M. J. Freedman, E. Freudenthal, D. Mazires. "Democratizing Content Publication with Coral", *USENIX NSDI*, 2004.
- [Friedman 1999]** T. Friedman e D. Towsley. "Multicast Session Membership Size Estimation", *Proc. 1999 IEEE Infocom*, Nova York, mar. 1999.
- [Frost 1994]** J. Frost. "BSD Sockets: A Quick and Dirty Primer", <http://world.std.com/~jimf/papers/sockets/sockets.html>.
- [Gallager 1983]** R. G. Gallagher, P. A. Humblet e P. M. Spira. "A Distributed Algorithm for Minimum Weight-Spanning Trees", *ACM Trans. on Programming Languages and Systems*, vol. 1, n. 5, jan. 1983, p. 66-77.
- [Gao 2001]** L. Gao e J. Rexford. "Stable Internet Routing Without Global Coordination", *IEEE/ACM Trans. Networking*, vol. 9, n. 6, dez. 2001, p. 681-692.
- [Garces-Erce 2003]** L. Garces-Erce, K. W. Ross, E. Biersack, P. Felber e G. Urvoy-Keller. "TOPLUS: Topology Centric Lookup Service", *Fifth International Workshop on Networked Group Communications (NGC'03)*, Munique, set. 2003. <http://cis.poly.edu/~ross/papers/TOPLUS.pdf>.
- [Gartner 2003]** F. C. Gartner. "A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms", *Technical Report IC/2003/38*, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, 10 jun. 2003. http://ic2.epfl.ch/publications/documents/IC_TECH_REPORT_200338.pdf.
- [Gauthier 1999]** L. Gauthier, C. Diot e J. Kurose. "End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet", *Proc. 1999 IEEE Infocom*, Nova York, abr. 1999.
- [Giacopelli 1990]** J. Giacopelli, M. Littlewood e W. D. Sincoskie. "Sunshine: A High Performance Self-Routing Broadband Packet Switch Architecture", *1990 International Switching Symposium*. Uma versão ampliada desse documento pode ser encontrada em *IEEE J. Sel. Areas in Commun.*, vol. 9, n. 8, out. 1991, p. 1289-1298.
- [Girard 1990]** A. Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Reading: Addison Wesley, 1990.
- [Glitho 1995]** R. Glitho e S. Hayes (eds.). "Special Issue on Telecommunications Management Network", *IEEE Communications Magazine*, vol. 33, n. 3, mar. 1995.
- [Glitho 1998]** R. Glitho. "Contrasting OSI Systems Management to SNMP and TMN", *Journal of Network and Systems Management*, vol. 6, n. 2, jun. 1998, p. 113-131.
- [Gnutella 2009]** "The Gnutella Protocol Specification, v0.4", http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.
- [Goodman 1997]** D.J. Goodman. *Wireless Personal Communications Systems*. Prentice-Hall, 1997.
- [Goralski 1999]** W. Goralski. *Frame Relay for High-Speed Networks*. Nova York: John Wiley, 1999.
- [Goralski 2001]** W. Goralski. *Optical Networking and WDM*. Berkeley: Osborne/McGraw-Hill, 2001.
- [Griffin 2009]** T. Griffin. "Interdomain Routing Links", <http://www.cl.cam.ac.uk/~tgg22/interdomain/>.

- [Guha 2006]** S. Guha, N. Daswani, R. Jain. "An Experimental Study of the Skype Peer-to-Peer VoIP System", *Proc. Fifth Int. Workshop on P2P Systems*, Santa Bárbara, 2006.
- [Guo 2005]** L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang. "Measurement, Analysis, and Modeling of BitTorrent-like Systems", *ACM Internet Measurement Conference (IMC)*, 2005.
- [Gupta 1998]** P. Gupta, S. Lin e N. McKeown. "Routing Lookups in Hardware at Memory Access Speeds", *Proc. 1998 IEEE Infocom 1998*, São Francisco, abr. 1998, p. 1241-1248.
- [Gupta 2001]** P. Gupta e N. McKeown. "Algorithms for Packet Classification", *IEEE Network Magazine*, vol. 15, n. 2, mar./abr. 2001, p. 24-32.
- [Ha 2008]** I. Rhee, S. Ha, L. Xu. "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *ACM SIGOPS Operating System Review*, 2008.
- [Hain 2005]** T. Hain. "A Programmatic Report on IPv4 Address Space Consumption", *Internet Protocol Journal*, vol. 8, n. 3.
- [Halabi 2000]** S. Halabi. *Internet Routing Architectures*, 2 ed. Cisco Press, 2000.
- [Halperin 2008]** D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, W. Maisel. "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses", *Proc. 29th Annual IEEE Symposium on Security and Privacy*, maio 2008.
- [Hamada 1997]** T. Hamada, H. Kamata e S. Hogg. "An Overview of the TINA Management Architecture", *Journal of Network and Systems Management*, vol. 5, n. 4, dez. 1997, p. 411-435.
- [Hei 2007]** X. Hei, C. Liang, J. Liang, Y Liu, K. W. Ross. "A measurement Study of a Large-scale P2P IPTV System", *IEEE Trans. on Multimedia*, dez. 2007.
- [Heidemann 1997]** J. Heidemann, K. Obraczka e J. Touch. "Modeling the Performance of HTTP over Several Transport Protocols", *IEEE/ACM Transactions on Networking*, vol. 5, n. 5, out. 1997, p. 616-630.
- [Held 2001]** G. Held. *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs*. McGraw-Hill, 2001.
- [Hersent 2000]** O. Hersent, D. Gurle e J-P. Petit. *IP Telephony: Packet-Based Multimedia Communication Systems*, Edinburgh: Pearson Education Limited, 2000.
- [Holland 2001]** G. Holland, N. Vaidya, V. Bahl. "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks", *Proc. 2001 ACM Int. Conference of Mobile Computing and Networking (Mobicom01)*, Roma, jul. 2001.
- [Holot 2002]** C. V. Holot, V. Misra, D. Towsley e W. Gong. "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows", *IEEE Transactions on Automatic Control*, vol. 47, n. 6, jun. 2002, p. 945-959.
- [Huang 2002]** C. Haung, V. Sharma, K. Owens e V. Makam. "Building Reliable MPLS Networks Using a Path Protection Mechanism", *IEEE Communications Magazine*, vol. 40, n. 3, mar. 2002, p. 156-162.
- [Huang 2005]** Y. Huang, R. Guerin. "Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?", *Proc. IEEE Int. Conf. Network Protocols (ICNP)*, Boston, nov. 2005.
- [Huang 2007]** C. Huang, J. Li, Angela Wang e K. W. Ross. "Can Internet VoD be Profitable?", *Proc. ACM SIGCOMM*, Kyoto, ago. 2007.
- [Huang 2008]** C. Huang, J. Li, Angela Wang e K. W. Ross. "Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own Red Swoosh", *NOSSDAV 2008*, Braunschweig, maio 2008.
- [Huitema 1998]** C. Huitema. *IPv6: The New Internet Protocol*, 2 ed. Englewood Cliffs: Prentice Hall, 1998.
- [Huston 1999a]** G. Huston. "Interconnection, Peering, and Settlements — Part I", *The Internet Protocol Journal*, vol. 2, n. 1, mar. 1999.
- [Huston 2001]** G. Huston. "Analyzing the Internet BGP Routing Table", *The Internet Protocol Journal*, vol. 4, n. 1, mar. 2001.
- [Huston 2004]** G. Huston. "NAT Anatomy: A Look Inside Network Address Translators". *The Internet Protocol Journal*, vol. 7, n. 3, set. 2004.

- [Huston 2008a]** G. Huston. “Confronting IPv4 Address Exhaustion”, <http://www.potaroo.net/ispcol/2008-10/v4depletion.html>.
- [Huston 2008b]** G. Huston, G. Michaelson. “IPv6 Deployment: Just where are we?”, <http://www.potaroo.net/ispcol/2008-04/ipv6.html>.
- [IAB 2009]** Internet Architecture Board homepage, <http://www.iab.org/>.
- [IANA 2009a]** Internet Assigned Number Authority homepage, <http://www.iana.org/>.
- [IANA 2009b]** Internet Assigned Number Authority, “Private Enterprise Numbers”, <http://www.iana.org/assignments/enterprise-numbers>.
- [IANA Protocol Numbers 2009]** Internet Assigned Numbers Authority, Protocol Numbers, <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.
- [ICANN 2009]** The Internet Corporation for Assigned Names and Numbers homepage, <http://www.icann.org>.
- [IEC Optical 2009]** IEC Online Education. “Optical Access”, http://www.iec.org/online/tutorials/opt_acc/.
- [IEEE 802 2009]** IEEE 802 LAN/MAN Standards Committee homepage, <http://www.ieee802.org/>.
- [IEEE 802.11 1999]** IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Network — Specific Requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. <http://standards.ieee.org/getieee802/download/802.11-1999.pdf>.
- [IEEE 802.11n]** IEEE, “IEEE P802.11 — Task Group N — Meeting Update: Status of 802.11n”, http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm.
- [IEEE 802.15 2009]** IEEE 802.15 Working Group for WPAN homepage, <http://grouper.ieee.org/groups/802/15/>.
- [IEEE 802.16d 2004]** IEEE, “IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems”, <http://standards.ieee.org/getieee802/download/802.16-2004.pdf>.
- [IEEE 802.16e 2005]** IEEE, “IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1”, <http://standards.ieee.org/getieee802/download/802.16e-2005.pdf>.
- [IEEE 802.1q 2005]** IEEE, “IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks”, <http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf>.
- [IEEE 802.1X]** IEEE Std 802.1X-2001 Port-Based Network Access Control. http://standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html.
- [IEEE 802.3 2009]** IEEE, “IEEE 802.3 CSMA/CD (Ethernet)”, <http://grouper.ieee.org/groups/802/3>.
- [IEEE 802.5 2009]** IEEE, IEEE 802.5 homepage, <http://ieee802.org/5/www802.5org/>.
- [IETF 2009]** Internet Engineering Task Force homepage. <http://www.ietf.org>.
- [IMAP 2009]** The IMAP Connection. <http://www imap.org/>.
- [Intel 2009]** Intel Corp, “Intel® 82544 Gigabit Ethernet Controller”, http://www.intel.com/design/network/products/lan/docs/8255_docs.htm.
- [Intel WiMax 2009]** Intgel Corp., “WiMax Technology”, <http://www.intel.com/technology/wimax/index.htm>.
- [Internet2 Multicast 2009]** Internet2 Multicast Working Group homepage, <http://multicast.internet2.edu/>.
- [IPv6 2009]** IPv6.com homepage, <http://www.ipv6.com/>.
- [ISC 2009]** Internet Systems Consortium homepage, <http://www.isc.org>.
- [ISI 1979]** Information Sciences Institute, “DoD Standard Internet Protocol”, Internet Engineering Note 123, dez. 1979, <http://www.isi.edu/in-notes/ien/ien123.txt>.

- [ISO 2009]** International Organization for Standardization homepage, International Organization for Standardization, <http://www.iso.org/>.
- [ISO X.680 2002]** International Organization for Standardization. “X.680: ITU-T Recommendation X.680 (2002) Information Technology – Abstract Syntax Notation One (ASN.1): Specification of Basic Notation”, <http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf>.
- [ITU 2005]** International Telecommunication Union. *The Internet of Things*, 2005, http://www.itu.int/osp/spu/publications/internetofthings/InternetofThings_summary.pdf.
- [ITU 2009]** The ITU homepage, <http://www.itu.int/>.
- [ITU Statistics 2009]** International Telecommunication Union. “ICT Statistics”, <http://www.itu.int/ITU-D/ict-eye/Reports.aspx>.
- [Iyer 2002]** S. Iyer, R. Zhang e N. McKeown. “Routers with a Single Stage of Buffering”, *Proc. 2002 ACM Sigcomm Pittsburgh*, ago. 2002.
- [Jacobson 1988]** V. Jacobson. “Congestion Avoidance and Control”, *Proceedings of ACM SIGCOMM*, Stanford, ago. 1988, p. 314-329.
- [Jacobson 1988]** V. Jacobson. “Congestion Avoidance and Control”, *Proc. 1988 ACM SIGCOMM*, Pittsburgh, ago. 2002.
- [Jain 1986]** R. Jain. “A timeout-based congestion control scheme for window flow-controlled networks”, *IEEE Journal of Selected Areas in Communications SAC-4*, 7, out. 1986.
- [Jain 1989]** R. Jain. “A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks”, *ACM SIGCOMM Computer Communications Review*, vol. 19, n. 5, 1989, p. 56-71.
- [Jain 1994]** R. Jain. *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*. Reading: Addison-Wesley, 1994.
- [Jain 1996]** R. Jain, S. Kalyanaraman, S. Fahmy, R. Goyal e S. Kim. “Tutorial Paper on ABR Source Behavior”, *ATM Forum/96-1270*, out. 1996, <http://www.cse.wustl.edu/~jain/atmfs/ftp/atm96-1270.pdf>.
- [Jaiswal 2003]** S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose e D. Towsley. “Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone”, *Proc. 2003 IEEE INFOCOM*.
- [Jakobson 1993]** G. Jacobson e M. Weissman. “Alarm Correlation”, *IEEE Network Magazine*, 1993, p. 52-59.
- [Ji 2003]** P. Ji, Z. Ge, J. Kurose e D. Towsley. “A Comparison of Hard-State and Soft-State Signaling Protocols”, *Proc. 2003 ACM SIGCOMM*, Karlsruhe, ago. 2003, <http://www.acm.org/sigs/sigcomm/sigcomm2003/papers/p251-ji.pdf>.
- [Jiang 2001]** W. Jiang, J. Lennox, H. Schulzrinne e K. Singh. “Towards Junking the PBX: Deploying IP Telephony”, *NOSSDAV'01*. Port Jefferson, jun. 2001.
- [Jin 2004]** C. Jin, D. X. We e S. Low. “FAST TCP: Motivation, Architecture, Algorithms, Performance”, *Proceeding of IEEE Infocom*, Hong Kong, mar. 2004.
- [Juniper 2009]** Juniper Networks. “Provider Network Management”, http://www.juniper.net/solutions/service_provider/provider_network_management/.
- [Kaaranen 2001]** H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen e V. Niemi. *UMTS Networks, Architecture, Mobility and Services*. Nova York: John Wiley & Sons, 2001.
- [Kahn 1978]** R. E. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, “Advances in Packet Radio Tehcnology”, *Proc. Of IEEE*, 66, 11, nov. 1978.
- [Kamerman 1997]** A. Kamerman, L. Monteban. “WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band”, *Bell Labs Technical Journal*, 1997, p. 118-133.
- [Kangasharju 2000]** J. Kangasharju, K. W. Ross e J. W. Roberts. “Performance Evaluation of Redirection Schemes in Content Distribution Networks”, *Proc. 5th Web Caching and Content Distribution Workshop*, Lisboa, maio 2000.

- [Kar 2000]** K. Kar, M. Kodialam e T.V. Lakshman. "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications", *IEEE J. Selected Areas in Communications*, dez. 2000.
- [Karol 1987]** M. Karol, M. Hluchyj e A. Morgan. "Input Versus Output Queuing on a Space-Division Packet Switch", *IEEE Transactions on Communications*, vol. COM-35, n. 12, dez. 1987, p. 1347-1356.
- [Katabi 2002]** D. Katabi, M. Handley, C. Rohrs. "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", *Proc. 2002 ACM SIGCOMM*, Pittsburgh, ago. 2002.
- [Katzela 1995]** I. Katzela e M. Schwartz. "Schemes for Fault Identification in Communication Networks", *IEEE/ACM Transactions on Networking*, vol. 3, n. 6, dez. 1995, p. 753-764.
- [Kaufman 1995]** C. Kaufman, R. Perlman e M. Speciner. *Network Security, Private Communication in a Public World*. Englewood Cliffs: Prentice Hall, 1995.
- [Kelly 1998]** F. P. Kelly. A. Maulloo, D. Tan. "Rate control for communication networks: Shadow prices, proportional fairness and stability", *J. Operations Res. Soc.*, vol. 49, n. 3, p. 237-252, mar. 1998.
- [Kelly 2003]** F. P. Kelly. "Scalable TCP: improving performance in high speed wide area networks", *ACM SIGCOMM Computer Communications Review*, vol. 33, n. 2, abr. 2003, p. 83-91.
- [Keshav 1998]** S. Keshav e R. Sharma. "Issues and Trends in Router Design", *IEEE Communications Magazine*, vol. 36, n. 5, maio 1998, p. 144-151.
- [Keslassy 2003]** I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, Mc Keown. "Scaling Internet Routers Using Optics", *Proc. 2003 ACM SIGCOMM*, Karlsruhe, ago. 2003.
- [Kilkki 1999]** K. Kilkki. *Differentiated Services for the Internet*. Indianápolis: Macmillan Technical Publishing, 1999.
- [Kim 2005]** H. Kim, S. Rixner, V. Pai. "Network Interface Data Caching", *IEEE Transactions on Computers*, vol. 54, n. 11, ov. 2005, p. 1394-1408.
- [Kim 2008]** C. Kim, M. Caesar, J. Rexford. "Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises", *Proc. ACM SIGCOMM '08*, ago. 2008, Seattle.
- [Kleinrock 1961]** L. Kleinrock. "Information Flow in Large Communication Networks", *RLE Quarterly Progress Report*, jul. 1961.
- [Kleinrock 1964]** L. Kleinrock. *1964 Communication Nets: Stochastic Message Flow and Delay*. Nova York: McGraw-Hill, 1964.
- [Kleinrock 1975]** L. Kleinrock. *Queuing Systems*, vol. 1. Nova York: John Wiley, 1975.
- [Kleinrock 1975b]** L. Kleinrock e F. A. Tobagi. "Packet Switching in Radio Channels: Part I – Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics", *IEEE Transactions on Communications*, vol. COM-23, n. 12, dez. 1975, p. 1400-1416.
- [Kleinrock 1976]** L. Kleinrock. *Queuing Systems*, vol. 2. Nova York: John Wiley, 1976.
- [Kleinrock 2004]** L. Kleinrock. "The Birth of the Internet", <http://www.lk.cs.ucla.edu/LK/Inet/birth.html>
- [Kohler 2006]** E. Kohler, M. Handley, S. Floyd. "DCCP: Designing DCCP: Congestion Control Without Reliability", *Proc. 2006 ACM SIGCOMM*, Pisa, set. 2006
- [Korhonen 2003]** J. Korhonen. *Introduction to 3G Mobile Communications*, 2 ed. Artech House, 2003.
- [Krishnamurthy 2001]** B. Krishnamurthy e J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, and Traffic Measurement*. Boston: Addison-Wesley, 2001.
- [Krishnamurthy 2001b]** B. Krishnamurthy, C. Wills, Y. Zhang. "On the Use and Performance of Content Distribution Networks", *ACM Internet Measurement Conference*, 2001.
- [Kulkarni 2005]** S. Kulkarni, C. Rosenberg. "Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces and Short Term Fairness", *Wireless Networks*, 11, 557-569, 2005.

- [Kumar 2006]** R. Kumar, K. W. Ross. "Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems", *IEEE Workshop on Hot Topics in Web Systems and Technologies*, Boston, 2006.
- [Kurose 1996]** J. F. Kurose. "Unix Network Programming", http://gaia.cs.umass.edu/ntu_socket/.
- [Labovitz 1997]** C. Labovitz, G.R. Malan e F. Jahanian. "Internet Routing Instability", *Proceedings of ACM SIGCOMM '97*, Cannes, 1997, p. 115-126. <http://www.acm.org/sigcomm/sigcomm97/papers/p109.html>.
- [Labrador 1999]** M. Labrador e S. Banerjee. "Packet Dropping Policies for ATM and IP Networks", *IEEE Communications Surveys*, vol. 2, n. 3, Third Quarter 1999, p. 2-14.
- [Lacage 2004]** M. Lacage, M. H. Manshaei, T. Turletti. "IEEE 802.11 Rate Adaption: A Practical Approach", *ACM Int Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)*, out. 2004, Veneza.
- [Lakshman 1997]** T. V. Lakshman e U. Madhow. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, vol. 5, n. 3, 1997, p. 336-350.
- [Lam 1980]** S. Lam. "A Carrier Sense Multiple Access Protocol for Local Networks", *Computer Networks*, vol. 4, 1980, p. 21-32, 1980.
- [Lamport 1981]** L. Lamport. "Password Authentication with Insecure Communication", *Communications of the ACM*, vol. 24, n. 11, nov. 1981, p. 770-772.
- [Larmouth 1996]** J. Larmouth. *Understanding OSI*. International Thomson Computer Press, 1996. O capítulo 8 fala sobre a ASN.1 e está disponível on-line em <http://www.salford.ac.uk/iti/books/osi/all.html#head8>.
- [Lawton 2001]** G. Lawton. "Is IPv6 Finally Gaining Ground?", *IEEE Computer Magazine*, ago. 2001, p. 11-15.
- [Leiner 1998]** B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts e S. Woolf. "A Brief History of the Internet", <http://www.isoc.org/internet/history/brief.html>.
- [Li 2004]** L. Li, D. Alderson, W. Willinger, J. Doyle. "A First-Principles Approach to Understanding the Internet's Router-Level Topology", *Proc. 2004 ACM SIGCOMM*, Portland, ago. 2004.
- [Li 2007]** J. Li, M. Guidero, Z. Wu, E. Purpus, T. Ehrenkranz. "BGP Routing Dynamics Revisited", *ACM Computer Communication Review*, abr. 2007.
- [Liang 2006]** J. Liang, N. Naoumov, K. W. Ross. "The Index Poisoning Attack in P2P File-Sharing Systems", *Proc. 2006 IEEE Infocom 2006*, Barcelona, abr. 2006. <http://cis.poly.edu/~ross/papers/>.
- [Lin 2001]** Y. Lin e I. Chlamtac. *Wireless and Mobile Network Architectures*. Nova York: John Wiley & Sons, 2001.
- [Liogkas 2006]** N. Liogkas, R. Nelson, E. Kohler, L. Zhang. "Exploiting BitTorrent For Fun (But Not Profit)", *6th International Workshop on Peer-to-Peer Systems (IPTPS 2006)*.
- [Liu 2002]** B. Liu, D. Goeckel e D. Towsley. "TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks", *Proc. 2002 Global Internet*.
- [Locher 2006]** T. Locher, P. Moor, S. Schmid, R. Wattenhofer. "Free Riding in BitTorrent is Cheap", *Proc. ACM HotNets 2006*, Irvine, nov. 2006.
- [Lui 2004]** J. Lui, V. Misra, D. Rubenstein. "On the Robustness of Soft State Protocols", *Proc. IEEE Int. Conference on Network Protocols (ICNP '04)*, p. 50-60.
- [Luotonen 1998]** A. Luotonen. *Web Proxy Servers*. Englewood Cliffs: Prentice Hall, 1998.
- [Lynch 1993]** D. Lynch e M. Rose. *Internet System Handbook*. Reading: Addison Wesley, 1993.
- [Macedonia 1994]** M. R. Macedonia e D. P. Brutzman. "MBone Provides Audio and Video Across the Internet", *IEEE Computer Magazine*, vol. 27, n. 4, abr. 1994, p. 30-36.
- [Mahdavi 1997]** J. Mahdavi e S. Floyd. "TCP-Friendly Unicast Rate-Based Flow Control", nota não publicada, jan. 1997.

- [Malware 2006]** Computer Economics. “2005 Malware Report: The Impact of Malicious Code Attacks”, <http://www.computereconomics.com>.
- [Manet 2009]** IETF Mobile Ad-Hoc Networks (manet) Working Group. <http://www.ietf.org/html.charters/manet-charter.html>.
- [Mao 2002]** Z. M. Mao, C. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, J. Wang. “A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers”, USENIX 2002.
- [Maymounkov 2002]** P. Maymounkov e D. Mazières. “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric”, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, mar. 2002, p. 53-65.
- [McKeown 1997a]** N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick e M. Horowitz. “The Tiny Tera: A Packet Switch Core”, *IEEE Micro Magazine*, jan./fev. 1997.
- [McKeown 1997b]** N. McKeown. “A Fast Switched Backplane for a Gigabit Switched Router”, *Business Communications Review*, vol. 27, n. 12. http://tiny-tera.stanford.edu/-nickm/papers/cisco_fasts-wp.pdf.
- [McQuillan 1980]** J. McQuillan, I. Richer e E. Rosen. “The New Routing Algorithm for the Arpanet”, *IEEE Transactions on Communications*, COM-28(5), maio 1980, p. 711-719.
- [Medhi 1997]** D. Medhi e D. Tipper (eds.). “Special Issue: Fault Management in Communication Networks”, *Journal of Network and Systems Management*, vol. 5, n. 2, jun. 1997.
- [Meng 2005]** X. Meng. “IPv4 Address Allocation and the BGP Routing Table Evolution”, *Computer Communication Reviews*, vol. 35, n. 1, 2005, p. 71-80.
- [Metcalfe 1976]** R.M. Metcalfe e D. R. Boggs. “Ethernet: Distributed Packet Switching for Local Computer Networks”, *Communications of the Association for Computing Machinery*, vol. 19, n. 7, jul. 1976, p. 395-404.
- [MFA Forum 2009]** Ip/MPLS Forum homepage, <http://www.ipmplsforum.org/>.
- [Microsoft Player Media 2009]** Microsoft Windows Media homepage. <http://www.microsoft.com/windows/windowsmedia/>.
- [Miller 1997]** M. A. Miller. *Managing Internetworks with SNMP*, 2 ed. Nova York: M & T Books, 1997.
- [Mirkovic 2005]** J. Mirkovic, S. Dietrich, D. Dittrich, P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms*, Prentice Hall, 2005.
- [Mockapetris 1988]** P. V. Mockapetris e K. J. Dunlap. “Development of the Domain Name System”, *Proceedings of SIGCOMM '88*, Stanford, 1988.
- [Mockapetris 2005]** P. Mockapetris, Sigcomm Award lecture, vídeo disponível em <http://www.postel.org/sigcomm>.
- [Mogul 2003]** J. Mogul. “TCP offload is a dumb idea whose time has come”, *Proc. HotOS IX: The 9th Workshop on Hot Topics in Operating Systems*, 2003, USENIX Association.
- [Molinero-Fernandez 2002]** P. Molinero-Fernandez, N. McKeown e H. Zhang. “Is IP Going to Take Over the World (of Communications)?”, *Proc. 2002 ACM Hotnets*.
- [Molle 1987]** M. L. Molle, K. Sohraby e A. N. Venetsanopoulos. “Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks”, *IEEE Journal on Selected Areas in Communications*, vol. 5, n. 6, 1987, p. 956-968.
- [Moore 2001]** D. Moore, G. Voelker, S. Savage. “Inferring Internet Denial of Service Activity”, *Proc. 2001 USENIX Security Symposium*, Washington, ago. 2001.
- [Moore 2003]** D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford e N. Weaver. “Inside the Slammer Worm”, *2003 IEEE Security and Privacy Conference*.
- [Moshchuk 2006]** A. Moshchuk, T. Bragin, S. Gribble, H. Levy. “A Crawler-based Study of Spyware on the Web”, *Proc. 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006)*, San Diego, fev. 2006.

- [Mouly 1992]** M. Mouly e M. Pautet. *The GSM System for Mobile Communications*. Palaiseau: Cell and Sys, 1992.
- [Moy 1998]** J. Moy. *OSPF: Anatomy of An Internet Routing Protocol*. Reading: Addison Wesley, 1998.
- [Mukherjee 1997]** B. Mukherjee. *Optical Communication Networks*. McGraw-Hill, 1997.
- [Murphy 2003]** veja [RFC 4272].
- [Nahum 2002]** E. Nahum, T. Barzilai e D. Kandlur. “Performance Issues in WWW Servers”, *IEEE/ACM Transactions on Networking*, vol. 10, n. 1, fev. 2002.
- [Naoumov 2006]** N. Naoumov, K. W. Ross. “Exploiting P2P Systems for DDoS Attacks”, *Intl Workshop on Peer-to-Peer Information Management*, Hong Kong, mai. 2006.
- [Neglia 2007]** G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, J. Danaher. “Availability in BitTorrent Systems”, *Proc. IEEE INFOCOM 2007*, mai. 2007.
- [Neumann 1997]** R. Neumann. “Internet Routing Black Hole”, *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems*, vol. 19, n. 12, mai. 1997, <http://catless.ncl.ac.uk/Risks/19.12.html#subj1.1>.
- [Newman 2008]** D. Newman. “802.11n Gear 10 Times Faster Than Current Wi-Fi Offerings”, *Network World*, 27 out, 2008. <http://www.networkworld.com/reviews/2008/102708-wlan-test.html?page=1>.
- [Nicholson 2006]** A. Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall. “Improved Access Point Selection”, *Proc. 2006 ACM Mobicom Conference*, Uppsala, 2006.
- [Nielsen 1997]** H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie e C. Lilley. “Network Performance Effects of HTTP/1.1, CSS1, and PNG”, *W3C Document*, 1997 (pode ser encontrado também em *Proceedings of ACM SIGCOMM '97*, Cannes, p. 155-166).
- [NIST 2001]** National Institute of Standards and Technology. “Advanced Encryption Standard (AES)”, *Federal Information Processing Standards* 197, nov. 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [Nmap 2009]** Nmap homepage, <http://www.insecure.com/nmap>.
- [Nonnenmacher 1998]** J. Nonnenmacher, E. Biersak e D. Towsley. “Parity-Based Loss Recovery for Reliable Multicast Transmission”, *IEEE/ACM Transactions on Networking*, vol. 6, n. 4, ago. 1998, p. 349-361.
- [NTIA 1998]** National Telecommunications and Information Administration (NTIA), U.S. Department of Commerce. “Management of Internet Names and Addresses”, Docket Number: 980212036-8146-02. http://www.ntia.doc.gov/ntiahome/domainname/6_5_98dns.htm.
- [Odlyzko 2003]** A. Odlyzko. “Internet Traffic Growth: Sources and Implications”, *A. M. Optical Transmission Systems and Equipment for WDM Networking II, Proc. SPIE*, 5247, 2003, p. 1-15. <http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf>.
- [OSI 2009]** International Organization for Standardization homepage, <http://www.iso.org/iso/en/ISOOnline.frontpage>.
- [OSS 2009]** OSS Nokalva. “ASN.1 Resources”, <http://www.oss.com/asn1/>.
- [Padhye 2000]** J. Padhye, V. Firoiu, D. Towsley, J. Kurose. “Modeling TCP Reno Performance: A Simple Model and its Empirical Validation”, *IEEE/ACM Transaction on Networking*, vol. 8, n. 2, abr., p. 133-145.
- [Padhye 2001]** J. Padhye, S. Floyd. “On Inferring TCP Behavior”, *Proc. 2001 ACM SIGCOMM*, San Diego, ago. 2001.
- [Pan 1997]** P. Pan e H. Schulzrinne. “Staged Refresh Timers for RSVP”, *2nd Global Internet Conference*. Phoenix, 1997.
- [Parekh 1993]** A. Parekh e R. Gallagher. “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-Node Case”, *IEEE/ACM Transactions on Networking*, vol. 1, n. 3, jun. 1993, p. 344-357.

- [Partridge 1992]** C. Partridge e S. Pink. "An Implementation of the Revised Internet Stream Protocol (ST-2)", *Journal of Internetworking: Research and Experience*, vol. 3, n. 1, mar. 1992.
- [Partridge 1998]** C. Partridge et al. "A Fifty Gigabit per second IP Router", *IEEE/ACM Transactions on Networking*, vol. 6, n. 3, jun. 1998, p. 237-248.
- [Paxson 1997]** V. Paxson. "End-to-End Internet Packet Dynamics", *Proc. 1997 ACM SIGCOMM '97*, Cannes, set. 1997.
- [Perkins 1994]** A. Perkins. "Networking with Bob Metcalfe", *The Red Herring Magazine*, nov. 1994.
- [Perkins 1998]** C. Perkins, O. Hodson e V. Hardman. "A Survey of Packet Loss Recovery Techniques for Streaming Audio", *IEEE Network Magazine*, set./out. 1998, p. 40-47.
- [Perkins 1998b]** C. Perkins. *Mobile IP: Design Principles and Practice*. Reading: Addison Wesley, 1998.
- [Perkins 2000]** C. Perkins. *Ad-Hoc Networking*. Reading: Addison-Wesley, 2000.
- [Perlman 1999]** R. Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2 ed. Reading: Addison Wesley Professional Computing Series, 1999.
- [PGPI 2009]** The International PGP Home Page, <http://www.pgpi.org>.
- [Phifer 2000]** L. Phifer. "The Trouble with NAT", *The Internet Protocol Journal*, vol. 3, n. 4, dez. 2000, http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html.
- [Piatek 2007]** M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani. "Do Incentives Build Robustness in BitTorrent?", *Proc. NSDI*, 2007.
- [Piatek 2008]** M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson. "One hop Reputations for Peer-to-Peer File Sharing Workloads", *Proc. NSDI*, 2008.
- [Pickholtz 1982]** R. Pickholtz, D. Schilling e L. Milstein. "Theory of Spread Spectrum Communication — a Tutorial", *IEEE Transactions on Communications*, vol. COM-30, n. 5, maio 1982, p. 855-884.
- [Pingplotter 2009]** pingplotter homepage, <http://www.pingplotter.com>.
- [Piscatello 1993]** D. Piscatello e A. Lyman Chapin. "Open Systems Networking", Reading: Addison Wesley, 1993.
- [Point Topic 2006]** Point Topic Ltd. *World Broadband Statistics*, Q1. 2006, <http://www.pointtopic.com>.
- [Potaroo 2009]** "Growth of the BGP Table — 1994 to Present", <http://bgp.potaroo.net/>.
- [PPLive 2009]** PPLive homepage, <http://www.pplive.com>.
- [PriMetrica 2009]** Global Internet Geography, <http://www.telegeography.com/products/gig/index.php>.
- [QuickTime 2009]** QuickTime homepage, <http://www.apple.com/quicktime>.
- [Quittner 1998]** J. Quittner, M. Slatalla. *Speeding the Net: The Inside Story of Netscape and How it Challenged Microsoft*, Atlantic Monthly Press, 1998.
- [Ramakrishnan 1990]** K. K. Ramakrishnan e R. Jain. "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", *ACM Transactions on Computer Systems*, vol. 8, n. 2, maio 1990, p. 158-181.
- [Raman 1999]** S. Raman e S. McCanne. "A Model, Analysis, and Protocol Framework for Soft State-based Communication", *Proceedings of ACM SIGCOMM '99*, Boston, ago. 1999.
- [Raman 2007]** B. Raman, K. Chebrolu. "Experiences in Using WiFi for Rural Internet in India", *IEEE Communications Magazine*, Special Issue on New Directions in Networking Technologies in Emerging Economies, jan. 2007.
- [Ramaswami 1998]** R. Ramaswami e K. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufman Publishers, 1998.
- [Ramjee 1994]** R. Ramjee, J. Kurose, D. Towsley e H. Schulzrinne. "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks", *Proc. 1994 IEEE Infocom*.

- [**Rao 1996**] K. R. Rao, J. J. Hwang. *Techniques and Standards for Image, Video and Audio Coding*, Prentice Hall, Englewood Cliffs, 1996.
- [**RAT 2009**] Robust Audio Tool, <http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/>.
- [**Ratnasamy 2001**] S. Ratnasamy, P. Francis, M. Handley, R. Karp e S. Shenker. “A Scalable Content-Addressable Network”, *Proceedings of ACM SIGCOMM*, San Diego, 2001.
- [**RealNetworks 2009**] RealNetworks homepage, <http://www.realnetworks.com>.
- [**Ren 2006**] S. Ren, L. Guo e X. Zhang. “ASAP: an AS-aware peer-relay protocol for high quality VoIP”, *Proc. 2006 IEEE ICDCS*, Lisboa, jul. 2006.
- [**Rescorla 2001**] E. Rescorla. *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, 2001.
- [**RFC 001**] S. Crocker. “Host Software”, RFC 001 (o primeiríssimo RFC!).
- [**RFC 768**] J. Postel. “User Datagram Protocol”, RFC 768, ago. 1980.
- [**RFC 789**] E. Rosen. “Vulnerabilities of Network Control Protocols”, RFC 789.
- [**RFC 791**] J. Postel. “Internet Protocol: DARPA Internet Program Protocol Specification”, RFC 791, set. 1981.
- [**RFC 792**] J. Postel. “Internet Control Message Protocol”, RFC 792, set. 1981.
- [**RFC 793**] J. Postel. “Transmission Control Protocol”, RFC 793, set. 1981.
- [**RFC 801**] J. Postel. “NCP/TCP Transition Plan”, RFC 801, nov. 1981.
- [**RFC 826**] D. C. Plummer. “An Ethernet Address Resolution Protocol — or — Converting Network Protocol Addresses to 48 bit Ethernet Address for Transmission on Ethernet Hardware”, RFC 826, nov. 1982.
- [**RFC 829**] V. Cerf. “Packet Satellite Technology Reference Sources”, RFC 829, nov. 1982.
- [**RFC 854**] J. Postel e J. Reynolds. “TELNET Protocol Specification”, RFC 854, maio 1993.
- [**RFC 950**] J. Mogul e J. Postel. “Internet Standard Subnetting Procedure”, RFC 950, ago. 1985.
- [**RFC 959**] J. Postel e J. Reynolds. “File Transfer Protocol (FTP)”, RFC 959, out 1985.
- [**RFC 977**] B. Kantor e P. Lapsley. “Network News Transfer Protocol”, RFC 977, fev. 1986.
- [**RFC 1028**] J. Davin, J. D. Case, M. Fedor e M. Schoffstall. “A Simple Gateway Monitoring Protocol”, RFC 1028, nov. 1987.
- [**RFC 1034**] P.V. Mockapetris. “Domain Names — Concepts and Facilities”, RFC 1034, nov. 1987.
- [**RFC 1035**] P. Mockapetris. “Domain Names — Implementation and Specification”, RFC 1035, nov. 1987.
- [**RFC 1058**] C. L. Hendrick. “Routing Information Protocol”, RFC 1058, jun. 1988.
- [**RFC 1071**] R. Braden, D. Borman e C. Partridge. “Computing The Internet Checksum”, RFC 1071, set. 1988.
- [**RFC 1075**] D. Waitzman, C. Partridge e S. Deering. “Distance Vector Multicast Routing Protocol”, RFC 1075, nov. 1988.
- [**RFC 1112**] S. Deering. “Host Extension for IP Multicasting”, RFC 1112, ago. 1989.
- [**RFC 1122**] R. Braden. “Requirements for Internet Hosts — Communication Layers”, RFC 1122, out. 1989.
- [**RFC 1123**] R. Braden (ed.) “Requirements for Internet Hosts — Application and Support”, RFC-1123, out. 1989.
- [**RFC 1142**] D. Oran. “OSI IS-IS Intra-Domain Routing Protocol”, RFC 1142, fev. 1990.
- [**RFC 1190**] C. Topolcic. “Experimental Internet Stream Protocol: Version 2 (ST-II)”, RFC 1190, out. 1990.
- [**RFC 1191**] J. Mogul e S. Deering. “Path MTU Discovery”, RFC 1191, nov. 1990.
- [**RFC 1213**] K. McCloghrie e M.T. Rose. “Management Information Base for Network Management of TCP/IP-based internets: MIB-II”, RFC 1213, mar. 1991.
- [**RFC 1256**] S. Deering. “ICMP Router Discovery Messages”, RFC 1256, set. 1991.
- [**RFC 1320**] R. Rivest. “The MD4 Message-Digest Algorithm”, RFC 1320, abr. 1992.

- [RFC 1321] R. Rivest. "The MD5 Message-Digest Algorithm", RFC 1321, abr. 1992.
- [RFC 1323] V. Jacobson, S. Braden e D. Borman. "TCP Extensions for High Performance", RFC 1323, maio 1992.
- [RFC 1422] S. Kent. "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422.
- [RFC 1547] D. Perkins. "Requirements for an Internet Standard Point-to-Point Protocol", RFC 1547, dez. 1993.
- [RFC 1584] J. Moy. "Multicast Extensions to OSPF", RFC 1584, mar. 1994.
- [RFC 1633] R. Braden, D. Clark e S. Shenker. "Integrated Services in the Internet Architecture: an Overview", RFC 1633, jun. 1994.
- [RFC 1636] R. Braden, D. Clark, S. Crocker e C. Huitema. "Report of IAB Workshop on Security in the Internet Architecture", RFC 1636, nov. 1994.
- [RFC 1661] W. Simpson (ed.). "The Point-to-Point Protocol (PPP)", RFC 1661, jul. 1994.
- [RFC 1662] W. Simpson (ed.). "PPP in HDLC-Like Framing", RFC 1662, jul. 1994.
- [RFC 1700] J. Reynolds e J. Postel. "Assigned Numbers", RFC 1700, out. 1994.
- [RFC 1752] S. Bradner e A. Mankin. "The Recommendations for the IP Next Generation Protocol", RFC 1752, jan. 1995.
- [RFC 1760] N. Haller. "The S/KEY One-Time Password System", RFC 1760, fev. 1995.
- [RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot e E. Lear. "Address Allocation for Private Internets", RFC 1918, fev. 1996.
- [RFC 1930] J. Hawkinson e T. Bates. "Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)", RFC 1930, mar. 1996.
- [RFC 1938] N. Haller e C. Metz. "A One-Time Password System", RFC 1938, maio 1996.
- [RFC 1939] J. Myers e M. Rose. "Post Office Protocol — Version 3", RFC 1939, maio 1996.
- [RFC 1945] T. Berners-Lee, R. Fielding e H. Frystyk. "Hypertext Transfer Protocol — HTTP/1.0", RFC 1945, maio 1996.
- [RFC 2003] C. Perkins. "IP Encapsulation within IP", RFC 2003, out. 1996.
- [RFC 2004] C. Perkins. "Minimal Encapsulation within IP", RFC 2004, out. 1996.
- [RFC 2018] M. Mathis, J. Mahdavi, S. Floyd e A. Romanow. "TCP Selective Acknowledgment Options", RFC 2018, out. 1996.
- [RFC 2050] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg e J. Postel. "Internet Registry IP Allocation Guidelines", RFC 2050, nov. 1996.
- [RFC 2104] H. Krawczyk, M. Bellare e R. Canetti. "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, fev. 1997.
- [RFC 2131] R. Droms. "Dynamic Host Configuration Protocol", RFC 2131, mar. 1997.
- [RFC 2136] P. Vixie, S. Thomson, Y. Rekhter e J. Bound. "Dynamic Updates in the Domain Name System", RFC 2136, abr. 1997.
- [RFC 2153] W. Simpson. "PPP Vendor Extensions", RFC 2153, maio 1997.
- [RFC 2205] R. Braden (ed.), L. Zhang, S. Berson, S. Herzog e S. Jamin. "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, set. 1997.
- [RFC 2210] J. Wroclawski. "The Use of RSVP with IETF Integrated Services", RFC 2210, set. 1997.
- [RFC 2211] J. Wroclawski. "Specification of the Controlled-Load Network Element Service", RFC 2211, set. 1997.
- [RFC 2215] S. Shenker e J. Wroclawski. "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, set. 1997.

- [RFC 2246] T. Dierks e C. Allen. "The TLS Protocol", RFC 2246, jan. 1998.
- [RFC 2253] M. Wahl, S. Kille e T. Howes. "Lightweight Directory Access Protocol (v3)", RFC 2253, dez. 1997.
- [RFC 2284] L. Blunk e J. Vollbrecht. "PPP Extensible Authentication Protocol (EAP)", RFC 2284, mar. 1998.
- [RFC 2326] H. Schulzrinne, A. Rao e R. Lanphier. "Real Time Streaming Protocol (RTSP)", RFC 2326, abr. 1998.
- [RFC 2328] J. Moy. "OSPF Version 2", RFC 2328, abr. 1998.
- [RFC 2420] H. Kummert. "The PPP Triple-DES Encryption Protocol (3DESE)", RFC 2420, set. 1998.
- [RFC 2437] B. Kaliski e J. Staddon. "PKCS #1: RSA Cryptography Specifications, Version 2", RFC 2437, out. 1998.
- [RFC 2453] G. Malkin. "RIP Version 2", RFC 2453, nov. 1998.
- [RFC 2460] S. Deering e R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, dez. 1998.
- [RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss. "An Architecture for Differentiated Services", RFC 2475, dez. 1998.
- [RFC 2578] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Structure of Management Information Version 2 (SMIV2)", RFC 2578, abr. 1999.
- [RFC 2579] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Textual Conventions for SMIV2", RFC 2579, abr. 1999.
- [RFC 2580] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Conformance Statements for SMIV2", RFC 2580, abr. 1999.
- [RFC 2581] M. Allman, V. Paxson e W. Stevens. "TCP Congestion Control", RFC 2581, abr. 1999.
- [RFC 2597] J. Heinanen, F. Baker, W. Weiss e J. Wroclawski. "Assured Forwarding PHB Group", RFC 2597, jun. 1999.
- [RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee e R. Feilding. "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, jun. 1999.
- [RFC 2663] P. Srisuresh e M. Holdrege. "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663.
- [RFC 2702] D. Awdanche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus. "Requirements for Traffic Engineering Over MPLS", set. 1999.
- [RFC 2716] B. Aboba e D. Simon. "PPP EAP TLS Authentication Protocol", RFC 2716, out. 1999.
- [RFC 2733] J. Rosenberg e H. Schulzrinne. "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, dez. 1999.
- [RFC 2827] P. Ferguson e D. Senie. "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing", RFC 2827, maio 2000.
- [RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson. "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, jun. 2000.
- [RFC 2960] R. Stewart, Q. Xie, K. Morneau, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. "Stream Control Transmission Protocol", RFC 2960, out. 2000.
- [RFC 2961] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini. "RSVP Refresh Overhead Reduction Extensions", RFC 2961, abr. 2001.
- [RFC 2988] V. Paxson e M. Allman. "Computing TCP's Retransmission Timer", RFC 2988, nov. 2000.
- [RFC 3007] B. Wellington. "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, nov. 2000.
- [RFC 3022] P. Srisuresh e K. Egevang. "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, jan. 2001.
- [RFC 3031] E. Rosen, A. Viswanathan e R. Callon. "Multiprotocol Label Switching Architecture", RFC 3031, jan. 2001.

- [RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li e A. Conta. "MPLS Label Stack Encoding", RFC 3032, jan. 2001.
- [RFC 3052] M. Eder e S. Nag. "Service Management Architectures Issues and Review", RFC 3052, jan. 2001.
- [RFC 3139] L. Sanchez, K. McCloghrie e J. Saperia. "Requirements for Configuration Management of IP-Based Networks", RFC 3139, jun. 2001.
- [RFC 3168] K. Ramakrishnan, S. Floyd, D. Black. "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, set. 2001.
- [RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan e G. Swallow. "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, dez. 2001.
- [RFC 3221] G. Huston. "Commentary on Inter-Domain Routing in the Internet", RFC 3221, dez. 2001.
- [RFC 3232] J. Reynolds. "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, jan. 2002.
- [RFC 3246] B. Davie, A. Charny, J. C. R. Bennet, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis. "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, mar. 2002.
- [RFC 3260] D. Grossman. "New Terminology and Classifications for DiffServ", RFC 3260, abr. 2002.
- [RFC 3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. "SIP: Session Initiation Protocol", RFC 3261, jul. 2002.
- [RFC 3272] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai. "Overview and Principles of Internet Traffic Engineering", RFC 3272, mai. 2002.
- [RFC 3286] L. Ong, J. Yoakum. "An Introduction to Stream Control Transmission Protocol (SCTP)", RFC 3286, mai. 2002.
- [RFC 3344] C. Perkins, (ed.). "IP Mobility Support for IPv4", RFC 3344, out. 2002.
- [RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai. "Applicability Statement for Traffic Engineering with MPLS", RFC 3346, ago. 2002.
- [RFC 3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner e A. Thyagarajan. "Internet Group Management Protocol, Version 3", RFC 3376, out. 2002.
- [RFC 3390] M. Allman, S. Floyd e C. Partridge. "Increasing TCP's Initial Window", RFC 3390, out. 2002.
- [RFC 3410] J. Case, R. Mundy, D. Partain e D. Partain. "Introduction and Applicability Statements for Internet Standard Management Framework", RFC 3410, dez. 2002.
- [RFC 3411] D. Harrington, R. Presuhn e B. Wijnen. "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", RFC 3411, dez. 2002.
- [RFC 3414] U. Blumenthal. "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 3414, dez. 2002.
- [RFC 3415] B. Wijnen, R. Presuhn e K. McCloghrie. "View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 3415, dez. 2002.
- [RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose e S. Waldbusser. "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", dez. 2002.
- [RFC 3417] R. Presuhn. "Transport Mappings for the Simple Network Management Protocol", (SNMP), RFC 3417, dez. 2002.
- [RFC 3439] R. Bush e D. Meyer. "Some Internet Architectural Guidelines and Philosophy", RFC 3439, dez. 2003.
- [RFC 3468] L. Andersson e G. Swallow. "The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols", RFC 3468, fev. 2003.
- [RFC 3469] V. Sharma e F. Hellstrand (eds.). "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery", RFC 3469, fev. 2003.

- [RFC 3501] M. Crispin. "Internet Message Access Protocol — Version 4rev 1", RFC 3501, mar. 2003.
- [RFC 3550] H. Schulzrinne, S. Casner, R. Frederick e V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, jul. 2003.
- [RFC 3569] S. Bhattacharyya (ed.). "An Overview of Source-Specific Multicast (SSM)", RFC 3569, jul. 2003.
- [RFC 3618] B. Fenner, D. Meyer (ed.). "Multicast Source Discovery Protocol (MSDP)", RFC 3618, out. 2003.
- [RFC 3649] S. Floyd. "High Speed TCP for Large Congestion Windows", RFC 3649, dez. 2003.
- [RFC 3782] S. Floyd, T. Henderson, A. Gurtov. "The New Reno Modification to TCP's Fast Recovery Algorithm", RFC 3782, abr. 2004.
- [RFC 3973] A. Adams, J. Nicholas, W. Siadak. "Protocol Independent Multicast — Dense Mode (PIM-DM): Protocol Specification (Revisado)", RFC 3973, jan. 2005.
- [RFC 4022] R. Raghunarayan (ed.). "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, mar. 2005.
- [RFC 4113] B. Fenner, J. Flick. "Management Information Base for the User Datagram Protocol (UDP)", RFC 4113, jun. 2005.
- [RFC 4213] E. Nordmark, R. Gilligan. "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, out. 2005.
- [RFC 4271] Y. Rekhter, T. Li, S. Hares (ed.). "A Border Gateway Protocol 4 (BGP-A)", RFC 4271, jan. 2006.
- [RFC 4272] S. Murphy. "BGP Security Vulnerabilities Analysis", RFC 4274, jan. 2006.
- [RFC 4274] Meyer, D. e K. Patel. "BGP-4 Protocol Analysis", RFC 4274, jan. 2006.
- [RFC 4276] S. Hares e A. Retana. "BGP 4 Implementation Report", RFC 4276, jan. 2006.
- [RFC 4291] R. Hinden, S. Deering. "IP Version 6 Addressing Architecture", RFC 4291, fev. 2006.
- [RFC 4293] S. Routhier (ed.). "Management Information Base for the Internet Protocol (IP)", RFC 4293, abr. 2003.
- [RFC 4301] S. Kent, K. Seo. "Security Architecture for the Internal Protocol", RFC 4301, dez. 2005.
- [RFC 4302] S. Kent. "IP Authentication Header", RFC 4302, dez. 2005.
- [RFC 4303] S. Kent. "IP Encapsulating Security Payload (ESP)", RFC 4303, dez. 2005.
- [RFC 4305] D. Eastlake. "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305. Dez. 2005.
- [RFC 4340] E. Kohler, M. Handley, S. Floyd. "Datagram Congestion Control Protocol (DCCP)", RFC 4340, mar. 2006.
- [RFC 4443] A. Conta, S. Deering, M. Gupta (ed.). "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, mar. 2006.
- [RFC 4346] T. Dierks, E. Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, abr. 2006.
- [RFC 4502] S. Waldbusser. "Remote Networking Monitoring Management Information Base Version 2", RFC 4502, mai. 2006.
- [RFC 4601] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas. "Protocol Independent Multicast — Sparse Mode (PIM-SM): Protocol Specification (Revisado)", RFC 4601, ago. 2006.
- [RFC 4607] H. Holbrook, B. Cain. "Source-Specific Multicast for IP", RFC 4607, ago. 2006.
- [RFC 4611] M. McBride, J. Meylor, D. Meyer. "Multicast Source Discovery Protocol (MSDP) Deployment Scenarios", RFC 4611, ago. 2006.
- [RFC 4632] V. Fuller, T. Li. "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, ago. 2006.

- [RFC 5000] RFC editor. "Internet Official Protocol Standards", RFC 5000, mai. 2008.
- [RFC 5110] P. Savola. "Overview of the Internet Multicast Routing Architecture", RFC 5110, jan. 2008.
- [RFC 5218] D. Thaler, B. Aboba. "What Makes for a Successful Protocol?", RFC 5218, jul. 2008.
- [RFC 5321] J. Klensin. "Simple Mail Transfer Protocol", RFC 5321, out. 2008.
- [RFC 5322] P. Resnick (ed.). "Internet Message Format", RFC 5322, out. 2008.
- [RFC 5348] S. Floyd, M. Handley, J. Padhye, J. Widmer. "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, set. 2008.
- [RFC 5411] J. Rosenberg. "A Hitchhiker's Guide to the Session Initiation Protocol (SIP)", RFC 5411, fev. 2009.
- [Rhee 1998] I. Rhee. "Error Control Techniques for Interactive Low-Bit Rate Video Transmission over the Internet", *Proc. 1998 ACM SIGCOMM*, Vancouver, 31 ago.-4 set. 1998.
- [Roberts 1967] L. Roberts e T. Merril. "Toward a Cooperative Network of Time-Shared Computers", *AFIPS Fall Conference*, out. 1966.
- [Roberts 2004] J. Roberts. "Internet Traffic, QoS and Pricing", *Proceeding of the IEEE*, vol. 92, n. 9, set. 2004, p. 1389-1399.
- [Rom 1990] R. Rom e M. Sidi. *Multiple Access Protocols: Performance and Analysis*. Nova York: Springer-Verlag, 1990.
- [Root-servers 2009] <http://www.root-servers.org/>.
- [Rose 1996] M. Rose. *The Simple Book: An Introduction to Internet Management*, 2 ed. rev. Englewood Cliffs: Prentice Hall, 1996.
- [Rosenberg 2000] J. Rosenberg, L. Qiu e H. Schulzrinne. "Integrating Packet FEC into Adaptive Playout Buffer Algorithms on the Internet", *IEEE INFOCOM 2000*. Tel Aviv, 2000.
- [Ross 1995] K. W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Berlim: Springer, 1995.
- [Ross 2009] K.W. Ross. PowerPoint Slides on Network Security, <http://cis.poly.edu/~ross>.
- [Rowston 2001] A. Rowston e P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems." *Proceedings of IFIP/ACM Middleware 2001*. Heidelberg, 2001.
- [RSA 1978] R. L. Rivest, A. Shamir e L. M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21, n. 2, fev. 1978, p. 120-126.
- [RSA Fast 2009] RSA Laboratories. "How Fast Is RSA?", <http://www.rsa.com/rsalabs/node.asp?id=2215>.
- [RSA Key 2009] RSA Laboratories. "How Large a Key Should Be Used in the RSA Crypto System?", <http://www.rsa.com/rsalabs/node.asp?id=2218>.
- [Rubenstein 1998] D. Rubenstein e J. Kurose, D. Towsley. "Real-Time Reliable Multicast Using Proactive Forward Error Correction", *Proceedings of NOSSDAV '98*, Cambridge, jul. 1998.
- [Rubin 2001] A. Rubin. *White-Hat Security Arsenal: Tackling the Threats*. Addison Wesley, 2001.
- [Ruiz-Sánchez 2001] M. Ruiz-Sánchez, E. Biersack, W. Dabbous. "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine*, v. 15, n. 2, p. 8-23.
- [Saltzer 1984] J. Saltzer, D. Reed e D. Clark. "End-to-End Arguments in System Design", *ACM Transactions on Computer Systems (TOCS)*, vol. 2, n. 4, nov. 1984.
- [Saroiu 2002] S. Saroiu, P. K. Gummadi, S. D. Gribble. "A Measurement Study of Peer-to-Peer File Sharing Systems", *Proc. of Multimedia Computing and Networking (MMCN)*, 2002.
- [Saroiu 2002b] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble e H. M. Levy. "An Analysis of Internet Content Delivery Systems", *USENIX OSDI 2002*.
- [Saydam 1996] T. Saydam e T. Magedanz. "From Networks and Network Management into Service and Service Management", *Journal of Networks and System Management*, vol. 4, n. 4, dez. 1996, p. 345-348.

- [Schiller 2003]** J. Schiller. *Mobile Communications 2 ed*, Addison Wesley, 2003.
- [Schneier 1995]** B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1995.
- [Schulzrinne 1997]** H. Schulzrinne. “A Comprehensive Multimedia Control Architecture for the Internet”, NOSSDAV97 (*Network and Operating System Support for Digital Audio and Video*), St. Louis, 19 maio 1997.
- [Schulzrinne-RTP 2009]** Henning Schulzrinne’s RTP site, <http://www.cs.columbia.edu/~hgs/rtp>.
- [Schulzrinne-RTSP 2009]** Henning Schulzrinne’s RTSP site, <http://www.cs.columbia.edu/~hgs/rtsp>.
- [Schulzrinne-SIP 2009]** Henning Schulzrinne’s SIP site, <http://www.cs.columbia.edu/~hgs/sip>.
- [Schwartz 1977]** M. Schwartz. *Computer-Communication Network Design and Analysis*. Englewood Cliffs: Prentice-Hall, 1997.
- [Schwartz 1980]** M. Schwartz. *Information, Transmission, Modulation, and Noise*. Nova York: McGraw Hill, 1980.
- [Schwartz 1982]** M. Schwartz. “Performance Analysis of the SNA Virtual Route Pacing Control”, *IEEE Transactions on Communications*, vol. 30, n. 1, jan. 1982, p. 172-184.
- [Scourias 2009]** J. Scourias. “Overview of the Global System for Mobile Communications: GSM”, <http://www.privateline.com/PCS/GSM0.html>
- [Segaller 1998]** S. Segaller. *Nerds 2.0.1, A Brief History of the Internet*. Nova York: TV Books, 1998.
- [Shacham 1990]** N. Shacham e P. McKenney. “Packet Recovery in High-Speed Networks Using Coding and Buffer Management”, *Proceedings of IEEE Infocom Conference*. São Francisco, 1990, p. 124-131.
- [Sharma 2003]** P. Sharma, E. Perry, R. Malpani. “IP Multicast Operational Network Management: Design, Challenges, and Experiences”, *IEEE Network Magazine*, mar. 2003, p. 49-55.
- [Sidor 1998]** D. Sidor. “TMN Standards: Satisfying Today’s Needs While Preparing for Tomorrow”, *IEEE Communications Magazine*, vol. 36, n. 3, mar. 1998, p. 54-64.
- [SIP Software 2009]** H. Schulzrinne. Software Package site, <http://www.cs.columbia.edu/IRT/software>.
- [Skoudis 2004]** E. Skoudis, L. Zeltser. *Malware: Fighting Malicious Code*, Englewood Cliffs: Prentice Hall, 2004.
- [Skoudis 2006]** E. Skoudis, T. Liston. *Counter Hack Reloaded: A Step-by-step Guide to Computer Attacks and Effective Defenses*, 2 ed., Englewood Cliffs: Prentice Hall, 2006.
- [Skype 2009]** Skype homepage, www.skype.com.
- [SMIL 2009]** W3C Synchronized Multimedia homepage, <http://www.w3.org/AudioVideo>.
- [Snort 2009]** Sourcefire Inc., Snort homepage, <http://www.snort.org/>.
- [Solari 1997]** S. J. Solari. *Digital Video and Audio Compression*. Nova York: McGraw Hill, 1997.
- [Solensky 1996]** F. Solensky. “IPv4 Address Lifetime Expectations”, In: S. Bradner e A. Mankin (eds.) *IPng: Internet Protocol Next Generation*. Reading: Addison-Wesley, 1996.
- [Spragins 1991]** J. D. Spragins. *Telecommunications Protocols and Design*. Reading: Addison Wesley, 1991.
- [Sprint 2009]** Sprint Corp., “Dedicated Internet Access Service Level Agreements”, http://sprint.com/business/resources/dedicated_internet_access.pdf.
- [Srikant 2004]** R. Srikant. *The Mathematics of Internet Congestion Control*, Birkhauser, 2004
- [Srinivasan 1999]** V. Srinivasan e G. Varghese. “Fast Address Lookup Using Controlled Prefix Expansion”, *ACM Transactions Computer Sys.* vol. 17, n. 1, fev. 1999, p. 1-40.
- [Sripanidkulchai 2004]** K. Sripanidkulchai, B. Maggs e H. Zhang. “An analysis of live streaming workloads on the Internet”, *Proc. 4th ACM SIGCOMM Internet Measurement Conference*, Taormina, p. 41-54, 2004.
- [Stallings 1993]** W. Stallings. *SNMP, SNMP v2, and CMIP: The Practical Guide to Network Management Standards*. Reading: Addison Wesley, 1993.

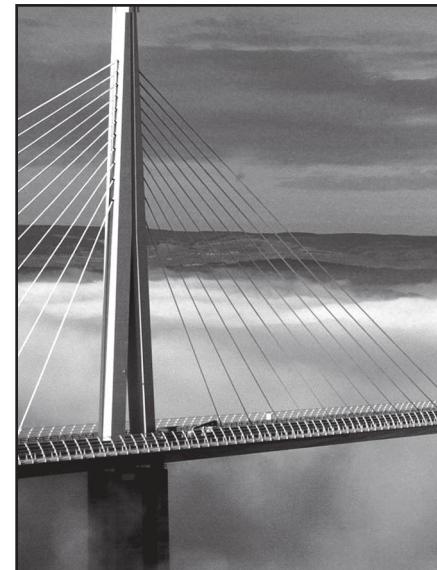
- [Stallings 1999]** W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading: Addison Wesley, 1999.
- [Steinder 2002]** M. Steinder e A. Sethi. "Increasing Robustness of Fault Localization through Analysis of Lost, Spurious, and Positive Symptoms", *Proc. 2002 IEEE Infocom*
- [Stevens 1990]** W. R. Stevens. *Unix Network Programming*. Englewood Cliffs: Prentice-Hall, 1990.
- [Stevens 1994]** W. R. Stevens. *TCP/IP Illustrated, vol. 1: The Protocols*. Reading: Addison-Wesley, 1994.
- [Stevens 1997]** W. R. Stevens. *Unix Network Programming, volume 1: Networking APIs-Sockets and XTI*, 2ed. Englewood Cliffs: Prentice-Hall, 1997.
- [Stewart 1999]** J. Stewart. *BGP4: Interdomain Routing in the Internet*. Addison-Wesley, 1999.
- [Stoica 2001]** I. Stoica, R. Morris, D. Karger, M. F. Kaashoek e H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. 2001 ACM SIGCOMM 2001*, San Diego, 2001.
- [Stone 1998]** J. Stone, M. Greenwald, C. Partridge e J. Hughes. "Performance of Checksums and CRC's Over Real Data", *IEEE/ACM Transactions on Networking*, vol. 6, n. 5, out. 1998, p. 529-543.
- [Stone 2000]** J. Stone e C. Partridge. "When Reality and the Checksum Disagree", *Proc. 2000 ACM SIGCOMM*, Estocolmo, ago. 2000.
- [Strayer 1992]** W. T. Strayer, B. Dempsey e A. Weaver. *XTP: The Xpress Transfer Protocol*. Reading: Addison Wesley, 1992.
- [Stubblefield 2002]** A. Stubblefield, J. Ioannidis e A. Rubin. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proceedings of the 2002 Network and Distributed Systems Security Symposium*, 2002, p. 17-22.
- [Subramanian 2000]** M. Subramanian. *Network Management: Principles and Practice*. Reading: Addison-Wesley, 2000.
- [Subramanian 2002]** L. Subramanian, S. Agarwal, J. Rexford e R. Katz. "Characterizing the Internet Hierarchy from Multiple Vantage Points", *Proc. 2002 IEEE Infocom*.
- [Sundaresan 2006]** K. Sundaresan, K. Papagiannaki. "The Need for Cross-layer Information in Acces Point Selection", *Proc. 2006 ACM Internet Measurement Conference, Rio de Janeiro*, out. 2006.
- [Su 2006]** A.-J. Su, D. Choffnes, A. Kuzmanovic, F. Bustamante. "Drafting Behind Akamai", *ACM SIGCOMM*, set. 2006.
- [Suh 2006]** K. Suh, D. R. Figueiredo, J. Kurose e D. Towsley. "Characterizing and detecting relayed traffic: A case study using Skype", *Proc. 2006 IEEE Infocom*, Barcelona, abr. 2006.
- [Sunshine 1978]** C. Sunshine e Y. K. Dalal. "Connection Management in Transport Protocols", *Computer Networks*. Amesterdã, 1978.
- [TechnOnLine 2009]** TechOnLine. "Protected Wireless Networks", Online Webcast Tutorial. http://www.techonline.com/community/tech_topic/internet/21752.
- [Thaler 1997]** D. Thaler e C. Ravishankar. "Distributed Center-Location Algorithms", *IEEE Journal on Selected Areas in Communications*, vol. 15, n. 3, abr. 1997, p. 291-303.
- [Think 2009]** Technical History of Network Protocols. "Cyclades", <http://www.cs.utexas.edu/users/chris/think/Cyclades/index.htm>.
- [Thottan 1998]** M. Thottan e C. Ji. "Proactive Anomaly Detection Using Distributed Intelligent Agents", *IEEE Network Magazine*, vol. 12, n. 5, set./out. 1998, p. 21-28.
- [Tobagi 1990]** F. Tobagi. "Fast Packet Switch Architectures for Broadband Integrated Networks", *Procs. of the IEEE*, vol. 78, n. 1, jan. 1990, p. 133-167.
- [TOR 2009]** Tor: Anonymity Online, <http://www.torproject.org>.
- [Turner 1988]** J. S. Turner. "Design of a Broadcast Packet Switching Network", *IEEE Transactions on Communications*, vol. 36, n. 6, jun. 1988, p. 734-743.
- [Turner 2009]** B. Turner. "2G, 3G, 4G Wireless Tutorial", <http://blogs.nmscommunications.com/communications/2008/10/2g-3g-4g-wireless-tutorial.html>.

- [UPnP Forum 2009]** UPnP Forum homepage, <http://www.upnp.org/>.
- [van der Berg 2008]** R. van der Berg. "How the Net Works: an introduction to peering and transit", <http://arstechnica.com/guides/other/peering-and-transit.ars>.
- [Varghese 1997]** G. Varghese e A. Lauck. "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility", *IEEE/ACM Transactions on Networking*, vol. 5, n. 6, dez. 1997, p. 824-834.
- [Vasudevan 2006]** S. Vasudevan, C. Diot, J. Kurose, D. Towsley. "Facilitating Access Point Selection in IEEE 802.11 Wireless Networks", *Proc. 2005 ACM Internet Measurement Conference*, San Francisco, out. 2005.
- [Verizon 2009]** Verizon, "US Products and Services", <http://www.verizonbusiness.com/terms/us/products/>.
- [Verizon FIOS 2009]** FIOS FAQ, <http://www22.verizon.com/Residential/FIOSIntenet/FAQ/FAQ.htm>.
- [Verma 2001]** D. C. Verma. *Content Distribution Networks: An Engineering Approach*. John Wiley, 2001.
- [Villamizar 1994]** C. Villamizar, C. Song. "High performance tcp in ansnet", *ACM SIGCOMM Computer Communications Review*, vol. 24, n. 5, 1994, p. 45-60.
- [Viterbi 1995]** A. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Reading: Addison-Wesley, 1995.
- [Voydock 1983]** V. L. Voydock e S. T. Kent. "Security Mechanisms in High-Level Network Protocols", *ACM Computing Surveys*, vol. 15, n. 2, jun. 1983, p. 135-171.
- [W3C 1995]** The World Wide Web Consortium. "A Little History of the World Wide Web", 1995. <http://www.w3.org/History.html>.
- [Wakeman 1992]** Ian Wakeman, Jon Crowcroft, Zheng Wang e Dejan Sirovica. "Layering Considered Harmful", *IEEE Network*, jan. 1992, p. 20-24.
- [Waldvogel 1997]** M. Waldvogel et al. "Scalable High Speed IP Routing Lookup", *Proceedings of ACM SIGCOMM '97*, Cannes, set. 1997, <http://www.acm.org/sigs/sigcomm/sigcomm97/papers/p182.html>.
- [Walker 2000]** J. Walker. "IEEE P802.11 Wireless LANs, Unsafe at Any Key Size; An Analysis of the WEP Encapsulation", out. 2000. <http://www.drizzle.com/~aboba/IEEE/0-362.zip>.
- [Wall 1980]** D. Wall, *Mechanisms for Broadcast and Selective Broadcast*, Ph.D thesis, Stanford University, jun. 1980.
- [Wang 2004]** B. Wang, J. Kurose, P. Shenoy, D. Towsley. "Multimedia Streaming via TCP: An Analytic Performance Study", *Proc. ACM Multimedia Conf.*, Nova York, out. 2004.
- [Weatherspoon 2000]** S. Weatherspoon. "Overview of IEEE 802.11b Security", *Intel Technology Journal*, 2nd Quarter 2000. http://download.intel.com/technology/itj/q22000/pdf/art_5.pdf.
- [Wei 2005]** W. Wei, B. Wang, J. Kurose e D. Towsley. "Classification of Access Network Types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup?", *Proc. 2005 IEEE Infocom*, abr. 2005.
- [Wei 2006]** W. Wei, C. Zang, J. Kurose, D. Towsley. "Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks", *Proc. Active and Passive Measurement Workshop*, Adelaide, mar. 2006.
- [Wei 2007]** D. X. Wei, C. Jin, S. H. Low, S. Hedge. "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Transactions on Networking*, 2007.
- [Weiser 1991]** M. Weiser. "The Computer for the Twenty-First Century", *Scientific American*, set. 1991: 94-10, <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>.
- [Wigle.net 2009]** Wireless Geographic Logging Engine, <http://www.wigle.net>.
- [Williams 1993]** R. Williams. "A Painless Guide to CRC Error Detection Algorithms", <http://www.ross.net/crc/crcpaper.html>.
- [WiMax Forum 2009]** WiMax Forum, <http://www.wimaxforum.org>.
- [Wireshark 2009]** Wireshark homepage, <http://www.wireshark.org>.

- [Wischik 2005]** D. Wischik, N. McKeown. "Part I: Buffer Sizes for Core Routers", *ACM SIGCOMM Computer Communications Review*, vol. 35, n. 3, jul. 2005.
- [Woo 1994]** T. Woo, R. Bindignavle, S. Su e S. Lam. "SNP: an Interface for Secure Network Programming", *Proceedings of 1994 Summer USENIX*, Boston, jun. 1994, p. 45-58.
- [Wood 2009]** L. Wood. "Lloyds Satellites Constellations", <http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/iridium.html>.
- [Xanadu 2009]** Xanadu Project homepage, <http://www.xanadu.com/>.
- [Xiao 2000]** X. Xiao, A. Hannan, B. Bailey e L. Ni. "Traffic Engineering with MPLS in the Internet", *IEEE Network*, mar./abr. 2000.
- [Xie 2008]** H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz. "P4P: Provider Portal for Applications", *Proc. ACM SIGCOMM*, Seattle, ago. 2008.
- [Yannuzzi 2005]** M. Yannuzzi, X. Masip-Bruin, O. Bonaventure. "Open Issues in Interdomain Routing: A Survey", *IEEE Network Magazine*, nov/dez. 2005.
- [Yavatkar 1994]** R. Yavatkar, N. Bhagwat. "Improving End-to-End Performance of TCP over Mobile Internetworks", *Proc. Mobile 94 Workshop on Mobile Computing Systems and Applications*, dez. 1994.
- [Youtube 2009]** Youtube homepage, www.youtube.com.
- [Yu 2006]** H. Yu, M. Kaminsky, O. B. Gibbons e A. Flaxman. "SybilGuard: Defending Against Sybil Attacks via Social Networks", *Proc. 2006 ACM*, Pisa, set. 2006.
- [Zegura 1997]** E. Zegura, K. Calvert e M. Donahoo. "A Quantitative Comparison of Graph-based Models for Internet Topology", *IEEE/ACM Transactions on Networking*, vol. 5, n. 6, dez. 1997, <http://www.cc.gatech.edu/fac/Ellen.Zegura/papers/ton-model.ps.gz>. Veja também <http://www.cc.gatech.edu/projects/gtim>, para um pacote de software que gera redes com estrutura realística.
- [Zhang 1993]** L. Zhang, S. Deering, D. Estrin, S. Shenker e D. Zappala. "RSVP: A New Resource Reservation Protocol", *IEEE Network Magazine*, vol. 7, n. 9, set. 1993, p. 8-18.
- [Zhao 2004]** B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph e J. Kubiatowicz. "Tapestry: A Resilient Global-scale Overlay for Service Deployment", *IEEE Journal on Selected Areas in Communications*, vol. 22, n. 1, jan. 2004, http://www.cs.berkeley.edu/~adj/publications/paper-files/tapestry_jsac.pdf.
- [Zimmermann 1980]** H. Zimmermann. "OSI Reference Model-The Iso Model of Architecture for Open Systems Connection", *IEEE Transactions on Communications*, vol. 28, n. 4, abr. 1980, p. 425-432
- [Zimmermann 2009]** P. Zimmermann. "Why do you need PGP?", <http://www.pgpi.org/doc/whypgp/en/>.



Índice remissivo



- Veja também acesso múltiplo
a cabo
dial-up
linha digital de assinante (DSL)
Ethernet
fiber-to-the-home (FTTH)
operadora local
acesso sem fio em amplas áreas
Wi-Fi
WiMax
Ponto de acesso (AP)
Campo de número de reconhecimento,
TCP
Reconhecimentos
cumulativos
duplicados
retransmissão rápida
perdido
negativo (NAK)
números
piggyback
positivo (ACK)
transferência de dados confiável
(rdt) e,
Protocolo de Controle de
Transmissão (TCP)
rede ótica ativa (AON)
Gerenciamento ativo de fila (AQM)
Redes sem fio ad hoc
Controle de congestionamento aumen-
to aditivo, diminuição multiplicativa
(AIMD)
Agregação de endereço
- Campos de endereço
Endereço indireto
Protocolo de Resolução de Endereços
(ARP)
endereçamento da camada de
enlace
pacote
envio de um datagrama para um
nó
endereço aos cuidados de
blocos
de broadcast
estabelecimento de ligação e, SIP,
classe cheia
Roteamento Interdomínio sem
Classe (Classless Interdomain Routing
CIDR),
Notação decimal separada por
pontos,
Protocolo de Configuração
Dinâmica de Hospedeiros (Dynamic
Host Configuration Protocol DHCP),
externos
hospedeiro
interface
Protocolo da Internet (IP)
IPv4
IPv6
independência das camadas e,
camada de enlace,
MAC
gerenciamento móvel e,
- tradução de endereços na rede
(NAT),
camada de rede
permanente
prefixo
privado
domínio
Protocolo de Inicialização de
Sessão (Session Initiation Protocol
SIP),
sub-redes,
temporário
UPnP
Agência de Projetos de Pesquisa
Avançada (ARPA),
Descoberta de agente,
Tempo de envelhecimento,
Apelidos, servidor de correio,
Algoritmos, veja algoritmos de rote-
mento,
Protocolos ALOHA,
Protocolo bit alternante,
Sistema baseado em anomalia,
Endereço anycast,
Gateways de aplicação,
Camada de aplicação,
processo de endereçamento,
Interface de Programação de
Aplicação (API)
sistema de nomes de domínio
(DNS),
correio eletrônico (e-mail),

Protocolo de Transferência de Arquivo (FTP)
 HTTP (HyperText Transfer Protocol Protocolo de Transferência de Hipertexto),
 mensagem,
 aplicações da rede,
 arquitetura da rede,
 ponto a ponto (P2P),
 comunicação entre processos,
 protocolos,
 transferência de dados confiável,
 Camada de Sockets Seguros (SSL),
 segurança,
 Protocolo Simples de Transferência de Correio (SMTP),
 programação de socket,
 vazão,
 temporização,
 Protocolo de Controle de Transmissão (TCP),
 serviços de transporte,
 Protocolo de Datagrama do Usuário (User Datagram Protocol UDP),
 World Wide Web,
 Interface de Programação de Aplicação (API),
 Roteadores de borda de área,
 Protocolo ATM (modo de transferência assíncrono asynchronous transfer mode),
 Sistemas de áudio,
 buffer cliente,
 compressão,
 aplicações FTP (Protocolo de Transferência de Arquivo),
 serviços de fluxo contínuo da Internet,
 variação,
 ao vivo,
 transdutor,
 recuperação de perda de pacote,
 protocolo de fluxo contínuo em tempo real,
 reparação de correntes realizada no receptor,
 armazenado,
 acesso ao servidor de fluxo contínuo,
 fluxo contínuo,
 acesso ao servidor Web,
 Protocolo de Autenticação de Cabeçalho (AH),
 Chave de autenticação,
 Servidores de nomes com autoridade
 Protocolos ARQ (Automatic Repeat reQuest —ARQ),
 reconhecimentos (ACK),
 bit alternante,
 reconhecimentos negativos (NAK),
 pare e espere,
 Sistemas autônomos (AS),
 Protocolo de Roteador de Borda (BGP),
 roteadores de borda,
 roteamento de batata quente,
 protocolo de roteamento inter-AS
 roteamento intra-ASOSPF,
 protocolos de algoritmo de roteamento,
 RIP (Protocolo de Informação de Roteamento - Routing Information Protocol),
 serviço de taxa de bits disponível (ABR),
 Vazão média,
 Área de backbone, OSPF,
 Largura de banda,
 frequência,
 busca,
 escalabilidade, RTCP,
 aplicações sensíveis,
 Estação-base,
 controlador (BSC),
 sistema (BSS),
 transceptor (BTS),
 infraestrutura sem fio,
 Regras Básicas de Codificação (Basic Encoding Rules BER)
 Quadros de sinalização,
 Equação Bellman-Ford,
 Serviço de melhor esforço,
 redes de distribuição de conteúdo (CDN),
 sistema de entrega,
 dimensionando redes,
 conexões fim a fim (atraso),
 aplicações de protocolo da Internet (IP),
 eliminação da variação,
 limitações de,
 recuperação de perda (de pacote),
 Transferência bidirecional de dados,
 Taxa de erro de bits (BER),
 Erros de bit,
 Gerador de bit,
 BitTorrent,

Cifras de bloco,
 Bloco, endereçamento IP
 Bluetooth (IEEE 802.15.1),
 Protocolo de Roteador de Borda (Border Gateway Protocol BGP),
 sessão externa (eBGP),
 roteamento entre sistemas autônomos,
 sessão interna (iBGP),
 atributos de caminho,
 Atraso de propagação de canal,
 pares,
 prefixos,
 seleção de rota,
 política de roteamento,
 Botnets,
 Enlace de gargalo,
 endereço de broadcast,
 Canais de broadcast,
 algoritmo,
 inundação,
 abordagem unicast de N caminhos
 repasse pelo caminho inverso
 (reverse path forwarding — RPF),
 spanning trees,
 Serviço de corrente de bytes,
 Byte stuffing,
 Redes a cabo,
 Caches,
 Cifra de César,
 Estabelecimento de chamada, SIP
 Configuração de chamada, garantia de QoS de,
 Endereço aos cuidados de (COA),
 Acesso múltiplo com detecção de portadora (CSMA)
 prevenção de colisão (CSMA/CA),
 detecção de colisão (CSMA/CD)
 aplicações Ethernet,
 protocolo da camada de enlace,
 aplicações de LAN sem fio (Wi-Fi),
 Torres celulares,
 Celulares,
 Sistemas celulares,
 classificações de geração,
 padrões do Sistema Global para Comunicações Móveis (GSM).
 transferência (handoffs),
 registro nativo de localização (home location register HLR),
 central de comutação de unidade móvel (MSC),

- rede pública terrestre móvel nativa (PLMN)
 acesso à Internet,
 extensão da Internet, (2.5 e 3G),
 gerenciamento da mobilidade,
 arquitetura da rede
 paging,
 roteando chamadas,
 Servidor de Nó de Suporte GPRS (SGSN).
 estações do transceptor,
 registro de localização de visitantes (visitor location register VLR)
 conexões por voz (2G),
 celular móvel (3G) versus LAN sem fio,
 acesso múltiplo sem fio,
 Construção com centro,
 Central telefônica (CT),
 Autoridade certificadora,
 Protocolos de divisão de canal,
 Atraso de propagação de canal,
 Soma de verificação,
 cabeçalho do datagrama,
 Internet,
 camada de enlace,
 transferência de dados confiável,
 segmentos TCP,
 camada de transporte,
 segmentos UDP,
 Velocidade de chipping, CDMA,
 Pacote de congestionamento (Choke packet),
 Encadeamento de bloco de cifra (CBC)
 Texto cifrado,
 Algoritmos de comutação de circuito,
 Comutação de circuito,
 movimento de dados e,
 conexões fim a fim,
 multiplexação,
 comutação de pacote comparada a, algoritmo,
 períodos de silêncio,
 multiplexação estatística,
 Endereçamento de classes cheias,
 Roteamento Interdomínio sem Classes (Classless Interdomain Routing CIDR),
 Pronto para Envio (Clear to Send – CTS)
 Buffer cliente,
 Programa cliente,
 Arquitetura cliente-servidor,
 camadas de aplicação,
- aplicações Java,
 programação de sockets,
 Protocolo de Controle de Transmissão (TCP),
 Protocolo de Datagrama do Usuário (User Datagram Protocol UDP),
 Clientes,
 Cabo coaxial,
 Acesso Múltiplo por Divisão de Código (Code Division Multiple Access CDMA),
 Colisões, veja acesso múltiplo com detecção de portadora (CSMA); acesso múltiplo
 Compressão de áudio/vídeo,
 Redes de computadores, Veja também acesso à Internet,
 camada de aplicação,
 arquitetura,
 ataques,
 cabo,
 comutação de circuito,
 programa cliente,
 núcleo,
 atraso,
 dial-up,
 linha digital do assinante (DSL),
 sistemas finais (hospedeiros),
 Ethernet,
 fiber-to-the-home (FTTH),
 história das,
 Provedores de Serviço da Internet (ISP),
 arquitetura em camadas,
 multiplexação,
 perda de pacote,
 comutação de pacote,
 meio físico,
 camadas de protocolo,
 protocolo,
 segurança,
 programa servidor,
 vazão,
 acesso sem fio em amplas áreas
 Wi-Fi,
 WiMAX,
 Mecanismo GET condicional,
 Controle de congestionamento,
 aumento aditivo, diminuição multiplicativa (AIMD),
 algoritmo,
 modo de transferência assíncrono (ATM),
- serviço de taxa de bits disponível (ABR),
 contenção,
 busca de largura de banda,
 enlaces de gargalo,
 causas e consequências do, bit de indicação de congestionamento (CI),
 fim a fim,
 justiça,
 recuperação rápida,
 assistido pela rede,
 conexões TCP paralelas,
 células de gerenciamento de recursos (resource-management cells - RM)
 partida lenta e,
 vazão e,
 intervalo de tempo e,
 Protocolo de Controle de Transmissão (TCP),
 camada de transporte,
 Protocolo de Datagrama do Usuário (User Datagram Protocol UDP),
 janela (cwnd),
 Serviços orientados a conexão,
 demultiplexação,
 multiplexação,
 camada de rede,
 gerenciamento TCP,
 camada de transporte,
 redes de circuito virtual (CV),
 Reversão de conexão,
 Serviços não orientados para conexão,
 soma de verificação,
 rede de datagramas,
 estrutura do segmento,
 camada de transporte,
 aplicações UDP,
 Serviço de rede de taxa constante de bits (constant bit rate CBR),
 Memórias de conteúdo endereçável (CAM),
 Redes de distribuição de conteúdo (CDN),
 Conexão de controle,
 Inundação controlada,
 Cookies,
 Correspondente,
 Acordo criptográfico, IPsec,
 Criptografia,
 blocos de cifras,
 cifra de César,

encadeamento de bloco de cifra (CBC),
 texto cifrado,
 algoritmo de decodificação,
 algoritmo de criptografia,
 funções de hash,
 cifra monoalfabética,
 cifra polialfabética,
 chave privada,
 criptografia de chave pública,
 chave pública,
 algoritmo RSA,
 chave de sessão,
 sistema de chave simétrica,
 Reconhecimento cumulativo,
 Cliente,
 Verificação de redundância cíclica (CRC)
 Conexão de dados,
 Integridade dos dados, IPsec,
 Redes de datagrama,
 serviço não orientado para conexão de,
 interface de enlace,
 origens das,
 prefixo compatível,
 roteadores,
 Datagramas,
 abordagem de pilha dupla,
 campos para,
 fragmentação,
 soma de verificação do cabeçalho,
 limite de salto,
 Protocolo da Internet (IP) e,
 IPsec,
 Formato IPv4,
 Formato IPv6,
 comprimento,
 protocolo da camada de rede e,
 inspeção de segurança,
 Banco de Dados de Política de Segurança (SPD),
 campo de tempo de vida,
 transição do IPv4 para o IPv6,
 abordagem implantação de túnel,
 bits de tipo de serviço (TOS),
 Algoritmos descentralizados,
 descompressão,
 Algoritmo de decodificação,
 roteador padrão,
 Agência de Projetos de Pesquisa Avançada de Defesa (Defense Advanced Research Projects Agency DARPA),
 Atraso,

serviço de melhor esforço,
 propagação de canal,
 comparação de transmissão e propagação,
 redes de distribuição de conteúdo (CDN),
 dimensionando redes,
 sistema final,
 fim a fim,
 variação,
 recuperação da perda,
 empacotamento de rede,
 rede multimídia,
 nodal,
 perda de pacote, redes comutadas por pacotes e,
 estratégias de reprodução,
 processamento,
 propagação,
 enfileiramento,
 programa Traceroute,
 transmissão,
 chamada deliver_data (),
 Zona desmilitarizada (DMZ),
 Demultiplexação,
 não orientada para conexão,
 orientada para conexão,
 uso da entrega de dados,
 números de porta,
 aplicações TCP,
 aplicações UDP,
 ataques DoS (Denial-of-service),
 Campo de número de porta da fonte,
 Roteador de destino,
 Redes dial-up,
 Serviço diferenciado (DiffServ),
 função de medição,
 aplicações de serviço multimídia,
 comportamento por salto (PHB),
 perfil de tráfego,
 Troca de Chaves Diffie-Hellman (Diffie Hellman Key Exchange),
 Assinaturas digitais,
 linha digital do assinante (DSL),
 algoritmo de Dijkstra,
 Dimensionando redes,
 Fibra direta,
 Roteamento direto,
 Mensagem de descoberta,
 Algoritmos de vetor de distâncias (DV),
 equação de Bellman-Ford para,
 roteamento descentralizado,

relação do caminho de menor custo,
 custo do enlace,
 Algoritmos de estado de enlace (LS) comparados a,
 reversão envenenada,
 laço de roteamento,
 tabelas de roteamento para,
 Protocolo de roteamento multicast por vetor de distância (distance vector multicast routing protocol DVMRP),
 Aplicações distribuídas,
 ataque DoS distribuído (DDoS),
 Distributed Hash Table (DHT),
 Espaçamento Interquadros Distribuído (Distributed Inter-Frame Space DIFS),
 Tempo de distribuição (D),
 Sistema de nomes de domínio (DNS),
 caching,
 problemas de um arranjo centralizado,
 banco de dados hierárquico (servidores raiz),
 apelidos de hospedeiro,
 nome de hospedeiro,
 tradução de nome de hospedeiro para endereço IP,
 endereços IP,
 distribuição de carga,
 apelido do servidor de correio,
 mensagens,
 aplicações da rede,
 consultas,
 registros de recursos (RR),
 Notação decimal separada por pontos,
 Abordagem de pilha dupla, IP,
 Campo de duração,
 Algoritmos dinâmicos,
 Protocolo de Configuração Dinâmica de Hospedeiros (DHCP),
 mensagem ACK,
 mensagem de descoberta,
 endereços IP da camada de rede,
 mensagem de oferta,
 protocolo plug and play,
 mensagem de requisição,
 Eficiência, Ethernet,
 Aplicações elásticas,
 Correio eletrônico (e-mail),
 protocolos de acesso,
 desenvolvimento do,
 Protocolo IMAP (Internet Mail Access Protocol),

- servidores de correio,
 formatos de mensagem,
 Post Office Protocol versão 3
 (POP3),
 Pretty Good Privacy (PGP),
 segurança,
 Protocolo Simples de Transferência
 de Correio (Simple Mail Transfer
 Protocol SMTP),
 agentes de usuário,
 baseado na Web,
 Codificação, veja também Criptografia,
 Autenticação do ponto de chegada,
 ataques homem no meio,
 defesa com um nonce,
 ataque de reprodução,
 autenticação da chave pública,
 Sistemas finais,
 Interface de Programação de
 Aplicação (API),
 clientes,
 enlaces de comunicação (mídia),
 atraso,
 dispositivos,
 aplicações distribuídas,
 hospedeiros como,
 Provedores de Serviço da Internet
 (ISP) e,
 pacotes,
 caminho (rota),
 roteadores,
 servidores,
 comutadores,
 taxa de transmissão,
 Conexões fim a fim,
 serviços de melhor esforço,
 controle de congestionamento,
 Política de descarte do final da fila,
 garantia definitiva,
 serviço multimídia e,
 garantia branda,
 projeto de sistema UDP,
 Taxa de Dados Otimizada para Evolução
 Global (EDGE),
 Erros,
 de bits,
 soma de verificação,
 Verificação de redundância cíclica
 (CRC),
 bits de detecção e de correção
 (EDC),
 correção de erros de repasse
 (FEC),
 serviços da camada de enlace,
- verificações de paridade,
 transferência de dados confiável
 (rtde),
 serviços da camada de transporte,
 bits não detectados,
 Ethernet,
 funções da rede de acesso,
 acesso múltiplo com detecção
 de portadora com detecção de colisão
 (CSMA/CD),
 desenvolvimento da,
 eficiência,
 estrutura do quadro,
 hubs,
 aplicações da camada de enlace,
 tecnologias de rede local (LAN)
- e,
- Encapsulamento,
 repetidor,
 comutador,
 tecnologias,
 conexão não confiável da,
 Comunicação baseada em eventos,
 bit de indicação explícita de conges-
 tionamento à frente (explicit forward
 congestion indication EFCI),
 Campo ER (explicit rate taxa explí-
 ita),
 Média móvel exponencial (EWMA)
 Protocolo de Autenticação Extensível
 (Extensible Authentication Protocol
 EAP),
 Recuperação rápida, TCP,
 Retransmissão rápida,
 FDDI (Fiber-distributed data inter-
 face),
 Fibras óticas,
 Redes FTTH (fiber-to-the-home),
 Distribuição de arquivo, P2P,
 Compartilhamento de arquivo, P2P,
 Protocolo de Transferência de Arquivo
 (FTP),
- aplicações de áudio,
 comandos,
 conexão de controle,
 conexão de dados,
 cenários de aplicação multimídia,
 protocolo de aplicação de rede,
 respostas,
 estado das informações do
 usuário,
 atraso,
- Filtragem,
- Máquina de estado finito (finite-state
 machine FSM),
 Firewalls,
 gateways de aplicação,
 filtragem de pacote,
 filtros de estado,
 servidor proxy TOR,
 Roteador do primeiro salto,
 Ordem primeiro a entrar/primeiro a sair
 (first-in-first-out — FIFO),
 Campo de flag, TCP,
 Inundação,
 Controle de fluxo,
 Agente externo,
 Rede externa (visitada),
 correção de erros de repasse (forward
 error correction FEC),
 Encaminhamento,
 endereçamento e,
 datagramas,
 fragmentação,
 Protocolo de Mensagens de
 Controle da Internet (ICMP),
 Protocolo da Internet (IP),
 IPv4,
 IPv6,
 comutadores de camada de
 enlace,
 tradução de endereços na rede
 (NAT),
 funções da camada de rede
 repasse pelo caminho inverso
 (reverse path forwarding — RPF)
 roteamento e,
 segurança (IPsec),
 tabelas,
 Universal Plug and Play (UPnP),
 Campo de controle do quadro,
 Quadros,
 varredura ativa e passiva,
 sinalização,
 byte stuffing,
 colisão,
 campos sem fio 802.11
 estrutura da Ethernet,
 camada de enlace,
 acesso múltiplo e,
 pacote (dados),
 camada física,
 Protocolo Ponto a Ponto (PPP),
 tempo,
 intervalos de tempo,
 conexões não confiáveis dos,
 LAN sem fio (WiFi),

multiplexação por divisão de frequência (frequency-division multiplexing FDM)

frequency-hopping spread spectrum (FHSS),

Central de Comutação para Portal de Serviços Móveis (Gateway Mobile Services Switching Center GMSC), Roteadores de borda,

Serviço de Pacote de Rádio Geral (General Packet Radio Service - GPRS),

Satélite geoestacionário,

Algoritmos globais,

Global System for Mobile Communications (Sistema Global para Comunicações Móveis GSM),

padrões de sistema celular, transferências (handoff) em, IP móvel comparado ao Grafo, roteamento,

Árvore compartilhada pelo grupo, Meio guiado,

Padrões H.323,

Transferência (Handoff), Apresentação (Handshake), abordagem de sinalização de estado hard,

funções de hash, criptográficas, Hash tables,

- circulares,
- distribuídas,
- peer churn,

Bloqueio HOL (head-of-the-line), Campos de cabeçalho,

Problema do terminal oculto, Roteamento hierárquico, . Veja também Sistemas autônomos (AS), Registro junto ao agente nativo, registro nativo de localização (home location register HLR), central de comutação móvel (MSC), Rede doméstica,

Límite de saltos,

Nomes de hospedeiros,

- apelido,
- canônico,
- caching DNS,
- banco de dados do servidor DNS e,
- tradução de endereço IP de, registros de recursos (RR),

Hospedeiros. Veja também Nós; Roteadores,

endereçamento, apelido, sistemas finais como, sem fio, Hubs, Ethernet, Rede híbrida fibra-coaxial (HFC), HTTP (HyperText Transfer Protocol Protocolo de Transferência de Hipertexto), protocolo da camada de aplicação, cookies, formato de mensagem, (continuação de HTTP (HyperText Transfer Protocol Protocolo de Transferência de Hipertexto), conexões não-persistentes, conexões persistentes, mensagem de requisição, mensagem de resposta, tempo de viagem de ida e volta (round-trip time — RTT), Protocolo Simples de Transferência de Correio (SMTP) comparado ao, Protocolo de Controle de Transmissão (TCP) e, páginas Web, World Wide Web (WWW), LAN sem fio IEEE 802.11, veja Wi-Fi, Política de importação, Informações na banda, Roteamento indireto, Portas de entrada, roteamento, Cadeia de entrada, Vazão instantânea, Serviços integrados (Intserv), roteamento inter-sistema autônomo, protocolo de algoritmos, Protocolo de Roteador de Borda (Border Gateway Protocol — BGP), intra-sistema autônomo comparado a, Interface (fronteiras), processadores de mensagens de interface (interface message processors — IMP), protocolos de roteadores internos (IGP Interior Gateway Protocols), veja protocolo de roteamento intra-sistema autônomo, Intercalação, Internet, redes de acesso, serviços de transporte da camada de aplicação providos pela, ataques, serviços de fluxo de áudio/vídeo, redes de backbone, redes de computadores e, atraso, desenvolvimento da, aplicações distribuídas, correio eletrônico (e-mail), sistemas finais (hospedeiros), padrões IETF ((Internet Engineering Task Force Força de Trabalho de Engenharia de Internet), suporte multimídia, estrutura de gerenciamento de rede, protocolo de rede, perda de pacote, camadas de protocolo, pedido de comentários (request for comments RFC), Camada de Sockets Seguros (SSL), vazão, serviços TCP (Transmission Control Protocol), Protocolo de Mensagens de Controle da Internet (ICMP), padrões IETF ((Internet Engineering Task Force Força de Trabalho de Engenharia de Internet), Protocolo de Gerenciamento de Grupo da Internet (Internet Group Management Protocol — IGMP), Troca de Chaves da Internet (Internet Key Exchange — IKE), IMAP (Internet Mail Access Protocol), Protocolo IP,. Veja também Protocolo de Internet Móvel (IP), endereçamento, serviço de melhor esforço do, Roteamento Interdomínio sem Classes (Classless Interdomain Routing CIDR), componentes, formatos de datagrama, abordagem de pilha dupla para transições de versão, Protocolo de Configuração Dinâmica de Hospedeiros (Dynamic Host Configuration Protocol DHCP), repasse, fragmentação, Protocolo de Mensagens de Controle da Internet (ICMP),

- IPv4,
IPv6,
aplicações multimídia,
Multiprotocol Label Switching (MPLS),
tradução de endereços na rede (NAT),
serviços da camada de rede, modelo de serviço, serviços da camada de transporte, abordagem implantação de túnel para transições de versão, Universal Plug and Play (UPnP), Segurança no Protocolo da Internet (IPsec), protocolo do Cabeçalho de Autenticação (AH), acordo criptográfico, integridade dos dados, datagramas, criptografia, Troca de Chaves da Internet (Internet Key Exchange — IKE), serviços da camada de rede, associação de segurança (SA), redes virtuais privadas (VPN), Provedores de Serviço da Internet (ISP), redes de backbone, sistemas finais e, pares, pontos de presença (points of presence POP), Telefonia por Internet, Entrevistas, Bellovin, Samuel M., Case Jeff, Cerf, Vinton G., Cohen, Bram, Floyd, Sally, Kleinrock, Leonard, Lam, Simon S., Perkins, Charlie, Schulzrinne, Henning, Roteamento intra-sistema autônomo, protocolo de algoritmo, inter-sistema autônomo comparado a, Open Shortest Path First (OSPF), RIP (Protocolo de Informação de Roteamento - Routing Information Protocol), Sistemas de Detecção de Invasão (IDS), Sistemas de Proteção à Invasão (IPS),
- IP spoofing,
IPTV,
IPv4, endereçamento, formato de datagrama, fragmentação, transição para IPv6,
IPv6, endereço anycast, formato de datagrama, abordagem pilha dupla para, rotulação de fluxo e prioridade, transição do IPv4 para, abordagem implantação de túnel para, consultas iterativas, Aplicações Java, programação de socket, Protocolo de Controle de Transmissão (TCP), Protocolo de Datagrama do Usuário (User Datagram Protocol UDP), Variação, rede multimídia e, pacote, estratégias de atraso de reprodução, remoção, números de sequência para, e áudio e vídeo armazenados de fluxo contínuo, marca de tempo para, Chaves, Política do tipo leaky bucket (balde furado), Caminho de menor custo (mais curto) Placa de linha, Velocidade de linha, roteadores, interface de enlace, Camada de enlace, reconhecimentos, Wi-Fi, Protocolo de Resolução de Endereços (Address Resolution Protocol ARP), endereçamento, canais de broadcast, Ethernet, quadros, implementação de, rede local (LAN), controle de acesso ao meio (MAC),
- acesso múltiplo, Multiprotocol Label Switching (MPLS), adaptador de rede, nós, conexões ponto a ponto, Protocolo Ponto a Ponto (PPP), protocolo, serviços providos por, comutadores, solicitação de página Web, Algoritmos de estado de enlace (LS), broadcast, de Dijkstra, algoritmos de Vetor de Distância (VD) comparados a, roteamento global, Pesos de enlace, OSPF, Algoritmo sensível/insensível à carga, Redes locais (LAN), Bluetooth (IEEE 802.15.1), celular versus tecnologia sem fio, evolução da Ethernet a partir de, interface de dados distribuída de fibra (FDDI), comutadores da camada de enlace, protocolos de acesso múltiplo, segurança, anéis de passagem de permissão (token rings), virtual (VLAN), Wi-Fi (IEEE 802.16), Privacidade Equivalente Cabeada (WEP), sem fio, Servidor DNS local, Comunicação lógica, Recuperação da perda, correção de erros de repasse (FEC), intercalação, pacotes, Reparação de correntes de áudio realizada no receptor, Satélites de baixa órbita (LEO), Servidores de correio, Malware, Ataques homem no meio, Base de Informações de Gerenciamento (Management Information Base MIB), tamanho máximo do segmento (maximum segment size MSS), unidade máxima de transmissão máxima (maximum transmission unit MTU),

Meio,
cabô coaxial,
fibra ótica,
guiado,
atraso de empacotamento,
transdutor,
canais de rádio por satélite,
canais de rádio terrestres,
par de fios de cobre trançado,
não guiado,
Integridade de mensagem,
chave de autenticação,
autoridade certificadora (certification authority — CA),
segurança na rede de computadores e,
funções de hash criptográficas e,
assinaturas digitais,
o código de autenticação da mensagem (MAC),
certificação da chave pública,
Mensagens,
camada de aplicação,
sistema de nomes de domínio (DNS),
Protocolo de Configuração Dinâmica de Hospedeiros (Dynamic Host Configuration Protocol — DHCP),
formatos de e-mail,
formato HTTP (HyperText Transfer Protocol — Protocolo de Transferência de Hipertexto),
Protocolo de Mensagens de Controle da Internet (ICMP),
aplicações da camada derrede,
pacotes,
requisição,
resposta,
resposta RIP (anúncios) [Protocolo de Informação de Roteamento - Routing Information Protocol],
Protocolo de Inicialização de Sessão (Session Initiation Protocol — SIP),
sinalização,
segmentos da camada de transporte e,
Metarquivos,
Função de medição, Diffserv,
redes de área metropolitana (metropolitan area networks — MANs),
Projeto Minitel,
redes móveis ad hoc (MANET),
Protocolo da Internet Móvel (IP),
anúncio de agente,

campo de endereço aos cuidados de (COA),
padrões celulares GSM comparados a,
registro junto ao agente nativo,
roteamento indireto,
número roaming da estação móvel (mobile station roaming number — MSRN),
Central de comutação de unidade móvel (MSC),
Gerenciamento de mobilidade,
endereçamento,
aos cuidados de (COA),
correspondente,
dimensões de,
roteamento direto,
encapsulamento,
rede externa (visitada),
agente externo,
transferências (handoffs),
rede doméstica,
roteamento indireto,
nós móveis,
roteamento,
Modems,
Cifra monoalfabética,
Rede stub com múltiplas interconexões
Redes de sobreposição multicast,
Roteamento multicast,
endereço indireto,
algoritmos,
Protocolo de roteamento multicast por vetor de distância (distance vector multicast routing protocol — DVMRP),
grupo,
aplicações da Internet,
Protocolo de Gerenciamento de Grupo da Internet (Internet Group Management Protocol — IGMP),
protocolo de roteamento PIM (protocol independent multicast — multicast independente de protocolo),
repasse pelo caminho inverso (reverse path forwarding — RPF),
protocolo Multicast de Fonte Específica (Source-Specific Multicast — SSM),
Rede multimídia,
áudio,
serviço de melhor esforço de compressão,
redes de distribuição de conteúdo (CDN),

serviço diferenciado (Diffserv),
aplicações FTP (Protocolo de Transferência de Arquivo),
serviços integrados (Intserv),
suporte da Internet para variação,
redes de sobreposição multicast, classes múltiplas de serviço para, perda de pacote, (continuação de Rede multimídia)
mecanismos de regulação, garantias de qualidade de serviço (QoS),
aplicações interativas em tempo real,
Protocolo de Recurso ReSerVation (RSVP),
mecanismos de escalonamento (enfileiramento),
fluxo,
vídeo,
Propagação multivias,
Acesso múltiplo,
protocolos ALOHA,
canais de broadcast,
acesso múltiplo com detecção de portadora (CSMA),
acesso à Internet celular,
protocolos de divisão de canal,
atraso de propagação de canal,
Acesso Múltiplo por Divisão de Código (Code Division Multiple Access — CDMA),
colisões,
Ethernet,
interface de dados distribuída de fibra (FDDI),
quadros e,
multiplexação por divisão de frequência (frequency-division multiplexing — FDM),
camada de enlace,
rede local (LAN) e,
protocolos de acesso aleatório,
protocolos de revezamento,
multiplexação por divisão de tempo (TDM),
Wi-Fi,
redes sem fio,
Multiplexação,
redes comutadas de circuito,
não orientada a conexão,
orientada a conexão,
uso de reunião e passagem de dados,

- multiplexação por divisão de frequência (frequency-division multiplexing FDM),
 comutação de pacote,
 número de porta,
 sockets,
 estatística,
 aplicações TCP,
 divisão de tempo (TDM),
 camada de transporte,
 aplicações UDP,
 Multiprotocol Label Switching (MPLS),
 abordagem unicast de N caminhos
 Reconhecimentos negativos (NAK),
 Adaptador de rede,
 tradução de endereços na rede (NAT),
 Aplicações de rede,
 protocolo da camada de aplicação comparado a, sistema de nomes de domínio (DNS),
 correio eletrônico (e-mail),
 Protocolo de Transferência de Arquivo (FTP)
 ponto a ponto (P2P),
 World Wide Web (WWW),
 Controle de congestionamento assistido pela rede,
 Controlador de Interface de Rede (NIC),
 Camada de rede,
 endereçamento,
 serviço ATM ABR (available bit rate),
 rede de datagrama,
 datagrama,
 repasse,
 Protocolo de Mensagens de Controle da Internet (ICMP),
 Protocolo da Internet (IP),
 Segurança no Protocolo da Internet (IPsec),
 roteamento multicast,
 protocolo,
 arquitetura do roteador,
 roteamento,
 algoritmos de roteamento,
 modelos de serviço,
 camadas de transporte e,
 redes de circuito virtual (CV),
 (continuação de Camada de rede)
 redes virtuais privadas (VPNs),
 Gerenciamento de rede,
 Notação de sintaxe abstrata 1 (Abstract Syntax Notation One ASN.1),
 infraestrutura,
 quadro da Internet para,
 Base de Informações de Gerenciamento (Management Information Base MIB),
 central de operações (network operations center NOC),
 cenários para,
 Protocolo de Monitoramento do Gateway Simples (Simple Gateway Monitoring Protocol - SGMP),
 Protocolo de Gerenciamento da Rede Simples (Simple Network Management Protocol - SNMP),
 Estrutura do Gerenciamento da Informação (SMI),
 Central de operações (network operations center NOC),
 Redes, veja Redes de computadores;
 redes sem fionmap, uso de varredura de porta,
 Bit NI (no increase não aumentar),
 Nós,
 endereçamento,
 correspondente,
 datagramas enviados para,
 atraso,
 detecção de erros,
 controle de fluxo,
 agente externo,
 grafos,
 transmissão half-duplex e full-duplex,
 agente nativo,
 camada de enlace,
 móvel,
 roteamento para dispositivos móveis,
 Conexões não persistentes,
 Defesa com um nonce,
 programa nslookup,
 arquivo objeto,
 Open Shortest Path First (OSPF),
 roteadores de borda de área,
 área de backbone,
 roteamento intra-sistema autônomo,
 pesos de enlace,
 avanços do roteamento,
 Segurança operacional,
 Campo de opções, TCP,
 Modelo de referência OSI,
 Informações fora da banda,
 Buffer de saída (fila),
 Portas de saída, roteadores,
 Cadeia de saída,
 Filtragem de pacote,
 Escalonador de pacote,
 Analisaadores de pacote,
 Comutação de pacote,
 comutação de circuito comparada à,
 movimentação de pacote,
 atraso,
 desenvolvimento de,
 sistemas finais e,
 base de encaminhamento,
 processadores de mensagens de interface (interface message processors — IMP),
 comutadores da camada de enlace,
 buffer de saída (fila),
 perda de pacote,
 caminho (rota),
 atraso de fila,
 roteadores,
 multiplexação estatística,
 transmissão armazena e envia,
 vazão,
 Pacotes,
 serviços de melhor esforço, congestionamento,
 duplicado,
 variação,
 perda,
 rede multimídia,
 Real-Time Transport Protocol (RTP),
 transferência de dados confiável (rtt),
 retransmissão,
 nímeros de sequência para,
 Verificações de paridade,
 Redes Ópticas Passivas (PON),
 Caminho (rota),
 Atenuação de percurso,
 Campo de carga útil,
 Peer churn,
 Pares,
 Protocolo de Roteador de Borda (BGP),
 Provedores de Serviço da Internet (ISP),
 unchoked (não sufocado),
 Ponto a Ponto (P2P),
 aplicações da camada de aplicação,
 arquitetura,
 protocolo BitTorrent,

processo cliente,
Distributed Hash Table (DHT) para, tempo de distribuição (D), distribuição de arquivo, compartilhamento de arquivo, telefonia por Internet, escalabilidade de, processo servidor, aplicação Skype, pares unchoked (não sufocados), Comportamentos por salto (PHB), Diffserv, Conexões persistentes, Camada física, Piconet, Bluetooth, Reconhecimento pegou uma carona, Transferência confiável de dados com paralelismo, reconhecimento cumulativo, protocolo Go-Back-N (GBN), repetição seletiva (SR), utilização do remetente, protocolo de janela deslizante (sliding-window protocol), protocolo pare e espere comparado à, eventos de esgotamento de temporização, Texto aberto, Ataque de reprodução, Atraso de reprodução, Protocolo plug and play, Conexões ponto a ponto (enlaces), Protocolo Ponto a Ponto (PPP), Pontos de presença (POP), Técnica da reversão envenenada, Mecanismos de regulação, Protocolo de polling, Cifra polialfabética, Números de porta, destino, inversão de, varredura, fonte, servidores Web, POP3 (Post Office Protocol versão 3), Gerenciamento de potência, Wi-Fi, Prefixos, Pretty Good Privacy (PGP), Enfileiramento prioritário, Domínio com endereços privados, NAT, Chave privada, Atraso de processamento,

Atraso de propagação, protocolo de roteamento PIM (protocol independent multicast — multicast independente de protocolo), Camadas do protocolo, aplicação, redes de computadores, datagramas de, quadros, arquitetura em camada, enlace, mensagens, modelo de referência OSI, física, segmentos de, modelo de serviço, pilha, transporte, Protocolos, analogia humana para, padrões IETF (Internet Engineering Task Force Força de Trabalho de Engenharia de Internet) para, rede, fora da banda, roteamento (fim a fim), janela deslizante, Provedor, Servidor proxy, SIP, Poda, Chave pública, criptografia de chave pública, Autoridade Certificadora (CA), Troca de Chaves Diffie-Hellman (Diffie Hellman Key Exchange), uso da autenticação do ponto de chegada, chaves (pública e privada), integridade de mensagem e, algoritmo RSA, rede pública terrestre móvel nativa (PLMN), Qualidade de serviço (QoS), admissão de chamada, configuração de chamada, abordagem de sinalização de estado hard, serviços integrados (Intserv), serviços da Internet, garantias de multimídia, O Protocolo de Recurso ReSerVation (RSVP), reserva de recursos,

abordagem de sinalização de estado soft, Consultas, DNS, Fila (buffer de saída), Enfileiramento, gerenciamento ativo (AQM), atraso, primeiro a entrar/primeiro a sair (first-in-first-out — FIFO), bloqueio HOL (head-of-the-line), comprimento, escalonamento multimídia, (continuação de Enfileiramento) roteadores da camada de rede, perda de pacote e, escalonador de pacote para, prioridade, algoritmo de detecção aleatória rápida (random early detection RED), varredura cíclica, mecanismos de escalonamento, taxa do elemento de comutação, intensidade do tráfego, ponderada justa (WFQ), Protocolos de acesso aleatório, algoritmo de detecção aleatória rápida (random early detection RED), Adaptação da taxa, Wi-Fi, chamada rdt_rcv(), chamada rdt_send(), multimídia interativa em tempo real, aplicações de áudio e vídeo, padrões H.23, telefonia por Internet, Real-Time Transport Protocol (RTP), Protocolo de Inicialização de Sessão (Session Initiation Protocol SIP), Protocolo de fluxo contínuo em tempo real (RTSP), Real-Time Transport Protocol (RTP), Protocolo de Controle (RTCP), cabeçalho, pacotes, escalabilidade da largura de banda do RTCP, sessão, aplicações de software para, janela de recepção, recuperação de fluxos de áudio baseada no receptor, Formato de registro,

- Consultas recursivas,
Registros, DNS,
Entidade registradora, SIP,
Transferência de dados confiável (rdt),
 reconhecimentos (ACK),
 protocolo bit alternante,
 serviços da camada de aplicação,
 protocolos ARQ (solicitação automática de repetição - Automatic Repeat reQuest),
 bidirecionais,
 erros de bit e,
 canal com erros de bit (protocolo rdt2.0),
 soma de verificação (detecção de erro),
 pacotes de dados e,
 chamada deliver_data(),
 Máquina de estado finito (finite-state machine FSM),
 protocolo Go-Back-N (GBN),
 canal com perda e com erros de bit (protocolo rdt3.0), reconhecimentos negativos (NAK),
 paralelismo,
 chamada rdt_rcv(),
 chamada rdt_send(),
 canal confiável (protocolo rdt1.0),
 repetição seletiva (SR) para,
 números de sequência,
 modelo de serviço para,
 protocolos pare e espere,
 serviços TCP (Protocolo de Controle de Transmissão),
 aplicações da camada de transporte,
 unidirecional,
Ponto de encontro,
Repetidor, Ethernet,
Mensagem de requisição, HTTP,
Solicitação de envio (Request to send RTS),
Pedido de comentários (request for comments RFC),
células RM (resource-management cells),
Registros de Recurso (RR),
Protocolo de Recurso ReSerVation (RSVP),
Mensagem de resposta, HTTP,
Repasse pelo caminho inverso (reverse path forwarding — RPF),
Servidor DNS raiz,
- Enfileiramento por varredura cíclica,
Tempo de viagem de ida e volta (round-trip time — RTT),
Roteadores,
 arquitetura,
 sistemas autônomos (AS),
 rede de datagrama,
 padrão,
 destino,
 primeiro salto,
 base de encaminhamento,
 borda,
 portas de entrada,
 velocidade de linha,
 comutadores da camada de enlace comparados a,
 arquitetura da camada de rede,
 portas de saída,
 redes comutadas por pacotes e,
 comutadores de pacotes como,
 enfileiramento,
 funções de roteamento,
 processador de roteamento,
 fonte,
 elemento de comutação,
 rede de circuito virtual (CV),
Roteamento,
 função de algoritmo, sistemas autônomos (AS),
 Protocolo de Roteador de Borda (BGP),
 broadcast,
 chamadas por sistema celular,
 direto,
 encapsulamento,
 repasse e,
 grafos para,
 hierárquico,
 batata quente,
 indireto,
 inter-sistema autônomo,
 intra-sistema autônomo,
 nós móveis,
 número roaming da estação móvel (mobile station roaming number MSRN),
 protocolo de localização de usuário móvel,
 multicast,
 função da camada de rede,
 Open Shortest Path First (OSPF),
 protocolo,
 mensagem de resposta (anúncios),
repasse pelo caminho inverso (reverse path forwarding — RPF),
RIP (Protocolo de Informação de Roteamento - Routing Information Protocol),
tabelas,
problema triangular,
sistemas sem fio,
Algoritmos de roteamento,
sistemas autônomos (AS),
broadcast,
circuito comutado,
descentralizado,
de Dijkstra,
vetor de distância (VD),
dinâmicos,
valores da tabela de repasse, globais,
roteamento hierárquico e,
protocolo de roteamento inter-sistema autônomo,
protocolo de roteamento intra-sistema autônomo,
estado de enlace (LS),
sensível/insensível à carga,
multicast,
enfileiramento,
Detecção Aleatória Rápida (random early detection RED),
 função de roteamento,
 laço de roteamento,
 estáticos,
RIP (Protocolo de Informação de Roteamento - Routing Information Protocol),
algoritmo RSA,
Canais de rádio por satélite,
Escalabilidade, P2P,
Mecanismos de escalonamento, veja Enfileiramento,
Camada de Sockets Seguros (SSL),
 encerramento de conexão,
 transferência de dados,
 apresentação (handshake),
 derivação de chave,
 formato de registro,
 segurança TCP,
Segurança,
 camada de aplicação,
 ataques,
 redes de computadores,
 funções de hash criptográficas,
 criptografia,
 inspeção de datagramas,

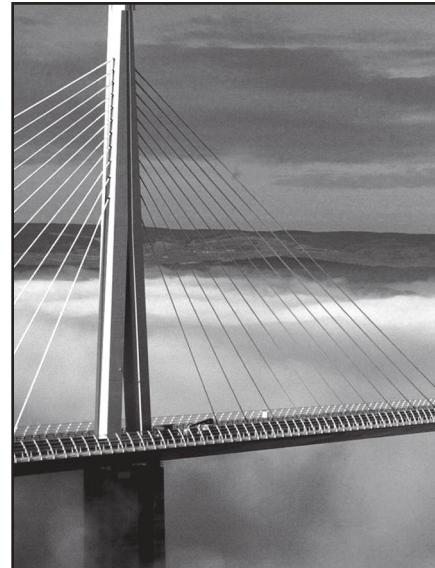
ataques DoS (Denial-of-Service),
correio eletrônico (e-mail),
criptografia,
autenticação do ponto de chegada,
firewalls,
Protocolo da Internet (IPsec),
sistemas de detecção de invasão (IDS),
IP spoofing,
malware,
ataques homem no meio,
integridade de mensagem,
camada de rede,
operacional,
analisadores de pacote,
ataque de reprodução,
Pretty Good Privacy (PGP),
criptografia de chave pública,
propriedades da comunicação segura,
Camada de Sockets Seguros (SSL),
Protocolo de Controle de Transporte (TCP),
redes virtuais privadas (VPN),
Privacidade Equivalente Cabeada (WEP),
LAN sem fio,
Associação de Segurança (SA),
Banco de Dados de Associação de Segurança (SAD),
Índice de Parâmetro de Segurança (SPI),
Banco de Dados de Política de Segurança (SPD),
Segmentos,
soma de verificação,
transporte não orientado a conexão e,
campos de cabeçalho,
tamanho máximo (MMS),
unidade de transmissão máxima (MTU),
uso de conversão de mensagem,
campos de números de porta para,
números de sequência,
SYN,
estrutura do TCP,
camada de transporte,
estrutura do UDP,
Buffer de envio,
Utilização do remetente,
Números de sequência,

protocolos ARQ,
inundação controlada,
campo,
protocolo GBN,
uso da eliminação da variação,
repetição seletiva (SR),
segmentos TCP,
Server farm,
Programa servidor,
Servidores,
Acordos de Nível de Serviços (Service Level Agreements SLAs),
Identificador de Conjunto de Serviços (Service Set Identifier SSID),
Servidor de Nó de Suporte GPRS (SGSN),
Protocolo de Inicialização de Sessão (Session Initiation Protocol SIP),
endereços,
estabelecimento de chamada,
mensagens,
servidor proxy,
registradora,
Chave de sessão,
Espaçamento Curto Interquadros (Short Inter-Frame Spacing SIFS),
razão sinal-ruído (SNR),
Mensagens de sinalização,
Sistema baseado em assinatura,
Protocolo de Monitoramento do Gateway Simples (Simple Gateway Monitoring Protocol - SGMP),
Protocolo Simples de Transferência de Correio (SMTP),
protocolo da camada de aplicação,
correio eletrônico (e-mail),
HTTP (HyperText Transfer Protocol Protocol de Transferência de Hipertexto) comparado ao,
acesso ao correio,
Protocolo de Gerenciamento da Rede Simples (Simple Network Management Protocol - SNMP),
aplicação P2P Skype,
Protocolo de janela deslizante (sliding-window protocol),
Protocolo slotted ALOHA,
Partida lenta, TCP,
Segmento SYNACK,
Programação de sockets,
Interface de Programação de Aplicação (API),
aplicações cliente-servidor,

interface entre o processo e a rede de computadores,
aplicações Java,
fluxos,
Protocolo de Controle de Transmissão (TCP),
Protocolo de Datagrama do Usuário (User Datagram Protocol UDP),
Sockets,
Abordagem de sinalização de estado soft,
Árvore baseada na fonte,
Campo número de porta de destino,
Roteador de destino,
Protocolo Multicast de Fonte Específica (Source-Specific Multicast SSM),
Spanning trees,
Pilha, camadas de protocolo,
Entrada/saída padrão,
Inanição,
Estado das informações do usuário,
Filtros de estado,
Protocolo sem estado,
Algoritmos estáticos,
Multiplexação estatística,
Protocolo pare e espere,
Transmissão armazena e envia,
Fluxo contínuo,
buffer cliente,
serviços da Internet,
eliminação da variação por,
áudio e vídeo em tempo real,
transdutor,
recuperação de perda de pacote,
protocolo de fluxo contínuo em tempo real (RTSP),
recuperação de áudio baseada no receptor,
servidor,
vídeo e áudio armazenados,
acesso ao servidor Web,
Cadeia, programação de socket e,
Estrutura das Informações de Gerenciamento (SMI),
Rede stub,
Sub-redes,
Comutadores,
desvantagens dos,
Ethernet,
filtragem,
repasse,
camada de enlace,
redes locais (LAN) e,
dispositivos plug-and-play,



Sobre os autores



Jim Kurose

Jim Kurose é um ilustre professor do Departamento de Ciências da Computação da University of Massachusetts, em Amherst.

Foi condecorado inúmeras vezes por suas atividades educacionais, incluindo o Outstanding Teacher Awards da National Technological University (oito vezes), University of Massachusetts e Northeast Association of Graduate Schools. Recebeu a IEEE Taylor Booth Education Medal e foi reconhecido por sua liderança no Massachusetts' Commonwealth Information Technology Initiative. Recebeu também o GE Fellowship, o IBM Faculty Development Award e o Lilly Teaching Fellowship.

Além disso, foi editor-chefe do IEEE Transactions on Communications e do IEEE/ACM Transactions on Networking. É membro ativo do comitê de programas do IEEE Infocom, da ACM SIGCOMM e da ACM SIGMETRICS há muitos anos e atuou como Technical Program Co-Chair para essas conferências. É membro do IEEE e da ACM. Seus estudos incluem protocolos e arquitetura de rede, medidas de rede, redes de sensores, comunicação multimídia e modelagem e avaliação de desempenho. É doutor em ciência da computação pela Columbia University.

Keith Ross

Keith Ross é chefe de Departamento e professor catedrático Leonard J. Shustek em ciências da computação na Polytechnic University da NYU (no Brooklyn). De 1995 a 1998, lecionou no Departamento de Engenharia de Sistemas da University of Pennsylvania. De 1998 a 2003, foi professor do Departamento de Comunicações em Multimídia no Institute Eurecom, na França. Também é o principal fundador e presidente da Wimba, que desenvolve tecnologias voice-over-IP para ensino a distância.

Seus interesses de pesquisa estão nas áreas de rede ponto a ponto, dimensão da Internet, fluxo de vídeo, cache Web, redes de distribuição de conteúdo, segurança na rede, voz sobre IP e modelagem estocástica. É membro do IEEE e editor associado do IEEE/ACM Transactions on Networking. Foi conselheiro da Comissão Federal do Comércio para o compartilhamento de arquivo P2P. Participa de comitês de programas, incluindo IEEE Infocom, ACM SIGCOMM, ACM CoNext, IPTPS, ACM Multimedia, ACM Internet Measurement Conference e ACM SIGMETRICS. É doutor em engenharia da computação, informação e controle pela University of Michigan.

