

Computer Networking

A Top-Down Approach



sixth edition

KUROSE | ROSS

KOMPUTER JARINGAN

EDISI KEENAM

Pendekatan Top-Down



James F. Kurose

Universitas Massachusetts, Amherst

Keith W. Ross

Institut Politeknik NYU

PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapura Taipei Tokyo

Wakil Presiden dan Direktur Editorial, ECS:

Marcia Horton

Pemimpin Redaksi: Michael Hirsch

Asisten Editorial: Emma Snider

Wakil Presiden Pemasaran: Patrice Jones

Manajer Pemasaran: Yez Alayan

Koordinator Pemasaran: Kathryn Ferranti

Wakil Presiden dan Direktur Produksi:

Vince O'Brien

Redaktur Pelaksana: Jeff Holcomb

Manajer Proyek Produksi Senior:

Marilyn Lloyd

Manajer Manufaktur: Nick Sklitsis

Spesialis Operasi: Lisa McDowell

Art Director, Sampul: Anthony Gemmellaro

Koordinator Seni: Janet Theurer/

Desain Theurer Briggs

Studio Seni: Patrice Rossi Calkin/

Ilustrasi dan Desain Rossi

Desainer Sampul: Liz Harasymcuk

Desainer Teks: Joyce Cosentino Wells

Gambar Sampul: ©Fancy/Alamy

Media Editor: Dan Sandin

Vendor Layanan Penuh: PreMediaGlobal

Manajer Proyek Senior: Andrea Stefanowicz

Pencetak/Pengikat: Edwards Brothers

Sampul Printer: Warna Lehigh-Phoenix

Buku ini disusun dalam Quark. Font dasarnya adalah Times. Font tampilan adalah Berkeley.

Hak Cipta © 2013, 2010, 2008, 2005, 2003 oleh Pearson Education, Inc., diterbitkan sebagai Addison-Wesley. Seluruh hak cipta. Diproduksi di Amerika Serikat. Publikasi ini dilindungi oleh Hak Cipta, dan izin harus diperoleh dari penerbit sebelum reproduksi yang dilarang, penyimpanan dalam sistem pengambilan, atau pengiriman dalam bentuk apa pun atau dengan cara apa pun, elektronik, mekanik, fotokopi, rekaman, atau sejenisnya. Untuk mendapatkan izin menggunakan materi dari karya ini, kirimkan permintaan tertulis ke Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, atau Anda dapat mengirim faks permintaan Anda ke 201-236 -3290.

Banyak sebutan oleh produsen dan penjual untuk membedakan produk mereka diklaim sebagai merek dagang. Jika sebutan tersebut muncul dalam buku ini, dan penerbit mengetahui adanya klaim merek dagang, sebutan tersebut telah dicetak dengan huruf besar awal atau huruf besar semua.

Library of Congress Cataloging-in-Publication Data Kurose,

James F.

Jaringan komputer: pendekatan top-down / James F. Kurose, Keith W. Ross.—edisi ke-6.

P. cm.

Termasuk referensi bibliografi dan indeks.

ISBN-13: 978-0-13-285620-1

ISBN-10: 0-13-285620-4

1. Internet. 2. Jaringan komputer. I. Ross, Keith W., 1956- II. Judul.

TK5105.875.I57K88 2012

004.6—dc23

2011048215

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN-10: 0-13-285620-4

ISBN-13: 978-0-13-285620-1

Tentang Penulis

Jim Kurose

Jim Kurose adalah Profesor Ilmu Komputer Universitas Terkemuka di University of Massachusetts, Amherst.

Dr. Kurose telah menerima sejumlah pengakuan atas pendidikannya kegiatan termasuk Penghargaan Guru Berprestasi dari Universitas Teknologi Nasional (delapan kali), Universitas Massachusetts, dan Asosiasi Sekolah Pascasarjana Timur Laut. Dia menerima Medali Pendidikan IEEE Taylor Booth dan diakui atas kepemimpinannya di Inisiatif Teknologi Informasi Persemakmuran Massachusetts. Dia telah menjadi penerima GE Fellowship, IBM Faculty Development Award, dan Lilly Teaching Fellowship.



Dr. Kurose adalah mantan Pemimpin Redaksi IEEE Transaksi aktif Komunikasi dari dan ~~IEEE~~ dalam Transaksi aktif Jaringan. Dia punya kepanitiaan program untuk dan ACM SIGMETRICS SIGCOMM, IEEE ACM Internet Konferensi Pengukuran , untuk sebuah beberapa tahun dan menjabat sebagai Technical Program Co-Chair untuk konferensi tersebut. Dia adalah Fellow dari IEEE dan ACM. Minat penelitiannya meliputi protokol dan arsitektur jaringan, pengukuran jaringan, jaringan sensor, komunikasi multimedia, dan pemodelan dan evaluasi kinerja. Dia memegang gelar PhD dalam Ilmu Komputer dari Universitas Columbia.

Keith Ross

Keith Ross adalah Profesor Ketua Leonard J. Shustek dan Kepala Departemen Ilmu Komputer di Institut Politeknik NYU. Sebelum bergabung dengan NYU-Poly pada tahun 2003, beliau adalah seorang profesor di University of Pennsylvania (13 tahun) dan seorang profesor di Eurecom Institute (5 tahun). Dia menerima BSEE dari Tufts University, MSEE dari Columbia University, dan Ph.D. dalam Teknik Komputer dan Kontrol dari The University of Michigan. Keith Ross juga merupakan pendiri dan CEO asli Wimba, yang mengembangkan aplikasi multimedia online untuk e-learning dan diakuisisi oleh Blackboard pada tahun 2010.

Minat penelitian Profesor Ross adalah keamanan dan privasi, jejaring sosial, jaringan peer-to-peer, pengukuran Internet, streaming video, jaringan distribusi konten, dan pemodelan stokastik. Dia adalah IEEE Fellow, penerima Infocom 2009 Best Paper Award, dan penerima Best Paper Awards 2011 dan 2008 untuk Multimedia Communications (diberikan oleh IEEE Communications Society). Dia telah bertugas di banyak dewan editorial jurnal dan komite program konferensi, termasuk IEEE/ACM Transaksi pada Jaringan , ACM SIGCOMM , ACM CoNext , dan ACM Internet Konferensi Pengukuran . Dia juga menjabat sebagai penasihat Komisi Perdagangan Federal untuk berbagi file P2P.



Halaman ini sengaja dikosongkan

Untuk Julie dan ketiga orang
tersayang kami—Chris, Charlie, dan Nina
JFK

TERIMA KASIH sebesar-besarnya kepada para profesor,
kolega, dan mahasiswa saya di seluruh dunia.

KWR

Halaman ini sengaja dikosongkan

Kata pengantar

Selamat datang di edisi keenam *Jaringan Komputer: Pendekatan Top-Down*. Sejak penerbitan edisi pertama 12 tahun lalu, buku kami telah diadopsi untuk digunakan di ratusan perguruan tinggi dan universitas, diterjemahkan ke dalam 14 bahasa, dan digunakan oleh lebih dari seratus ribu mahasiswa dan praktisi di seluruh dunia. Kami telah mendengar dari banyak pembaca ini dan terbebani oleh tanggapan positifnya.

Apa yang Baru di Edisi Keenam?

Menurut kami salah satu alasan penting untuk keberhasilan ini adalah karena buku kami terus menawarkan pendekatan baru dan tepat waktu untuk instruksi jaringan komputer. Kami telah membuat perubahan dalam edisi keenam ini, tetapi kami juga tidak mengubah apa yang kami yakini (dan para instruktur serta siswa yang telah menggunakan buku kami telah mengonfirmasi) sebagai aspek terpenting dari buku ini: pendekatan dari atas ke bawah , fokusnya pada Internet dan perlakuan modern terhadap jaringan komputer, perhatiannya pada prinsip dan praktik, serta gaya dan pendekatannya yang dapat diakses untuk mempelajari jaringan komputer. Meskipun demikian, edisi keenam telah direvisi dan diperbarui secara substansial:

- Situs Web Pendamping telah diperluas dan diperkaya secara signifikan untuk menyertakan VideoNotes dan latihan interaktif, seperti yang akan dibahas nanti di Kata Pengantar ini.
- Dalam Bab 1, perlakuan terhadap jaringan akses telah dimodernisasi, dan deskripsi ekosistem ISP Internet telah direvisi secara substansial, mengingat munculnya jaringan penyedia konten baru-baru ini, seperti jaringan Google. Presentasi dari packet switching dan circuit switching juga telah diatur ulang, memberikan orientasi yang lebih topikal daripada sejarah.
- Dalam Bab 2, Python telah menggantikan Java untuk presentasi ming program socket. Sementara masih secara eksplisit memaparkan ide-ide kunci di balik socket API, kode Python lebih mudah dipahami oleh programmer pemula. Selain itu, tidak seperti Java, Python menyediakan akses ke soket mentah, memungkinkan siswa untuk membangun variasi aplikasi jaringan yang lebih besar. Lab pemrograman soket berbasis Java telah diganti dengan lab Python yang sesuai, dan lab ICMP Ping berbasis Python baru telah ditambahkan. Seperti biasa, saat materi dihentikan dari buku, seperti materi pemrograman soket berbasis Java, materi tetap tersedia di situs Web Pendamping buku (lihat teks berikut).
- Di Bab 3, penyajian salah satu protokol transfer data yang andal telah disederhanakan dan sidebar baru pada pemisahan TCP, yang biasa digunakan untuk mengoptimalkan kinerja layanan cloud, telah ditambahkan. • Dalam Bab 4, bagian tentang arsitektur router telah diperbarui secara signifikan, mencerminkan perkembangan dan praktik terkini di lapangan. Beberapa sidebar integratif baru yang melibatkan DNS, BGP, dan OSPF disertakan.

- Bab 5 telah ditata ulang dan disederhanakan, memperhitungkan keberadaan Ethernet yang diaktifkan di mana-mana di jaringan area lokal dan akibatnya peningkatan penggunaan Ethernet dalam skenario point-to-point. Juga, bagian baru tentang jaringan pusat data telah ditambahkan.
- Bab 6 telah diperbarui untuk mencerminkan kemajuan terkini dalam jaringan nirkabel, khususnya jaringan data seluler dan layanan serta arsitektur 4G.
- Bab 7, yang berfokus pada jaringan multimedia, telah mengalami revisi besar-besaran. Bab ini sekarang mencakup diskusi mendalam tentang video streaming, termasuk streaming adaptif, dan diskusi CDN yang sepenuhnya baru dan modern. Bagian yang baru ditambahkan menjelaskan sistem streaming video Netflix, YouTube, dan Kankan. Materi yang telah dihapus untuk memberi jalan bagi topik baru ini masih tersedia di situs Web Pengiring. • Bab 8 sekarang berisi diskusi yang diperluas tentang autentifikasi titik akhir. • Materi baru yang signifikan yang melibatkan masalah akhir bab telah ditambahkan. Seperti semua edisi sebelumnya, soal-soal PR telah direvisi, ditambah, dan dihilangkan.

Hadirin

Buku teks ini untuk kursus pertama tentang jaringan komputer. Ini dapat digunakan di departemen ilmu komputer dan teknik elektro. Dalam hal bahasa pemrograman, buku ini hanya mengasumsikan bahwa siswa memiliki pengalaman dengan C, C++, Java, atau Python (itupun hanya di beberapa tempat). Meskipun buku ini lebih tepat dan analitis daripada banyak teks pengantar jaringan komputer lainnya, buku ini jarang menggunakan konsep matematika yang tidak diajarkan di sekolah menengah. Kami telah melakukan upaya yang disengaja untuk menghindari penggunaan konsep kalkulus lanjutan, probabilitas, atau proses stokastik (walaupun kami telah memasukkan beberapa masalah pekerjaan rumah untuk siswa dengan latar belakang lanjutan ini). Oleh karena itu, buku ini sesuai untuk program sarjana dan untuk program pascasarjana tahun pertama. Ini juga harus berguna bagi para praktisi di industri telekomunikasi.

Apa yang Unik tentang Buku Pelajaran Ini?

Subjek jaringan komputer sangat kompleks, melibatkan banyak konsep, protokol, dan teknologi yang dijalankan bersama dengan cara yang rumit. Untuk mengatasi ruang lingkup dan kompleksitas ini, banyak teks jaringan komputer sering diatur di sekitar "lapisan" arsitektur jaringan. Dengan organisasi berlapis, siswa dapat melihat melalui kompleksitas jaringan komputer—mereka belajar tentang konsep dan protokol yang berbeda di satu bagian arsitektur sambil melihat gambaran besar tentang bagaimana semua bagian cocok satu sama lain. Dari perspektif pedagogis, pengalaman pribadi kami adalah pendekatan berlapis itu

memang bekerja dengan baik. Namun demikian, kami telah menemukan bahwa pendekatan pengajaran tradisional — dari bawah ke atas; yaitu, dari lapisan fisik menuju lapisan aplikasi— bukanlah pendekatan terbaik untuk kursus modern tentang jaringan komputer.

Pendekatan Top-Down

Buku kami membuat terobosan baru 12 tahun yang lalu dengan memperlakukan jaringan secara top-down—yaitu, dengan memulai dari lapisan aplikasi dan bergerak ke bawah menuju lapisan fisik. Umpulan yang kami terima dari guru dan siswa telah mengkonfirmasi bahwa pendekatan top-down ini memiliki banyak keuntungan dan memang bekerja dengan baik secara pedagogis. Pertama, ini menekankan pada lapisan aplikasi ("area pertumbuhan tinggi" dalam jaringan). Memang, banyak revolusi baru-baru ini dalam jaringan komputer—termasuk Web, berbagi file peer-to-peer, dan streaming media—telah terjadi di lapisan aplikasi. Penekanan awal pada masalah lapisan aplikasi berbeda dari pendekatan yang diambil di sebagian besar teks lain, yang hanya memiliki sedikit materi pada aplikasi jaringan, persyaratannya, paradigma lapisan aplikasi (misalnya, client-server dan peer-to-peer), dan antarmuka pemrograman aplikasi. Kedua, pengalaman kami sebagai instruktur (dan dari banyak instruktur yang telah menggunakan teks ini) adalah bahwa aplikasi jaringan pengajaran di awal kursus adalah alat motivasi yang kuat. Siswa senang mempelajari tentang cara kerja aplikasi jaringan — aplikasi seperti email dan Web, yang digunakan sebagian besar siswa setiap hari. Setelah siswa memahami aplikasi, siswa kemudian dapat memahami layanan jaringan yang diperlukan untuk mendukung aplikasi tersebut. Siswa kemudian dapat, pada gilirannya, memeriksa berbagai cara di mana layanan tersebut dapat disediakan dan diterapkan di lapisan bawah. Meliputi aplikasi lebih awal dengan demikian memberikan motivasi untuk sisa teks.

Ketiga, pendekatan top-down memungkinkan instruktur untuk memperkenalkan pengembangan aplikasi jaringan pada tahap awal. Siswa tidak hanya melihat bagaimana aplikasi dan protokol populer bekerja, tetapi juga belajar betapa mudahnya membuat aplikasi jaringan dan protokol tingkat aplikasi mereka sendiri. Dengan pendekatan top-down, siswa mendapatkan paparan awal tentang gagasan pemrograman soket, model layanan, dan protokol —konsep penting yang muncul kembali di semua lapisan berikutnya. Dengan memberikan contoh pemrograman soket dengan Python, kami menyoroti ide sentral tanpa membingungkan siswa dengan kode yang rumit. Sarjana teknik elektro dan ilmu komputer seharusnya tidak mengalami kesulitan mengikuti kode Python.

Fokus Internet

Meskipun kami menghilangkan frase "Menampilkan Internet" dari judul buku ini pada edisi keempat, ini tidak berarti bahwa kami kehilangan fokus kami pada Internet! Memang, tidak ada yang lebih jauh dari kasus ini! Alih-alih, karena Internet telah menyebar begitu luas, kami merasa bahwa buku teks jaringan apa pun pasti memiliki pengaruh yang signifikan

fokus pada Internet, dan dengan demikian frasa ini agak tidak perlu. Kami terus menggunakan arsitektur dan protokol Internet sebagai kendaraan utama untuk mempelajari konsep jaringan komputer dasar yang menyenangkan. Tentu saja, kami juga menyertakan konsep dan protokol dari arsitektur jaringan lain. Tapi sorotan jelas ada di jaringan Inter, sebuah fakta yang tercermin dalam penyusunan buku ini di seputar arsitektur lima lapis Internet: lapisan aplikasi, transportasi, jaringan, tautan, dan fisik.

Manfaat lain dari menyoroti Internet adalah sebagian besar mahasiswa ilmu komputer dan teknik elektro sangat ingin belajar tentang Internet dan protokolnya.

Mereka tahu bahwa Internet telah menjadi teknologi yang revolusioner dan mengganggu dan dapat melihat bahwa itu sangat mengubah dunia kita. Mengingat relevansi Internet yang sangat besar, para siswa secara alami ingin tahu tentang apa yang "di balik terpal". Dengan demikian, mudah bagi instruktur untuk membuat siswa bersemangat tentang prinsip-prinsip dasar saat menggunakan Internet sebagai fokus panduan.

Mengajar Prinsip Jaringan

Dua fitur unik buku ini—pendekatan dari atas ke bawah dan fokusnya pada Internet—telah muncul di judul buku kami. Jika kita dapat memasukkan frasa *ketiga* ke dalam subjudul, itu akan berisi kata *prinsip*. Bidang jaringan sekarang sudah cukup matang sehingga sejumlah isu fundamental penting dapat diidentifikasi. Sebagai contoh, pada lapisan transport, masalah mendasar termasuk komunikasi yang dapat diandalkan melalui lapisan jaringan yang tidak dapat diandalkan, pembentukan/penghancuran koneksi dan handshaking, kongesti dan kontrol aliran, dan multiplexing. Dua masalah lapisan jaringan yang sangat penting dan menyenangkan adalah menentukan jalur "baik" antara dua router dan menghubungkan sejumlah besar jaringan heterogen. Di lapisan tautan, masalah mendasar adalah berbagi saluran akses ganda. Dalam keamanan jaringan, teknik untuk menyediakan kerahasiaan, autentikasi, dan integritas pesan semuanya didasarkan pada dasar kriptografi. Teks ini mengidentifikasi isu-isu jaringan dasar dan pendekatan studi untuk mengatasi masalah ini. Siswa yang mempelajari prinsip-prinsip ini akan memperoleh pengetahuan dengan "umur simpan" yang panjang—jauh setelah standar dan protokol jaringan saat ini menjadi usang, prinsip-prinsip yang terkandung di dalamnya akan tetap penting dan relevan. Kami percaya bahwa kombinasi penggunaan Internet untuk mendapatkan kaki siswa di pintu dan kemudian menekankan masalah fundamental dan pendekatan solusi akan memungkinkan siswa untuk dengan cepat memahami hampir semua teknologi jaringan.

Situs Web

Setiap salinan baru dari buku teks ini mencakup enam bulan akses ke situs Web Pendamping untuk semua pembaca buku di <http://www.pearsonhighered.com/kurose-ross>, yang meliputi:

- *Materi pembelajaran interaktif.* Komponen baru yang penting dari edisi keenam adalah materi pembelajaran online dan interaktif yang diperluas secara signifikan. Situs Web Pendamping buku sekarang berisi VideoNotes—presentasi video dari

topik penting di seluruh buku yang dilakukan oleh penulis, serta panduan solusi untuk masalah yang serupa dengan yang ada di akhir bab.

Kami juga telah menambahkan Latihan Interaktif yang dapat membuat (dan memberikan solusi untuk) masalah yang mirip dengan masalah akhir bab yang dipilih. Karena siswa dapat menghasilkan (dan melihat solusi untuk) contoh masalah serupa dalam jumlah yang tidak terbatas, mereka dapat bekerja sampai materi benar-benar dikuasai. Kami telah menyemai situs Web dengan VideoNotes dan masalah online untuk bab 1 sampai 5 dan akan terus menambah dan memperbarui materi ini secara aktif dari waktu ke waktu. Seperti pada edisi sebelumnya, situs Web berisi applet Java interaktif yang menggerakkan banyak konsep kunci jaringan. Situs ini juga memiliki kuis interaktif yang memungkinkan siswa memeriksa pemahaman dasar mereka tentang materi pelajaran. Profesor dapat mengintegrasikan fitur interaktif ini ke dalam perkuliahan mereka atau menggunakan sebagai lab mini.

- *Materi teknis tambahan.* Karena kami telah menambahkan materi baru di setiap edisi buku kami, kami harus menghapus cakupan beberapa topik yang ada untuk menjaga panjang buku tetap dapat diatur. Misalnya, untuk memberi ruang bagi materi baru di edisi ini, kami telah menghapus materi di jaringan ATM dan protokol RTSP untuk multimedia. Materi yang muncul di edisi teks sebelumnya masih diminati, dan dapat ditemukan di situs web buku tersebut.
- *Pemrograman tugas.* Situs Web juga menyediakan sejumlah tugas pemrograman terperinci, yang meliputi membangun server Web multithreaded, membangun klien email dengan antarmuka GUI, memprogram sisi pengirim dan penerima dari protokol transportasi data yang andal, memprogram algoritma perutean terdistribusi , dan banyak lagi. • *Laboratorium Wireshark.* Pemahaman seseorang tentang protokol jaringan dapat sangat diperdalam dengan melihatnya beraksi. Situs Web menyediakan banyak tugas Wireshark yang memungkinkan siswa untuk benar-benar mengamati urutan pesan yang dipertukarkan antara dua entitas protokol. Situs Web mencakup laboratorium hiu Wire terpisah di HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, WiFi, SSL, dan melacak semua protokol yang terlibat dalam memenuhi permintaan untuk mengambil halaman web.

Kami akan terus menambahkan lab baru dari waktu ke waktu.

Fitur Pedagogis

Kami masing-masing telah mengajar jaringan komputer selama lebih dari 20 tahun. Bersama-sama, kami membawa lebih dari 50 tahun pengalaman mengajar ke teks ini, selama itu kami telah mengajar ribuan siswa. Kami juga telah menjadi peneliti aktif dalam jaringan komputer selama ini. (Faktanya, Jim dan Keith pertama kali bertemu satu sama lain sebagai mahasiswa master dalam kursus jaringan komputer yang diajarkan oleh Mischa Schwartz pada tahun 1979 di Universitas Columbia.) Kami pikir semua ini memberi kami perspektif yang baik tentang di mana jaringan telah ada dan di mana kemungkinannya. pergi di masa depan. Namun demikian, kami telah menolak godaan untuk membiaskan materi dalam buku ini

menuju proyek penelitian hewan peliharaan kita sendiri. Kami pikir Anda dapat mengunjungi situs Web pribadi kami jika Anda tertarik dengan penelitian kami. Jadi, buku ini adalah tentang jaringan komputer modern—ini tentang protokol dan teknologi kontemporer serta prinsip-prinsip dasar di balik protokol dan teknologi ini. Kami juga percaya bahwa belajar (dan mengajar!) tentang jaringan bisa menyenangkan. Selera humor, penggunaan analogi, dan contoh dunia nyata dalam buku ini diharapkan akan membuat materi ini lebih menyenangkan.

Suplemen untuk Instruktur

Kami menyediakan paket suplemen lengkap untuk membantu instruktur dalam mengajar kursus ini. Materi ini dapat diakses dari Pusat Sumber Daya Instruktur Pearson (<http://www.pearsonhighered.com/irc>). Kunjungi Pusat Sumber Daya Instruktur atau kirim email ke computing@aw.com untuk informasi tentang cara mengakses suplemen instruktur ini.

- *Slide PowerPoint®*. Kami menyediakan slide PowerPoint untuk sembilan bab. Slide telah sepenuhnya diperbarui dengan edisi keenam ini. Slide mencakup setiap bab secara rinci. Mereka menggunakan grafik dan animasi (daripada hanya mengandalkan peluru teks yang monoton) untuk membuat slide menarik dan menarik secara visual. Kami menyediakan slide PowerPoint asli sehingga Anda dapat menyesuaikannya agar sesuai dengan kebutuhan pengajaran Anda sendiri. Beberapa dari slide ini telah disumbangkan oleh instruktur lain yang telah mengajar dari buku kami. • *Solusi pekerjaan rumah*. Kami menyediakan manual solusi untuk masalah pekerjaan rumah dalam teks, tugas pemrograman, dan lab Wireshark. Seperti disebutkan sebelumnya, kami telah memperkenalkan banyak masalah pekerjaan rumah baru di lima bab pertama buku ini.

Dependensi Bab

Bab pertama teks ini menyajikan ikhtisar mandiri tentang kerja jaringan komputer. Memperkenalkan banyak konsep kunci dan terminologi, bab ini menetapkan tahapan untuk sisanya buku ini. Semua bab lainnya secara langsung bergantung pada bab pertama ini. Setelah menyelesaikan Bab 1, kami merekomendasikan agar instruktur membahas Bab 2 hingga 5 secara berurutan, mengikuti filosofi top-down kami. Masing-masing dari kelima bab ini memanfaatkan materi dari bab-bab sebelumnya. Setelah menyelesaikan lima bab pertama, instruktur memiliki sedikit fleksibilitas. Tidak ada saling ketergantungan di antara empat bab terakhir, sehingga dapat diajarkan dalam urutan apa pun. Namun, masing-masing dari empat bab terakhir bergantung pada materi di lima bab pertama. Banyak instruktur pertama-tama mengajarkan lima bab pertama dan kemudian mengajarkan salah satu dari empat bab terakhir untuk “hidangan penutup”.

Satu Catatan Terakhir: Kami Ingin Mendengar dari Anda

Kami mendorong siswa dan instruktur untuk mengirim email kepada kami dengan komentar apa pun yang mungkin mereka miliki tentang buku kami. Sungguh luar biasa bagi kami mendengar dari begitu banyak instruktur dan siswa dari seluruh dunia tentang empat edisi pertama kami. Kami telah memasukkan banyak saran ini ke dalam edisi buku selanjutnya. Kami juga mendorong instruktur untuk mengirimkan kami soal (dan solusi) pekerjaan rumah baru yang akan melengkapi masalah pekerjaan rumah saat ini. Kami akan memposting ini di bagian khusus instruktur di situs Web. Kami juga mendorong instruktur dan siswa untuk membuat applet Java baru yang mengilustrasikan konsep dan protokol dalam buku ini. Jika Anda memiliki applet yang menurut Anda sesuai untuk teks ini, harap kirimkan kepada kami. Jika applet (termasuk notasi dan terminologi) sesuai, kami akan dengan senang hati memasukkannya ke dalam situs web teks, dengan referensi yang sesuai untuk pembuat applet.

Jadi, seperti kata pepatah, "Simpan kartu dan surat itu!" Serius, tolong *terus* kirimkan URL yang menarik, tunjukkan salah ketik, tidak setuju dengan klaim kami, dan beri tahu kami apa yang berhasil dan apa yang tidak. Beri tahu kami apa yang menurut Anda harus atau tidak boleh disertakan dalam edisi berikutnya. Kirim email Anda ke kurose@cs.umass.edu dan ross@poly.edu.

Terima kasih

Sejak kami mulai menulis buku ini pada tahun 1996, banyak orang telah memberi kami bantuan yang tak ternilai dan telah berpengaruh dalam membentuk pemikiran kami tentang cara terbaik mengatur dan mengajarkan kursus jaringan. Kami ingin mengucapkan TERIMA KASIH yang sebesar-besarnya kepada semua orang yang telah membantu kami sejak draf awal buku ini, hingga edisi kelima ini. Kami juga *sangat* berterima kasih kepada ratusan pembaca dari seluruh dunia—mahasiswa, dosen, praktisi—yang telah mengirimkan pemikiran dan komentar pada edisi buku sebelumnya dan saran untuk edisi buku yang akan datang. Terima kasih khusus ditujukan kepada:

Al Aho (Universitas Kolombia)
Hisham Al-Mubaid (Universitas Houston-Clear Lake)
Pratima Akkunoor (Universitas Negeri Arizona)
Paul Amer (Universitas Delaware)
Shamiul Azom (Universitas Negeri Arizona)
Lichun Bao (Universitas California di Irvine)
Paul Barford (Universitas Wisconsin)
Bobby Bhattacharjee (Universitas Maryland)
Steven Bellovin (Universitas Columbia)
Pravin Bhagwat (Wibhu)
Supratik Bhattacharyya (sebelumnya di Sprint)
Ernst Biersack (Institut Eurecom)

Shahid Bokhari (Universitas Teknik & Teknologi, Lahore)
Jean Bolot (Penelitian Technicolor)
Daniel Brushteyn (mantan mahasiswa Universitas Pennsylvania)
Ken Calvert (Universitas Kentucky)
Evandro Cantu (Universitas Federal Santa Catarina)
Kasus Jeff (Penelitian SNMP Internasional)
Jeff Chaltas (Sprint)
Vinton Cerf (Google)
Byung Kyu Choi (Universitas Teknologi Michigan)
Bram Cohen (BitTorrent, Inc.)
Constantine Coutras (Universitas Pace)
John Daigle (Universitas Mississippi)
Edmundo A. de Souza e Silva (Universitas Federal Rio de Janeiro)
Philippe Decuetos (Institut Eurecom)
Christophe Diot (Penelitian Technicolor)
Prithula Dhunghel (Akamai)
Deborah Estrin (Universitas California, Los Angeles)
Michalis Faloutsos (Universitas California di Riverside)
Wu-chi Feng (Lembaga Pascasarjana Oregon)
Sally Floyd (ICIR, Universitas California di Berkeley)
Paul Francis (Institut Max Planck)
Lixin Gao (Universitas Massachusetts)
JJ Garcia-Luna-Aceves (Universitas California di Santa Cruz)
Mario Gerla (Universitas California di Los Angeles)
David Goodman (NYU-Poly)
Yang Guo (Alcatel/Lucent Bell Labs)
Tim Griffin (Universitas Cambridge)
Max Hailperin (Universitas Gustavus Adolphus)
Bruce Harvey (Universitas A&M Florida, Universitas Negeri Florida)
Carl Hauser (Universitas Negeri Washington)
Rachelle Heller (Universitas George Washington)
Phillipp Hoschka (INRIA/W3C)
Wen Hsin (Universitas Park)
Albert Huang (mantan mahasiswa Universitas Pennsylvania)
Cheng Huang (Penelitian Microsoft)
Esther A. Hughes (Universitas Persemakmuran Virginia)
Van Jacobson (Xerox PARC)
Pinak Jain (mantan mahasiswa NYU-Poly)
Jobin James (Universitas California di Riverside)
Sugih Jamin (University of Michigan)
Shivkumar Kalyanaraman (Penelitian IBM, India)
Jussi Kangasharju (Universitas Helsinki)
Rumah Sneha (Universitas Utah)
Parviz Kermani (sebelumnya dari IBM Research)

Hyojin Kim (mantan mahasiswa Universitas Pennsylvania)
Leonard Kleinrock (Universitas California di Los Angeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Universitas Negeri Arizona)
Rakesh Kumar (Bloomberg)
Miguel A. Labrador (Universitas Florida Selatan)
Simon Lam (Universitas Texas)
Steve Lai (Universitas Negeri Ohio)
Tom LaPorta (Universitas Negeri Penn)
Tim-Berners Lee (Konsorsium World Wide Web)
Arnaud Legout (INRIA)
Lee Leitner (Universitas Drexel)
Brian Levine (Universitas Massachusetts)
Chunchun Li (mantan mahasiswa NYU-Poly)
Yong Liu (NYU-Poli)
William Liang (mantan mahasiswa Universitas Pennsylvania)
Willis Marti (Universitas A&M Texas)
Nick McKeown (Universitas Stanford)
Josh McKinzie (Universitas Park)
Deep Medhi (Universitas Missouri, Kansas City)
Bob Metcalfe (Grup Data Internasional)
Sue Moon (KAIST)
Jenni Moyer (Komcast)
Erich Nahum (Penelitian IBM)
Christos Papadopoulos (Universitas Sate Colorado)
Craig Partridge (Teknologi BBN)
Radia Perlman (Intel)
Jitendra Padhye (Penelitian Microsoft)
Vern Paxson (Universitas California di Berkeley)
Kevin Phillips (Sprint)
George Polyzos (Universitas Ekonomi dan Bisnis Athena)
Sriram Rajagopalan (Universitas Negeri Arizona)
Ramachandran Ramjee (Penelitian Microsoft)
Ken Reek (Institut Teknologi Rochester)
Martin Reisslein (Universitas Negeri Arizona)
Jennifer Rexford (Universitas Princeton)
Leon Reznik (Institut Teknologi Rochester)
Pablo Rodríguez (Telepon)
Sumit Roy (Universitas Washington)
Avi Rubin (Universitas Johns Hopkins)
Dan Rubenstein (Columbia University)
Douglas Sallane (Universitas John Jay)
Despina Saparilla (Sistem Cisco)
John Schanz (Komcast)

Henning Schulzrinne (Universitas Columbia)
Mischa Schwartz (Universitas Columbia)
Ardash Sethi (Universitas Delaware)
Harish Sethu (Universitas Drexel)
K. Sam Shanmugan (Universitas Kansas)
Prashant Shenoy (Universitas Massachusetts)
Clay Shields (Universitas Georgetown)
Subin Shrestra (Universitas Pennsylvania)
Bojie Shu (mantan siswa NYU-Poly)
Mihail L. Sichitiu (Universitas Negeri NC)
Peter Steenkiste (Universitas Carnegie Mellon)
Tatsuya Suda (Universitas California di Irvine)
Kin Sun Tam (Universitas Negeri New York di Albany)
Don Towsley (Universitas Massachusetts)
David Turner (Universitas Negeri California, San Bernardino)
Nitin Vaidya (Universitas Illinois)
Michele Weigle (Universitas Clemson)
David Wetherall (Universitas Washington)
Ira Winston (Universitas Pennsylvania)
Di Wu (Universitas Sun Yat-sen)
Shirley Wynn (NYU-Poly)
Raj Yavatkar (Intel)
Yechiam Yemini (Universitas Columbia)
Ming Yu (Universitas Negeri New York di Binghamton)
Ellen Zegura (Institut Teknologi Georgia)
Hongkong Zhang (Universitas Suffolk)
Hui Zhang (Universitas Carnegie Mellon)
Lixia Zhang (Universitas California di Los Angeles)
Meng Zhang (mantan siswa NYU-Poly)
Shuchun Zhang (mantan mahasiswa Universitas Pennsylvania)
Xiaodong Zhang (Universitas Negeri Ohio)
ZhiLi Zhang (Universitas Minnesota)
Phil Zimmermann (konsultan independen)
Cliff C. Zou (Universitas Florida Tengah)

Kami juga ingin berterima kasih kepada seluruh tim Addison-Wesley—khususnya, Michael Hirsch, Marilyn Lloyd, dan Emma Snider—yang telah melakukan pekerjaan yang benar-benar luar biasa pada edisi keenam ini (dan yang telah bekerja sama dengan dua penulis yang sangat rewel yang tampaknya penipu). secara genetis tidak dapat memenuhi tenggat waktu!. Terima kasih juga kepada seniman kami, Janet Theurer dan Patrice Rossi Calkin, atas karya mereka pada sosok-sosok cantik dalam buku ini, dan kepada Andrea Stefanowicz dan timnya di PreMediaGlobal atas karya produksi mereka yang luar biasa pada edisi ini. Akhirnya, terima kasih yang sebesar-besarnya kami sampaikan kepada Michael Hirsch, editor kami di Addison Wesley, dan Susan Hartman, mantan editor kami di Addison-Wesley. Buku ini tidak akan menjadi seperti sekarang ini (dan mungkin tidak akan menjadi sama sekali) tanpa pengelolaan mereka yang anggun, dorongan terus-menerus, kesabaran yang hampir tak terbatas, humor yang baik, dan ketekunan.

Daftar isi

Bab 1 Jaringan Komputer dan Internet	1
1.1 Apa itu Internet?	2
1.1.1 Uraian Mur dan Baut	2
1.1.2 Uraian Layanan	5
1.1.3 Apa Itu Protokol?	7
1.2 Tepi Jaringan	9
1.2.1 Jaringan Akses	12
1.2.2 Media	18
Fisik	
1.3 Inti Jaringan	18
Pengalihan Paket	22
1.3.1 Sirkuit	22
1.3.2 Kehilangan, dan Throughput	27
1.3.3 Jaringan dalam Jaringan	32
1.4 Switched	35
1.4.1 Gambaran Umum Delay pada Jaringan	35
1.4.2 Delay Antrian dan Packet Loss	39
1.4.3 End-to-End Delay	39
1.4.4 Throughput pada Lapisan Protokol Jaringan	42
Komputer dan Model Layanannya	
1.5.1 Arsitektur Berlapis	42
1.5.2 Enkapsulasi	44
1.6 Networks Under Attack	44
1.7 Sejarah Jaringan	47
1.7.1 Komputer dan Internet	47
1.7.2 Perkembangan Packet Switching: 1961–1972	47
1.7.3 Jaringan Proprietary dan Internetworking: 1972–1980	53
1.7.4 Proliferasi Jaringan: 1980–1990	55
1.7.5 Ledakan Internet: The 1990-an	55
1.8 Ringkasan	60
Pekerjaan Rumah Masalah dan Pertanyaan	62
Wireshark Lab	63
Wawancara: Leonard Kleinrock	64
	65
	66
	68
	78
	80

xviii Daftar isi

Bab 2	Prinsip Lapisan Aplikasi	83
2.1	Aplikasi Jaringan 2.1.1 Arsitektur	84
	Aplikasi Jaringan 2.1.2 Proses	86
	Berkomunikasi 2.1.3 Layanan	88
	Transportasi yang Tersedia untuk Aplikasi	91
	2.1.4 Layanan Transportasi yang Disediakan oleh Internet 2.1.5 Protokol Lapisan Aplikasi	93
	2.1.6 Aplikasi Jaringan yang Dicakup dalam Buku Ini 2.2 Web dan HTTP 2.2.1 Ikhtisar HTTP 2.2.2	96
	Koneksi Non-Persistent dan Persistent 2.2.3	97
	Format Pesan HTTP 2.2.4 Interaksi Pengguna-Server: Cookie 2.2.5 Web Caching 2.2.6	98
	Transfer Berkas GET Bersyarat : FTP 2.3.1	100
	Perintah dan Balasan FTP Surat Elektronik di Internet 2.4.1 SMTP 2.4.2 Perbandingan dengan HTTP 2.4.3 Format Pesan Surat 2.4.4	103
2.3	Protokol Akses Surat	110
2.4		114
		116
		118
		118
		121
		124
		125
		125
2.5	DNS—Layanan Direktori Internet 2.5.1	130
	Layanan yang Disediakan oleh DNS 2.5.2	131
	Gambaran Umum tentang Cara Kerja	133
	DNS 2.5.3 Data dan Pesan DNS Aplikasi	139
2.6	Peer-to-Peer 2.6.1 Distribusi File P2P	144
	2.6.2 Tabel Hash Terdistribusi (DHT)	145
		151
2.7	Pemrograman Soket: Membuat Aplikasi Jaringan	156
	2.7.1 Pemrograman Soket dengan UDP	157
	2.7.2 Pemrograman Soket dengan TCP	163
2.8	Ringkasan Pekerjaan Rumah Masalah dan Pertanyaan Tugas Pemrograman Soket Wireshark	168
	Labs: HTTP, DNS Wawancara: Marc Andreessen	1181
		179
		2

Bab 3 Lapisan Transport**185**

3.1	Pendahuluan dan Layanan Lapisan Transport 3.1.1	186
	Hubungan Antara Lapisan Transport dan Lapisan Jaringan 3.1.2 Tinjauan	186
	Lapisan Transport pada Internet Multiplexing dan Demultiplexing Transport	189
3.2	Tanpa Sambungan: UDP 3.3.1 Struktur Segmen UDP 3.3.2 UDP Checksum	191
3.3	Prinsip Transfer Data yang Andal 3.4 .1 Membangun Protokol Transfer Data yang Andal 3.4.2 Protokol Transfer Data yang Andal Pipeline 3.4.3 Go-Back-N (GBN)	198
3.4		202
		204
		206
		215
		218
	3.4.4 Pengulangan Selektif (SR)	223
3.5	Transportasi Berorientasi Koneksi: TCP 3.5.1	230
	Koneksi TCP 3.5.2 Struktur Segmen TCP	231
	3.5.3 Estimasi Waktu Pulang Pergi dan	233
	Timeout 3.5.4 Transfer Data yang Andal 3.5.5 Kontrol Aliran	238
	3.5.6 Prinsip Manajemen Koneksi TCP untuk Kontrol	242
	Kemacetan 3.6 .1 Penyebab dan Biaya Kemacetan 3.6.2	250
	Pendekatan untuk Pengendalian Kemacetan 3.6.3 Contoh	252
3.6	Pengendalian Kemacetan Berbantuan Jaringan:	259
		259
		265
	Pengendalian Kemacetan ATM ABR	266
3.7	Pengendalian Kemacetan TCP 3.7.1 Kewajaran	269
		279
3.8	Ringkasan	283
	Pekerjaan Rumah Masalah dan Pertanyaan	285
	Tugas Pemrograman Wireshark Labs: TCP,	300
	UDP Wawancara: Van Jacobson	301
		302

Bab 4 Lapisan Jaringan**305**

4.1	Pendahuluan	306
	4.1.1 Forwarding dan Routing 4.1.2	308
	Model Layanan Jaringan Virtual Circuit	310
4.2	dan Jaringan Datagram 4.2.1 Jaringan Virtual-Circuit 4.2.2 Jaringan Datagram 4.2.3 Asal VC	313
	dan Jaringan Datagram	317
		319

xx Daftar isi

4.3	Apa yang Ada di Dalam Router?	320
4.3.1	Pemrosesan Input 4.3.2	322
4.3.2	Pengalihan 4.3.3 Pemrosesan	324
4.3.3	Output 4.3.4 Dimana Terjadi	326
4.3.4	Antrian?	327
4.3.5	Bidang Kontrol Perutean	331
4.4	Protokol Internet (IP): Penerusan dan Pengalamatan di Internet	331
4.4.1	Format Datagram	331
4.4.2	Pengalamatan IPv4 4.4.3 Protokol Pesan Kontrol Internet (ICMP)	332
4.4.3		338
4.4.4		353
4.4.5		356
4.5	Perintisan Singkat ke dalam Keamanan IP 4.5	362
4.5.1	Algoritma Perutean Link-State	363
4.5.2	(LS) 4.5.2 Algoritma Perutean Distance-Vector (DV) 4.5.3	366
4.5.3	Perutean Hierarkis 4.6 Perutean di Internet 4.6.1 Perutean Intra-	371
4.6.1	AS di Internet: RIP 4.6.2 Perutean Intra-AS di Internet: OSPF 4.6.3	379
4.6.2	AS di Internet: OSPF 4.6.3	379
4.6.3	Perutean Antar-AS: Perutean Siaran dan Multicast BGP 4.7.1 Algoritma	383
4.7.1	Perutean Siaran 4.7.2 Multicast	384
4.7.2		388
4.7.3		390
4.7.4		399
4.7.5		400
4.7.6		405
4.8	Ringkasan	412
	Pekerjaan Rumah Masalah dan Pertanyaan	413
	Tugas Pemrograman Wireshark Labs: IP,	429
	ICMP Wawancara: Vinton G. Cerf	430
		431

Bab 5 Lapisan Tautan: Tautan, Jaringan Akses, dan LAN**433**

5.1	Pengantar Lapisan Tautan 5.1.1	434
5.1.1	Layeran yang Disediakan oleh Lapisan Tautan 5.1.2 Di	436
5.1.2	mana Lapisan Tautan Diimplementasikan?	437
5.2	Teknik Deteksi Kesalahan dan Koreksi 5.2.1 Cek	438
5.2.1	Paritas 5.2.2 Metode Checksumming 5.2.3 Cyclic	440
5.2.2	Redundancy Check (CRC)	442
5.2.3		443
5.3	Tautan dan Protokol Akses Ganda 5.3.1	445
5.3.1	Protokol Pemisahan Saluran 5.3.2 Protokol	448
5.3.2	Akses Acak 5.3.3 Protokol Bergiliran 5.3.4	449
5.3.3	DOCSIS: Protokol Lapisan Tautan untuk Akses	459
5.3.4	Internet Kabel	460

5.4 Switched Local Area Networks 5.4.1	461
Pengalaman Link-Layer dan ARP 5.4.2 Ethernet	462
5.4.3 Link-Layer Switches 5.4.4 Virtual Local Area	469
Networks (VLAN)	476
	482
5.5 Virtualisasi Tautan: Jaringan sebagai Lapisan Tautan 5.5.1	486
Multiprotocol Label Switching (MPLS)	487
5.6 Jaringan Pusat Data 5.7	490
Retrospektif: Sejarah dalam Kehidupan	495
Permintaan Hukuman WEB 5.7.1 Memulai: DHCP, UDP, IP, dan	495
Ethernet 5.7.2 Masih Memulai: DNS dan ARP 5.7.3 Masih Memulai:	495
Intra- Perutean Domain ke Server DNS 5.7.4 Interaksi Klien-Server	497
Web: TCP dan HTTP	498
	499
5.8 Rangkuman	500
Pekerjaan Rumah Masalah dan Pertanyaan	502
Wireshark Labs: Ethernet dan ARP, DHCP	510
Wawancara: Simon S. Lam	511

Bab 6 Jaringan Nirkabel dan Seluler

513

6.1 Pendahuluan	514
6.2 Link Nirkabel dan Karakteristik Jaringan 6.2.1 CDMA	519
WiFi: LAN Nirkabel 802.11 6.3.1 Arsitektur 802.11	522
6.3 6.3.2 Protokol MAC 802.11 6.3.3 Bingkai IEEE 802.11	526
6.3.4 Mobilitas dalam Subnet IP yang Sama 6.3.5	527
Fitur Canggih dalam 802.11 6.3.6 Jaringan Area	531
Pribadi: Akses Internet Seluler Bluetooth dan Zigbee	537
6.4.1 Tinjauan Arsitektur Jaringan Seluler 6.4.2	541
Jaringan Data Seluler 3G: Memperluas Internet ke	542
Pelanggan Seluler 6.4.3 Aktif ke 4G: Manajemen Mobilitas LTE:	544
6.4 Prinsip 6.5 .1 Pengalaman 6.5.2 Routing ke Mobile Node	546
	547
	550
	553
6.5	555
	557
	559
6.6 Mobile IP 6.7	564
Mengelola Mobilitas dalam Jaringan Seluler 6.7.1 Merutekan	570
Panggilan ke Pengguna Seluler 6.7.2 Handoff	571
dalam GSM	572

xxii Daftar isi

6.8 Nirkabel dan Mobilitas: Dampak pada Protokol Lapisan Tinggi 6.9	575
Rangkuman Masalah Pekerjaan Rumah dan Pertanyaan Wireshark Lab: IEEE	578
802.11 (WiFi)	578
583	
Wawancara: Deborah Estrin	584
Bab 7 Jaringan Multimedia Aplikasi Jaringan Multimedia	587
7.1	588
7.1.1 Sifat Video 7.1.2 Sifat Audio 7.1.3 Jenis	588
Aplikasi Jaringan Multimedia Streaming Video	590
Tersimpan 7.2.1 Streaming UDP 7.2.2	591
Streaming HTTP 7.2.3 Streaming Adaptif dan DASH 7.2.4	593
7.2	595
Jaringan Distribusi Konten 7.2.5 Studi Kasus: Netflix, YouTube,	595
dan Kakan Voice-over-IP 7.3.1 Batasan Layanan IP Best-Effort	596
7.3.2 Menghilangkan Jitter pada Penerima untuk Audio 7.3.3	600
Memulihkan dari Kehilangan Paket 7.3.4 Studi Kasus: VoIP	602
dengan Protokol Skype untuk Aplikasi Percakapan Real-Time	608
7.4.1 RTP 7.4.2 SIP 7.5 Dukungan Jaringan untuk Multimedia	612
7.3	612
7.5.1 Dimensi Jaringan Best-Effort 7.5.2 Menyediakan Beberapa	614
Kelas Layanan 7.5.3 Diffserv 7.5.4 Jaminan Kualitas Layanan	617
(QoS) Per Sambungan:	620
7.4	623
Reservasi Sumber Daya dan Penerimaan Panggilan	624
7.6 Ringkasan	627
Masalah Pekerjaan Rumah dan Pertanyaan	632
Tugas Pemrograman Wawancara: Henning Schulzrinne	634
	636
	648
Bab 8 Keamanan dalam Jaringan Komputer	671
8.1 Apa Itu Keamanan Jaringan? 8.2	672
Prinsip Kriptografi 8.2.1 Kripsi dan Kunci	675
Simetris 8.2.2 Enkripsi Publik	676
Plik	683

8.3 Integritas Pesan dan Tanda Tangan Digital 8.3.1	688
Fungsi Hash Kriptografis 8.3.2 Kode Otentikasi	689
Pesan 8.3.3 Otentikasi Titik Akhir Tanda	691
Tangan Digital 8.4.1 Protokol Otentikasi <i>ap1.0</i>	693
8.4 8.4.2 Protokol Otentikasi <i>ap2.0</i> 8.4.3 Protokol	700
Otentikasi <i>ap3.0</i> 8.4.4 Authentication Protocol	700
<i>ap3.1</i> 8.4.5 Authentication Protocol <i>ap4.0</i>	701
Mengamankan E-Mail 8.5.1 Secure E-Mail 8.5.2	702
PGP Mengamankan Koneksi TCP: SSL 8.6.1	703
Gambaran Besar 8.6.2 Gambaran Lebih	703
8.5 Lengkap 8.7 Keamanan Lapisan Jaringan: IPsec dan Jaringan Privat Virtual 8.7.1 IPsec dan Jaringan Privat Virtual (VPN)	705
8.6	711
	713
	716
	718
	718
8.7.2 Protokol AH dan ESP 8.7.3 Asosiasi	720
Keamanan 8.7.4 Datagram IPsec 8.7.5	720
IKE: Manajemen Kunci di IPsec	721
Mengamankan LAN Nirkabel 8.8.1 Wired	725
8.8 8.8.2 Keamanan	726
8.9 Operasional IEEE 802.11i: Firewall dan Sistem Deteksi Intrusi 8.9.1 Firewall	731
8.9.2 Sistem Deteksi Intrusi 8.10 Rangkuman Masalah Pekerjaan Rumah	731
dan Pertanyaan Wireshark Lab: SSL IPsec Lab Wawancara: Steven M. Bellovin	739
	742
	744
	752
	752
	752 731 731 739 7

Bab 9 Manajemen Jaringan 9.1 Apa itu Manajemen**755**

Jaringan?	756
9.2 Infrastruktur untuk Manajemen Jaringan 9.3 Kerangka	760
Manajemen Standar Internet	764
9.3.1 Struktur Informasi Manajemen: SMI 9.3.2 Basis	766
Informasi Manajemen: MIB	770

xxiv Daftar isi

9.3.3 Operasi Protokol SNMP dan Pemetaan Transportasi 9.3.4	772
Keamanan dan Administrasi 9.4 ASN.1 9.5 Kesimpulan Masalah	775
Pekerjaan Rumah dan Pertanyaan <i>Wawancara: Jennifer Rexford</i>	778
	783
	783
	786
Referensi	789
Indeks	823

KOMPUTER JARINGAN

EDISI KEENAM

Pendekatan Top-Down



Halaman ini sengaja dikosongkan



BAB

1

Komputer Jaringan dan Internet

Internet saat ini bisa dibilang merupakan sistem rekayasa terbesar yang pernah dibuat oleh umat manusia, dengan ratusan juta komputer, tautan komunikasi, dan sakelar yang terhubung; dengan miliaran pengguna yang terhubung melalui laptop, tablet, dan smartphone; dan dengan serangkaian perangkat baru yang terhubung ke Internet seperti sensor, Web cam, konsol game, bingkai foto, dan bahkan mesin cuci. Mengingat bahwa Internet begitu besar dan memiliki begitu banyak komponen dan kegunaan yang beragam, apakah ada harapan untuk memahami cara kerjanya? Adakah prinsip dan struktur panduan yang dapat memberikan landasan untuk memahami sistem yang luar biasa besar dan kompleks ini? Dan jika demikian, mungkinkah belajar tentang jaringan komputer itu menarik *dan menyenangkan*? Untungnya, jawaban atas semua pertanyaan ini adalah YA! Memang, tujuan kami dalam buku ini adalah untuk memberi Anda pengantar modern ke bidang dinamis jaringan komputer, memberi Anda prinsip dan wawasan praktis yang Anda perlukan untuk memahami tidak hanya jaringan hari ini, tetapi juga jaringan besok.

Bab pertama ini menyajikan gambaran luas tentang jaringan komputer dan Internet. Tujuan kami di sini adalah untuk melukis gambaran luas dan mengatur konteks untuk siswa buku ini, untuk melihat hutan melalui pepohonan. Kita akan membahas banyak hal dalam bab pengantar ini dan membahas banyak bagian dari jaringan komputer, tanpa melupakan gambaran besarnya.

2 BAB 1 • JARINGAN KOMPUTER DAN INTERNET

Kami akan menyusun ikhtisar kami tentang jaringan komputer dalam bab ini sebagai berikut. Setelah memperkenalkan beberapa terminologi dan konsep dasar, pertama-tama kita akan memeriksa komponen perangkat keras dan perangkat lunak dasar yang membentuk jaringan. Kami akan mulai dari tepi jaringan dan melihat sistem akhir dan aplikasi jaringan yang berjalan di jaringan. Kami kemudian akan mengeksplorasi inti dari jaringan komputer, memeriksa tautan dan saklar yang mengangkut data, serta jaringan akses dan media fisik yang menghubungkan sistem akhir ke inti jaringan. Kita akan mempelajari bahwa Internet adalah jaringan dari jaringan, dan kita akan mempelajari bagaimana jaringan ini terhubung satu sama lain.

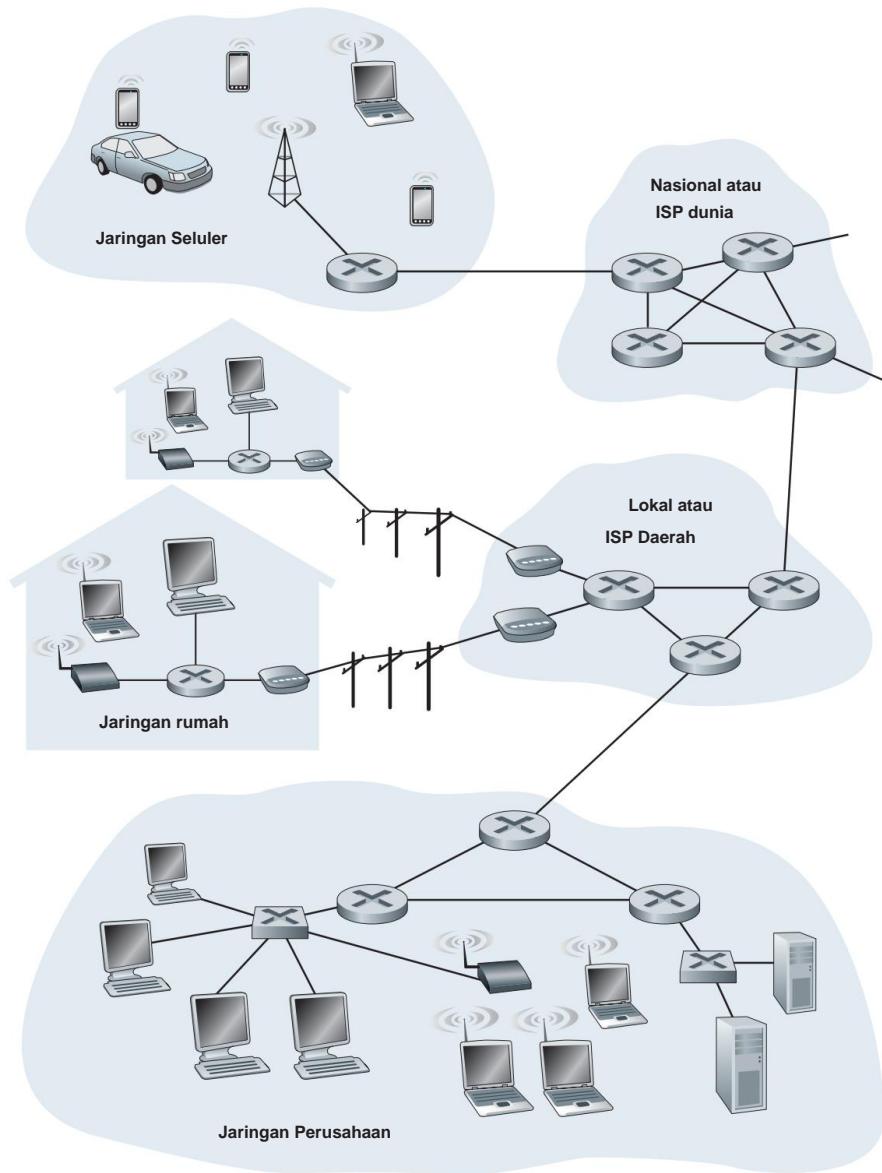
Setelah menyelesaikan ikhtisar tepi dan inti jaringan komputer ini, kami akan mengambil pandangan yang lebih luas dan lebih abstrak di paruh kedua bab ini. Kita akan memeriksa delay, loss, dan throughput data dalam jaringan komputer dan memberikan model kuantitatif sederhana untuk end-to-end throughput dan delay: model yang memperhitungkan delay transmisi, propagasi, dan antrian. Kami kemudian akan memperkenalkan beberapa prinsip arsitektur utama dalam jaringan komputer, yaitu, pelapisan protokol dan model layanan. Kita juga akan mengetahui bahwa jaringan komputer rentan terhadap berbagai jenis serangan; kami akan mensurvei beberapa serangan ini dan mempertimbangkan bagaimana jaringan komputer dapat dibuat lebih aman. Terakhir, kami akan menutup bab ini dengan sejarah singkat jaringan komputer.

1.1 Apa itu Internet?

Dalam buku ini, kita akan menggunakan Internet publik, jaringan komputer tertentu, sebagai kendaraan utama kita untuk mendiskusikan jaringan komputer dan protokolnya. Tapi apa *itu* Internet? Ada beberapa cara untuk menjawab pertanyaan ini. Pertama, kita dapat menjelaskan inti dan baut dari Internet, yaitu komponen perangkat keras dan perangkat lunak dasar yang membentuk Internet. Kedua, kita dapat menggambarkan Internet dalam hal infrastruktur kerja bersih yang menyediakan layanan untuk aplikasi terdistribusi. Mari kita mulai dengan deskripsi mur dan baut, menggunakan Gambar 1.1 untuk mengilustrasikan diskusi kita.

1.1.1 Deskripsi Mur dan Baut

Internet adalah jaringan komputer yang menghubungkan ratusan juta perangkat komputer di seluruh dunia. Belum lama berselang, perangkat komputasi ini utamanya adalah PC desktop tradisional, workstation Linux, dan apa yang disebut server yang menyimpan dan mengirimkan informasi seperti halaman Web dan pesan email. Akan tetapi, sistem akhir Internet nontradisional seperti laptop, telepon pintar, tablet, TV, konsol game, kamera web, mobil, perangkat penginderaan lingkungan, bingkai foto, dan sistem kelistrikan dan keamanan rumah terhubung ke Internet. Memang, istilah *jaringan komputer* mulai terdengar agak kuno, mengingat banyaknya perangkat nontradisional yang terhubung ke Internet. Dalam jargon Internet, semua perangkat ini disebut **host** atau **sistem akhir**. Pada Juli 2011, ada



Kunci:

**Gambar 1.1** Beberapa bagian dari Internet

4 BAB 1 • JARINGAN KOMPUTER DAN INTERNET

hampir 850 juta sistem akhir terhubung ke Internet [ISC 2012], belum termasuk smartphone, laptop, dan perangkat lain yang hanya sesekali terhubung ke Internet. Secara keseluruhan, diperkirakan lebih dari 2 miliar pengguna Internet [ITU 2011].

Sistem akhir dihubungkan bersama oleh jaringan **tautan komunikasi** dan **sakelar paket**. Kita akan melihat di Bagian 1.2 bahwa ada banyak jenis tautan komunikasi, yang dibuat dari berbagai jenis media fisik, termasuk kabel koaksial, kabel tembaga, serat optik, dan spektrum radio. Tautan yang berbeda dapat mentransmisikan data dengan laju yang berbeda, dengan **laju transmisi** tautan diukur dalam bit/detik.

Ketika satu sistem akhir memiliki data untuk dikirim ke sistem akhir lainnya, sistem akhir pengiriman membagi data dan menambahkan byte header ke setiap segmen. Paket informasi yang dihasilkan, yang dikenal sebagai **paket** dalam jargon jaringan komputer, kemudian dikirim melalui jaringan ke sistem akhir tujuan, di mana mereka disusun kembali menjadi data asli.

Sakelar paket mengambil paket yang tiba di salah satu tautan komunikasi masuk dan meneruskan paket itu di salah satu tautan komunikasi keluarnya. Sakelar paket datang dalam berbagai bentuk dan rasa, tetapi dua jenis yang paling menonjol di Internet saat ini adalah **router** dan **sakelar lapisan tautan**. Kedua jenis switch untuk paket ward menuju tujuan akhir mereka. Sakelar lapisan tautan biasanya digunakan dalam jaringan akses, sedangkan router biasanya digunakan dalam inti jaringan. Urutan tautan komunikasi dan sakelar paket yang dilalui oleh paket dari sistem ujung pengirim ke sistem ujung penerima dikenal sebagai rute **atau** jalur **melalui** jaringan. Jumlah pasti lalu lintas yang dibawa di Internet sulit diperkirakan, tetapi Cisco [Cisco VNI 2011] memperkirakan lalu lintas Internet global akan mencapai hampir 40 exabyte per bulan pada tahun 2012.

Jaringan packet-switched (yang mengangkut paket) dalam banyak hal serupa dengan jaringan transportasi jalan raya, jalan raya, dan persimpangan (yang mengangkut kendaraan). Pertimbangkan, misalnya, sebuah pabrik yang perlu memindahkan kargo dalam jumlah besar ke gudang tujuan yang jaraknya ribuan kilometer. Di pabrik, kargo disegmentasi dan dimuat ke dalam armada truk. Masing-masing truk kemudian berjalan sendiri-sendiri melalui jaringan jalan raya, jalan raya, dan persimpangan menuju gudang tujuan. Di gudang tujuan, kargo dibongkar dan dikelompokkan dengan sisa kargo yang datang dari kiriman yang sama. Jadi, dalam banyak hal, paket dianalogikan dengan truk, jalur komunikasi dianalogikan dengan jalan raya dan jalan raya, sakelar paket dianalogikan dengan persimpangan, dan sistem akhir dianalogikan dengan bangunan. Sama seperti sebuah truk mengambil jalan melalui jaringan transportasi, sebuah paket mengambil jalan melalui jaringan komputer.

Sistem akhir mengakses Internet melalui **Penyedia Layanan Internet (ISP)**, termasuk ISP perumahan seperti perusahaan kabel atau telepon lokal; ISP perusahaan; ISP universitas; dan ISP yang menyediakan akses WiFi di bandara, hotel, kedai kopi, dan tempat umum lainnya. Setiap ISP itu sendiri merupakan jaringan sakelar paket dan tautan komunikasi. ISP menyediakan berbagai jenis akses jaringan ke sistem akhir, termasuk akses broadband perumahan seperti modem kabel atau DSL,

akses jaringan area lokal berkecepatan tinggi, akses nirkabel, dan akses modem dial-up 56 kbps. ISP juga menyediakan akses Internet ke penyedia konten, menghubungkan situs Web langsung ke Internet. Internet adalah tentang menghubungkan sistem akhir satu sama lain, sehingga ISP yang menyediakan akses ke sistem akhir juga harus saling terhubung. ISP tingkat bawah ini saling terhubung melalui ISP tingkat atas nasional dan internasional seperti Level 3 Communications, AT&T, Sprint, dan NTT.

ISP tingkat atas terdiri dari router berkecepatan tinggi yang saling terhubung dengan tautan serat optik berkecepatan tinggi. Setiap jaringan ISP, baik tingkat atas atau tingkat bawah, dikelola secara independen, menjalankan protokol IP (lihat di bawah), dan sesuai dengan konvensi penamaan dan alamat tertentu. Kami akan memeriksa ISP dan interkoneksi lebih dekat di Bagian 1.3.

Sistem akhir, saklar paket, dan bagian lain dari Internet menjalankan **protokol** yang mengontrol pengiriman dan penerimaan informasi di dalam Internet. Transmission **Control Protocol (TCP)** dan **Internet Protocol (IP)** adalah dua protokol terpenting di Internet. Protokol IP menentukan format paket yang dikirim dan diterima di antara router dan sistem akhir. Protokol utama Internet secara kolektif dikenal sebagai **TCP/IP**. Kita akan mulai mempelajari protokol dalam bab pengantar ini. Tapi itu baru permulaan—banyak dari buku ini berkaitan dengan protokol jaringan komputer!

Mengingat pentingnya protokol ke Internet, penting bagi setiap orang untuk menyetujui apa yang dilakukan setiap protokol, sehingga orang dapat membuat sistem dan produk yang saling beroperasi. Di sinilah standar berperan. **Standar internet** dikembangkan oleh Internet Engineering Task Force (IETF) [IETF 2012].

Dokumen standar IETF disebut **permintaan komentar (RFC)**. RFC dimulai sebagai permintaan umum untuk komentar (karena itu namanya) untuk menyelesaikan masalah desain jaringan dan protokol yang dihadapi pendahulu ke Internet [Allman 2011]. RFC cenderung sangat teknis dan terperinci. Mereka mendefinisikan protokol seperti TCP, IP, HTTP (untuk Web), dan SMTP (untuk e-mail). Saat ini ada lebih dari 6.000 RFC. Badan lain juga menentukan standar untuk komponen jaringan, terutama untuk tautan jaringan. Komite Standar IEEE 802 LAN/MAN [IEEE 802 2012], misalnya, menentukan standar Ethernet dan WiFi nirkabel.

1.1.2 Uraian Layanan

Pembahasan kita di atas telah mengidentifikasi banyak bagian yang membentuk Internet. Tapi kita juga bisa menggambarkan Internet dari sudut yang sama sekali berbeda—yaitu, sebagai *infrastruktur yang menyediakan layanan untuk aplikasi*. Aplikasi ini termasuk surat elektronik, penjelajahan Web, jejaring sosial, pesan instan, Voice over-IP (VoIP), streaming video, permainan terdistribusi, berbagi file peer-to-peer (P2P), televisi melalui Internet, login jarak jauh, dan banyak, lebih banyak lagi. Aplikasi dikatakan sebagai **aplikasi terdistribusi**, karena melibatkan banyak sistem akhir yang bertukar data satu sama lain. Yang penting, aplikasi internet

6 BAB 1 • JARINGAN KOMPUTER DAN INTERNET

berjalan di sistem akhir—mereka tidak berjalan di sakelar paket di inti jaringan. Meskipun sakelar paket memfasilitasi pertukaran data di antara sistem akhir, mereka tidak peduli dengan aplikasi yang menjadi sumber atau wadah data.

Mari jelajahi lebih dalam apa yang dimaksud dengan infrastruktur yang menyediakan layanan untuk aplikasi. Untuk tujuan ini, misalkan Anda memiliki ide baru yang menarik untuk aplikasi Internet terdistribusi, yang mungkin sangat bermanfaat bagi umat manusia atau yang mungkin membuat Anda kaya dan terkenal. Bagaimana Anda bisa mengubah ide ini menjadi aplikasi Internet yang sebenarnya? Karena aplikasi berjalan di sistem akhir, Anda perlu menulis program yang berjalan di sistem akhir. Anda mungkin, misalnya, menulis program Anda di Java, C, atau Python. Sekarang, karena Anda sedang mengembangkan aplikasi Internet terdistribusi, program yang berjalan pada sistem akhir yang berbeda harus saling mengirim data. Dan di sini kita sampai pada masalah utama —yang mengarah pada cara alternatif untuk menggambarkan Internet sebagai platform untuk aplikasi. Bagaimana satu program yang berjalan di satu sistem ujung menginstruksikan Internet untuk mengirimkan data ke program lain yang berjalan di sistem ujung lainnya?

Sistem akhir yang terhubung ke Internet menyediakan **Antarmuka Pemrograman Aplikasi (API)** yang menentukan bagaimana program yang berjalan di satu sistem akhir meminta infrastruktur Internet untuk mengirimkan data ke program tujuan tertentu yang berjalan di sistem akhir lainnya. API Internet ini adalah seperangkat aturan yang harus diikuti oleh program pengiriman agar Internet dapat mengirimkan data ke program tujuan. Kita akan membahas Internet API secara detail di Bab 2. Untuk saat ini, mari kita gunakan analogi sederhana, yang akan sering kita gunakan dalam buku ini. Misalkan Alice ingin mengirim surat kepada Bob menggunakan jasa pos. Alice, tentu saja, tidak bisa begitu saja menulis surat (data) dan menjatuhkan surat itu ke luar jendelanya. Sebaliknya, layanan pos mengharuskan Alice memasukkan surat itu ke dalam amplop; tulis nama lengkap, alamat, dan kode pos Bob di tengah amplop; tutup amplop; beri cap di sudut kanan atas amplop; dan terakhir, masukkan amplop ke kotak surat layanan pos resmi. Dengan demikian, layanan pos memiliki "API layanan pos" sendiri, atau seperangkat aturan, yang harus diikuti Alice agar layanan pos mengirimkan suratnya kepada Bob. Dengan cara yang sama, Internet memiliki API yang harus diikuti oleh program pengiriman data agar Internet mengirimkan data ke program yang akan menerima data.

Layanan pos tentu saja menyediakan lebih dari satu layanan kepada pelanggannya. Ini menyediakan pengiriman ekspres, konfirmasi penerimaan, penggunaan biasa, dan banyak layanan lainnya. Dengan cara yang sama, Internet menyediakan banyak layanan untuk aplikasinya. Saat Anda mengembangkan aplikasi Internet, Anda juga harus memilih salah satu layanan Internet untuk aplikasi Anda. Kami akan menjelaskan layanan Internet di Bab 2.

Kami baru saja memberikan dua deskripsi tentang Internet; satu dalam hal komponen perangkat keras dan perangkat lunaknya, yang lain dalam hal infrastruktur untuk menyediakan layanan ke aplikasi terdistribusi. Tapi mungkin Anda masih bingung apa itu

Internet adalah. Apa itu pengalihan paket dan TCP/IP? Apa itu router? Jenis tautan komunikasi apa yang ada di Internet? Apa itu aplikasi terdistribusi?

Bagaimana pemanggang roti atau sensor cuaca dapat dipasang ke Internet? Jika Anda merasa sedikit kewalahan dengan semua ini sekarang, jangan khawatir—tujuan buku ini adalah untuk memperkenalkan Anda pada seluk-beluk Internet dan prinsip-prinsip yang mengatur bagaimana dan mengapa internet bekerja. Kami akan menjelaskan istilah dan pertanyaan penting ini di bagian dan bab berikut.

1.1.3 Apa Itu Protokol?

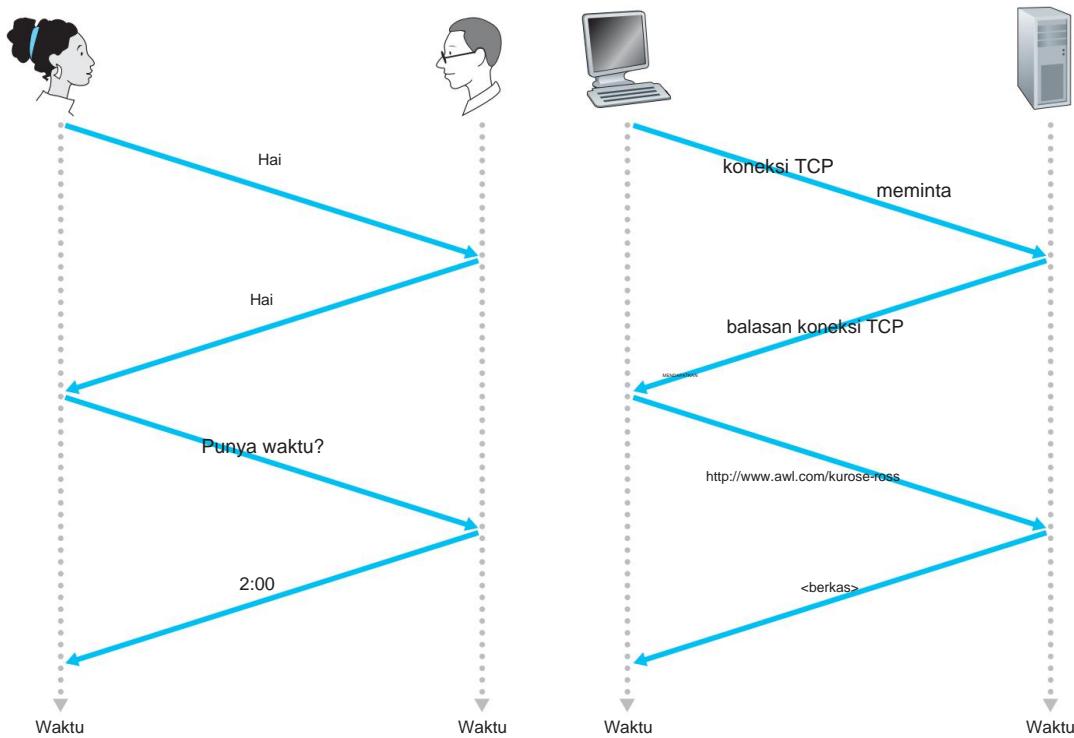
Sekarang setelah kita sedikit memahami apa itu Internet, mari pertimbangkan kata kunci penting lainnya dalam jaringan komputer: *protokol*. Apa itu protokol? Apa yang dilakukan protokol?

Analogi Manusia

Mungkin paling mudah untuk memahami pengertian protokol jaringan komputer dengan terlebih dahulu mempertimbangkan beberapa analogi manusia, karena kita manusia menjalankan protokol sepanjang waktu. Pertimbangkan apa yang Anda lakukan ketika Anda ingin menanyakan waktu kepada seseorang. Pertukaran tipikal ditunjukkan pada Gambar 1.2. Protokol manusia (atau setidaknya perilaku yang baik) menentukan bahwa seseorang pertama-tama menawarkan salam ("Hai" pertama pada Gambar 1.2) untuk memulai komunikasi dengan orang lain. Tanggapan khas untuk "Hai" adalah pesan "Hai" yang dikembalikan. Secara implisit, seseorang kemudian mengambil respons "Hai" yang ramah sebagai indikasi bahwa seseorang dapat melanjutkan dan menanyakan waktu hari itu. Tanggapan yang berbeda terhadap awalan "Hai" (seperti "Jangan ganggu saya!" atau "Saya tidak berbicara bahasa Inggris," atau balasan yang tidak dapat dicetak) mungkin menunjukkan keengganhan atau ketidakmampuan untuk berkomunikasi. Dalam hal ini, protokol manusia adalah tidak menanyakan waktu. Kadang-kadang seseorang tidak mendapat jawaban sama sekali untuk sebuah pertanyaan, dalam hal ini seseorang biasanya menyerah menanyakan waktu kepada orang itu. Perhatikan bahwa dalam protokol manusia kami, *ada pesan khusus yang kami kirim, dan tindakan khusus yang kami ambil sebagai tanggapan atas pesan balasan yang diterima atau kejadian lain* (seperti tidak ada balasan dalam jangka waktu tertentu). Jelas, pesan yang dikirim dan diterima, dan tindakan yang diambil saat pesan ini dikirim atau diterima atau peristiwa lain terjadi, memainkan peran sentral dalam protokol manusia. Jika orang menjalankan protokol yang berbeda (misalnya, jika satu orang memiliki tata krama tetapi yang lain tidak, atau jika yang satu memahami konsep waktu dan yang lain tidak) protokol tidak saling beroperasi dan tidak ada pekerjaan yang bermanfaat yang dapat diselesaikan. Hal yang sama berlaku dalam jaringan —dibutuhkan dua (atau lebih) entitas yang berkomunikasi yang menjalankan protokol yang sama untuk menyelesaikan tugas.

Mari pertimbangkan analogi manusia yang kedua. Misalkan Anda berada di kelas perguruan tinggi (kelas jaringan komputer, misalnya!). Guru mengoceh tentang protokol dan Anda bingung. Guru berhenti untuk bertanya, "Apakah ada pertanyaan?" (A

8 BAB 1 • JARINGAN KOMPUTER DAN INTERNET



Gambar 1.2 Protokol manusia dan protokol jaringan komputer

pesan yang ditransmisikan ke, dan diterima oleh, semua siswa yang tidak tidur). Anda mengangkat tangan (menyampaikan pesan implisit kepada guru). Guru Anda mengakui Anda dengan senyuman, mengatakan “Ya . . .” (sebuah pesan yang dijelaskan dalam bagian *Anggukan* ditanyai pertanyaan), dan Anda kemudian mengajukan pertanyaan Anda (yaitu, mengirimkan pesan Anda kepada guru Anda). Guru Anda mendengar pertanyaan Anda (menerima pesan pertanyaan Anda) dan jawaban (mengirim balasan kepada Anda). Sekali lagi, kami melihat bahwa pengiriman dan penerimaan pesan, dan serangkaian tindakan konvensional yang diambil saat pesan ini dikirim dan diterima, merupakan inti dari protokol tanya-jawab ini.

Protokol Jaringan

Protokol jaringan mirip dengan protokol manusia, kecuali bahwa entitas yang bertukar pesan dan mengambil tindakan adalah komponen perangkat keras atau perangkat lunak dari beberapa perangkat (misalnya, komputer, ponsel cerdas, tablet, router, atau perangkat berkemampuan jaringan lainnya).

perangkat). Semua aktivitas di Internet yang melibatkan dua atau lebih entitas jarak jauh yang berkomunikasi diatur oleh protokol. Misalnya, protokol yang diimplementasikan perangkat keras dalam dua komputer yang terhubung secara fisik mengontrol aliran bit pada "kawat" antara dua kartu antarmuka jaringan; protokol kontrol kongesti di sistem akhir mengontrol laju paket yang ditransmisikan antara pengirim dan penerima; protokol di router menentukan jalur paket dari sumber ke tujuan. Protokol berjalan di mana-mana di Internet, dan akibatnya banyak dari buku ini tentang protokol jaringan komputer.

Sebagai contoh protokol jaringan komputer yang mungkin sudah Anda kenal, pertimbangkan apa yang terjadi saat Anda membuat permintaan ke server Web, yaitu saat Anda mengetikkan URL halaman Web ke browser Web Anda. Skenario diilustrasikan di bagian kanan Gambar 1.2. Pertama, komputer Anda akan mengirimkan pesan permintaan koneksi ke server Web dan menunggu balasan. Server Web pada akhirnya akan menerima pesan permintaan koneksi Anda dan mengembalikan pesan balasan koneksi. Mengetahui bahwa sekarang OK untuk meminta dokumen Web, komputer Anda kemudian mengirimkan nama halaman Web yang ingin diambil dari server Web tersebut dalam pesan GET. Terakhir, server Web mengembalikan halaman Web (file) ke komputer Anda.

Mengingat contoh-contoh manusia dan jaringan di atas, pertukaran pesan dan tindakan yang diambil ketika pesan-pesan ini dikirim dan diterima adalah elemen penentu utama dari sebuah protokol:

*Protokol menentukan format dan urutan pesan yang dipertukarkan antara dua atau lebih **entitas** yang berkomunikasi, serta tindakan yang diambil pada pengiriman dan/atau penerimaan pesan atau kejadian lainnya.*

Internet, dan jaringan komputer pada umumnya, menggunakan protokol secara ekstensif. Protokol yang berbeda digunakan untuk menyelesaikan tugas komunikasi yang berbeda. Saat Anda membaca buku ini, Anda akan belajar bahwa beberapa protokol sederhana dan lugas, sementara yang lain rumit dan mendalam secara intelektual. Menguasai bidang jaringan komputer setara dengan memahami apa, mengapa, dan bagaimana protokol jaringan.

1.2 Tepi Jaringan

Pada bagian sebelumnya kami menyajikan ikhtisar tingkat tinggi dari Internet dan protokol kerja bersih. Kita sekarang akan mempelajari lebih dalam tentang komponen jaringan komputer (dan Internet, khususnya). Kita mulai di bagian ini di tepi jaringan dan melihat komponen yang paling kita kenal—yaitu, komputer, telepon pintar, dan perangkat lain yang kita gunakan sehari-hari. Pada bagian selanjutnya kita akan beralih dari tepi jaringan ke inti jaringan dan memeriksa peralihan dan perutean di jaringan komputer.

10 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****SEJARAH KASUS****JARINGAN SISTEM AKHIR INTERNET YANG Pusing**

Belum lama ini, perangkat sistem akhir yang terhubung ke Internet terutama adalah komputer tradisional seperti mesin desktop dan server yang kuat. Dimulai pada akhir 1990-an dan berlanjut hingga hari ini, berbagai perangkat menarik dihubungkan ke Internet, memanfaatkan kemampuannya untuk mengirim dan menerima data digital.

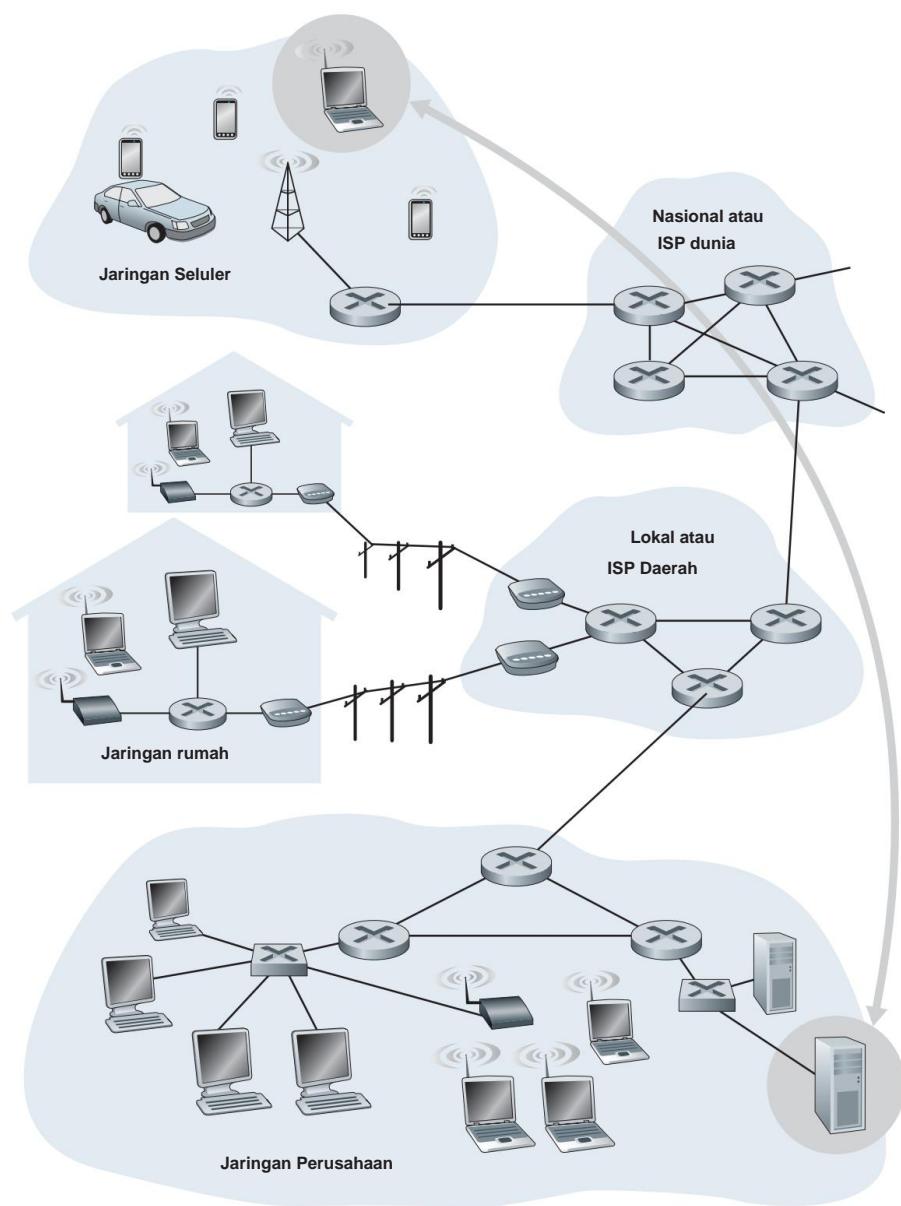
Mengingat Internet ada di mana-mana, protokolnya yang terdefinisi dengan baik (standar), dan ketersediaan perangkat keras komoditas yang siap untuk Internet, wajar menggunakan teknologi Internet untuk membuat jaringan perangkat ini bersama-sama dan ke server yang terhubung ke Internet.

Banyak dari perangkat ini berbasis di rumah—konsol permainan video (misal, Xbox Microsoft), televisi siap-Internet, bingkai foto digital yang mengunduh dan menampilkan gambar digital, mesin cuci, lemari es, dan bahkan pemanggang roti yang mengunduh informasi meteorologi dan membakar gambar bayangan hari ini (misalnya, campuran awan dan matahari) pada roti panggang pagi Anda [BBC 2001]. Ponsel berkemampuan IP dengan kemampuan GPS menempatkan layanan yang bergantung pada lokasi (peta, informasi tentang layanan atau orang terdekat) di ujung jari Anda. Sensor jaringan tertanam ke dalam lingkungan fisik memungkinkan pemantauan bangunan, jembatan, aktivitas seismik, habitat satwa liar, muara sungai, dan cuaca. Perangkat biomedis dapat disematkan dan dijejarkan dalam jaringan area tubuh. Dengan begitu banyak perangkat yang beragam yang digabungkan menjadi jaringan, Internet memang menjadi “Internet of things”

[ITU 2005b].

Ingin dari bagian sebelumnya bahwa dalam jargon jaringan komputer, komputer dan perangkat lain yang terhubung ke Internet sering disebut sebagai sistem akhir. Mereka disebut sebagai sistem akhir karena mereka duduk di tepi Internet, seperti yang ditunjukkan pada Gambar 1.3. Sistem akhir Internet termasuk komputer desktop (misalnya, PC desktop, Mac, dan kotak Linux), server (misalnya, server web dan email), dan komputer seluler (misalnya, laptop, smartphone, dan tablet). Selain itu, semakin banyak perangkat non-tradisional yang terhubung ke Internet sebagai sistem akhir (lihat sidebar).

Sistem akhir juga disebut sebagai *host* karena mereka menghosting (yaitu, menjalankan) program aplikasi seperti program browser Web, program server Web, program klien email, atau program server email. Sepanjang buku ini kita akan menggunakan istilah host dan sistem akhir secara bergantian; yaitu, *tuan rumah = sistem akhir*. Host terkadang dibagi lagi menjadi dua kategori: **klien** dan **server**. Secara informal, klien cenderung berupa PC desktop dan seluler, ponsel cerdas, dan sebagainya, sedangkan server cenderung merupakan mesin yang lebih kuat yang menyimpan dan mendistribusikan halaman Web, streaming video, menyampaikan email, dan sebagainya. Saat ini, sebagian besar server dari mana kami men-

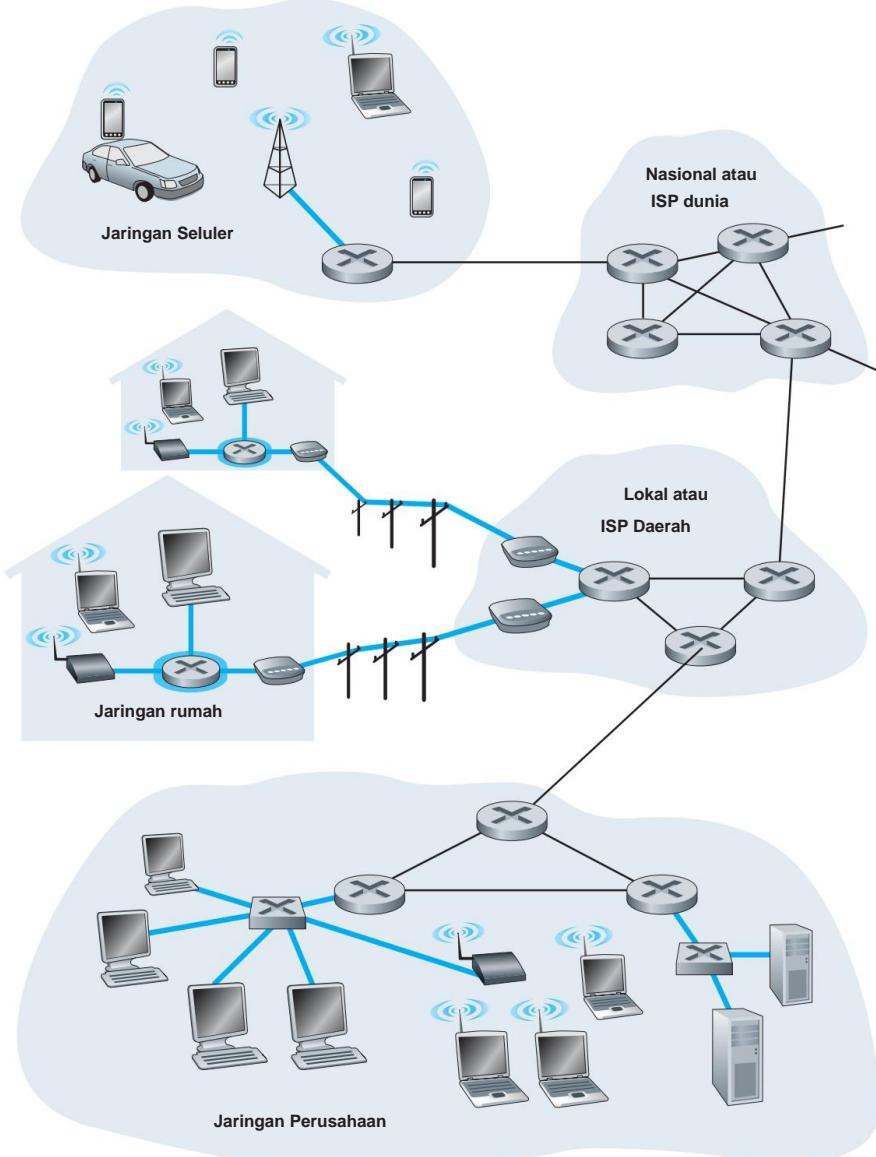


Gambar 1.3 Interaksi sistem akhir

hasil pencarian, email, halaman Web, dan video berada di **pusat data besar**. Misalnya, Google memiliki 30–50 pusat data, dengan banyak yang memiliki lebih dari seratus ribu server.

12 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****1.2.1 Jaringan Akses**

Setelah mempertimbangkan aplikasi dan sistem akhir di "tepi jaringan", selanjutnya mari kita pertimbangkan jaringan akses—jaringan yang secara fisik menghubungkan sistem akhir ke router pertama (juga dikenal sebagai "router tepi") di jalur dari sistem akhir ke sistem akhir jauh lainnya. Gambar 1.4 menunjukkan beberapa jenis akses

**Gambar 1.4** Jaringan akses

jaringan dengan garis tebal dan berbayang, dan pengaturan (rumah, perusahaan, dan nirkabel seluler area luas) di mana mereka digunakan.

Akses Rumah: DSL, Kabel, FTTH, Dial-Up, dan Satelit

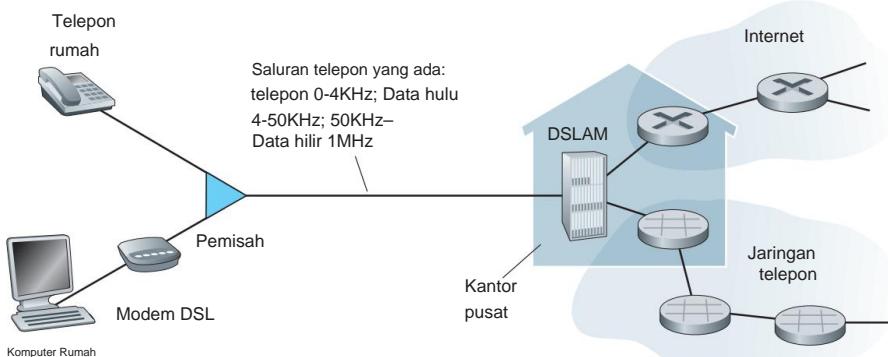
Di negara maju saat ini, lebih dari 65 persen rumah tangga memiliki akses Internet, dengan Korea, Belanda, Finlandia, dan Swedia memimpin dengan lebih dari 80 persen rumah tangga memiliki akses Internet, hampir semuanya melalui koneksi broadband berkecepatan tinggi [ITU 2011]. Finlandia dan Spanyol baru-baru ini mendeklarasikan akses Internet berkecepatan tinggi sebagai "hak legal". Mengingat minat yang kuat pada akses rumah, mari kita mulai ikhtisar jaringan akses dengan mempertimbangkan bagaimana rumah terhubung ke Internet.

Saat ini, dua jenis akses perumahan broadband yang paling umum adalah **digital subscriber line (DSL)** dan kabel. Tempat tinggal biasanya memperoleh akses Internet DSL dari perusahaan telepon lokal (telco) yang sama yang menyediakan akses telepon kabel lokalnya. Jadi, ketika DSL digunakan, perusahaan telekomunikasi pelanggan juga merupakan ISP-nya. Seperti yang ditunjukkan pada Gambar 1.5, setiap modem DSL pelanggan menggunakan saluran telepon yang ada (kabel tembaga twisted pair, yang akan kita bahas di Bagian 1.2.2) untuk bertukar data dengan digital subscriber line access multiplexer (DSLAM) yang terletak di jaringan telekomunikasi. kantor pusat setempat (CO). Modem DSL rumah mengambil data digital dan menerjemahkannya ke nada frekuensi tinggi untuk ditransmisikan melalui kabel telepon ke CO; sinyal analog dari banyak rumah seperti itu diterjemahkan kembali ke format digital di DSLAM.

Saluran telepon perumahan membawa data dan sinyal telepon tradisional secara bersamaan, yang dikodekan pada frekuensi yang berbeda:

- Saluran hilir berkecepatan tinggi, dalam pita 50 kHz hingga 1 MHz
- Saluran hulu berkecepatan menengah, dalam pita 4 kHz hingga Pita 50 kHz
- Saluran telepon dua arah biasa, dalam pita 0 hingga 4 kHz

Pendekatan ini membuat tautan DSL tunggal tampak seolah-olah ada tiga tautan terpisah, sehingga panggilan telepon dan koneksi Internet dapat berbagi tautan DSL pada saat yang bersamaan. (Kami akan menjelaskan teknik multiplexing pembagian frekuensi ini di



Gambar 1.5 Akses Internet DSL

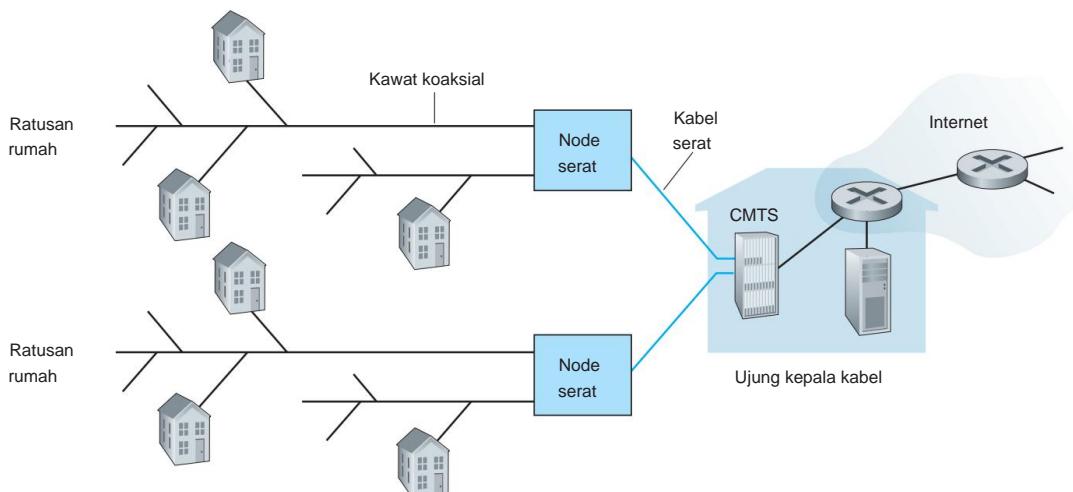
14 BAB 1

• JARINGAN KOMPUTER DAN INTERNET

Bagian 1.3.1). Di sisi pelanggan, sebuah splitter memisahkan sinyal data dan telepon yang tiba di rumah dan meneruskan sinyal data ke modem DSL. Di sisi telekomunikasi, di CO, DSLAM memisahkan data dan sinyal telepon dan mengirimkan data ke Internet. Ratusan atau bahkan ribuan rumah tangga terhubung ke satu DSLAM [Dischinger 2007].

Standar DSL menentukan tingkat transmisi downstream 12 Mbps dan upstream 1,8 Mbps [ITU 1999], dan downstream 24 Mbps dan upstream 2,5 Mbps [ITU 2003]. Karena laju hilir dan hulu berbeda, maka aksesnya dikatakan asimetris. Tingkat transmisi hilir dan hulu aktual yang dicapai mungkin kurang dari tarif yang disebutkan di atas, karena penyedia DSL dapat membatasi sepenuhnya tarif residensial ketika layanan berjenjang (tarif berbeda, tersedia dengan harga berbeda) ditawarkan, atau karena tarif maksimum dapat dibatasi oleh jarak antara rumah dan CO, ukuran garis pasangan bengkok dan tingkat gangguan listrik. Insinyur telah secara tegas merancang DSL untuk jarak pendek antara rumah dan CO; umumnya, jika tempat tinggal tidak terletak dalam jarak 5 sampai 10 mil dari CO, tempat tinggal tersebut harus menggunakan bentuk akses Internet alternatif.

Sementara DSL memanfaatkan infrastruktur telefon lokal telco yang sudah ada, **akses Internet kabel** memanfaatkan infrastruktur televisi kabel perusahaan televisi kabel yang sudah ada. Tempat tinggal memperoleh akses Internet kabel dari perusahaan yang sama yang menyediakan televisi kabelnya. Seperti yang diilustrasikan pada Gambar 1.6, serat optik menghubungkan ujung kepala kabel ke persimpangan tingkat lingkungan, dari mana kabel koaksial tradisional kemudian digunakan untuk menjangkau rumah dan apartemen individu. Setiap persimpangan lingkungan biasanya mendukung 500 hingga 5.000 rumah. Karena kedua kabel serat dan koaksial digunakan dalam sistem ini, sering disebut sebagai coax serat hibrida (HFC).



Gambar 1.6 Jaringan akses serat-koaksial hybrid

Akses internet kabel memerlukan modem khusus yang disebut modem kabel. Seperti modem DSL, modem kabel biasanya merupakan perangkat eksternal dan terhubung ke PC rumahan melalui port Ethernet. (Kita akan membahas Ethernet dengan sangat rinci di Bab 5.) Di ujung kepala kabel, sistem terminasi modem kabel (CMTS) memiliki fungsi yang sama dengan DSLAM jaringan DSL—mengubah sinyal analog yang dikirim dari modem kabel di banyak rumah hilir kembali ke format digital.

Modem kabel membagi jaringan HFC menjadi dua saluran, saluran hilir dan saluran hulu. Seperti halnya DSL, akses biasanya asimetris, dengan saluran down stream biasanya mengalokasikan tingkat transmisi yang lebih tinggi daripada saluran upstream. Standar DOCSIS 2.0 menetapkan kecepatan downstream hingga 42,8 Mbps dan kecepatan upstream hingga 30,7 Mbps. Seperti dalam kasus jaringan DSL, kecepatan maksimum yang dapat dicapai mungkin tidak tercapai karena kecepatan data kontrak yang lebih rendah atau gangguan media.

Salah satu karakteristik penting dari akses Internet kabel adalah media penyiaran bersama. Secara khusus, setiap paket yang dikirim oleh ujung kepala mengalir ke bawah pada setiap tautan ke setiap rumah dan setiap paket yang dikirim oleh sebuah rumah berjalan di saluran hulu ke ujung kepala. Untuk alasan ini, jika beberapa pengguna mengunduh file video secara bersamaan di saluran downstream, kecepatan sebenarnya di mana setiap pengguna menerima file videonya akan jauh lebih rendah daripada kecepatan downstream kabel agregat. Di sisi lain, jika hanya ada sedikit pengguna aktif dan mereka semua menjelajahi Web, maka masing-masing pengguna mungkin benar-benar menerima halaman Web dengan kecepatan downstream kabel penuh, karena pengguna jarang meminta halaman Web pada waktu yang persis sama. waktu. Karena saluran upstream juga dibagi, protokol akses ganda terdistribusi diperlukan untuk mengoordinasikan transmisi dan menghindari tabrakan.

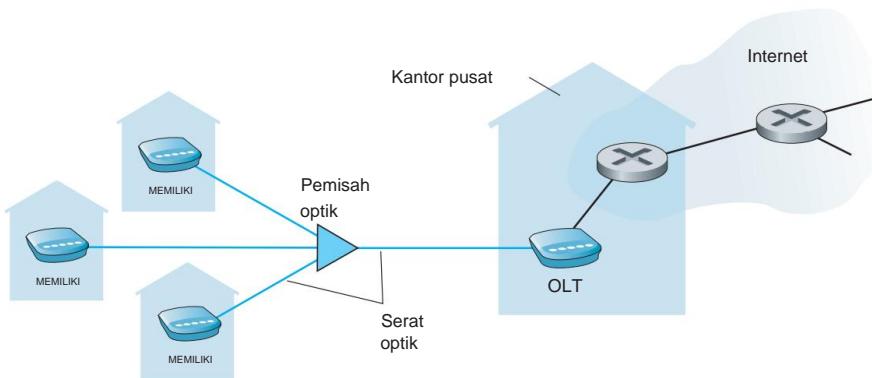
(Kita akan membahas masalah tabrakan ini secara mendetail di Bab 5.)

Meskipun jaringan DSL dan kabel saat ini mewakili lebih dari 90 persen akses broadband perumahan di Amerika Serikat, teknologi yang sedang naik daun yang menjanjikan kecepatan lebih tinggi adalah penyebaran serat **ke rumah (FTTH)**.

[Dewan FTTH 2011a]. Seperti namanya, konsep FTTH sederhana—menyediakan jalur serat optik dari CO langsung ke rumah. Di Amerika Serikat, Verizon sangat agresif dengan FTTH dengan layanan FIOS-nya [Verizon FIOS 2012].

Ada beberapa teknologi bersaing untuk distribusi optik dari CO ke rumah. Jaringan distribusi optik paling sederhana disebut serat langsung, dengan satu serat meninggalkan CO2 untuk setiap rumah. Lebih umum, setiap serat yang meninggalkan kantor pusat sebenarnya digunakan bersama oleh banyak rumah; baru setelah serat tersebut relatif dekat dengan rumah, serat tersebut dipecah menjadi serat khusus pelanggan individu. Ada dua arsitektur jaringan distribusi optik bersaing yang melakukan pemisahan ini: jaringan optik aktif (AON) dan jaringan optik pasif (PON). AON pada dasarnya adalah Ethernet yang diaktifkan, yang dibahas di Bab 5.

Di sini, kami membahas secara singkat PON, yang digunakan dalam layanan FIOS Verizon. Gambar 1.7 menunjukkan FTTH menggunakan arsitektur distribusi PON. Setiap rumah memiliki

16 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****Gambar 1.7** Akses Internet FTTH

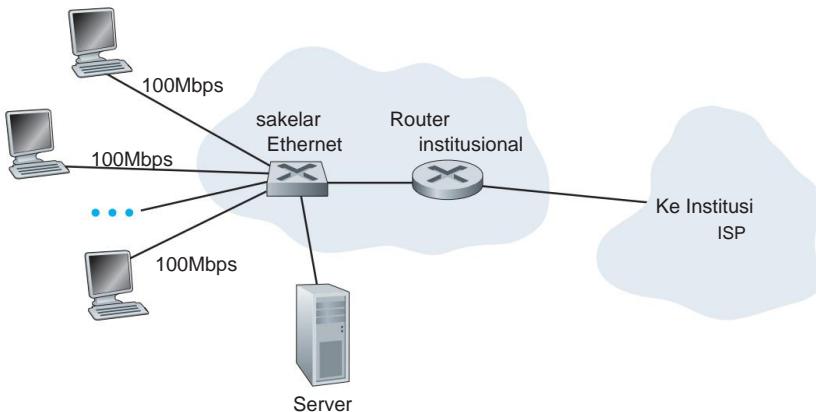
terminator jaringan optik (ONT), yang dihubungkan dengan serat optik khusus ke pembagi lingkungan. Pemisah menggabungkan sejumlah rumah (biasanya kurang dari 100) menjadi satu, serat optik bersama, yang terhubung ke terminator jalur optik (OLT) di CO perusahaan telekomunikasi. OLT, menyediakan konversi antara sinyal optik dan listrik, terhubung ke Internet melalui router telco. Di rumah, pengguna menghubungkan router rumah (biasanya router nirkabel) ke ONT dan mengakses Inter melalui router rumah ini. Dalam arsitektur PON, semua paket yang dikirim dari OLT ke splitter direplikasi di splitter (mirip dengan ujung kepala kabel).

FTTH berpotensi memberikan tarif akses Internet dalam kisaran gigabit per detik. Namun, sebagian besar ISP FTTH memberikan penawaran tarif yang berbeda, dengan tarif yang lebih tinggi tentu saja membutuhkan lebih banyak uang. Kecepatan hilir rata-rata pelanggan FTTH AS adalah sekitar 20 Mbps pada tahun 2011 (dibandingkan dengan 13 Mbps untuk jaringan akses kabel dan kurang dari 5 Mbps untuk DSL) [Dewan FTTH 2011b].

Dua teknologi jaringan akses lainnya juga digunakan untuk menyediakan akses Internet ke rumah. Di lokasi di mana DSL, kabel, dan FTTH tidak tersedia (misalnya, di beberapa pengaturan pedesaan), tautan satelit dapat digunakan untuk menghubungkan tempat tinggal ke jaringan Internet dengan kecepatan lebih dari 1 Mbps; StarBand dan HughesNet adalah dua penyedia akses satelit tersebut. Akses dial-up melalui saluran telepon tradisional didasarkan pada model yang sama seperti DSL—modem rumah terhubung melalui saluran telepon ke modem di ISP. Dibandingkan dengan DSL dan jaringan akses broadband lainnya, akses dial-up sangat lambat pada 56 kbps.

Akses di Perusahaan (dan Rumah): Ethernet dan WiFi

Di kampus perusahaan dan universitas, dan semakin banyak di pengaturan rumah, jaringan area lokal (LAN) digunakan untuk menghubungkan sistem akhir ke router tepi. Meskipun ada banyak jenis teknologi LAN, Ethernet sejauh ini merupakan teknologi akses yang paling lazim di jaringan perusahaan, universitas, dan rumah. Seperti yang ditunjukkan pada Gambar 1.8, pengguna Ethernet menggunakan kabel tembaga twisted-pair untuk terhubung ke switch Ethernet, a

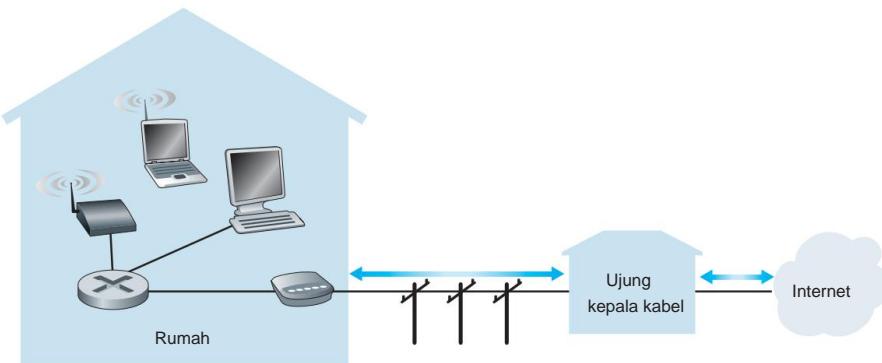


Gambar 1.8 Akses Internet Ethernet

teknologi dibahas secara rinci di Bab 5. Switch Ethernet, atau jaringan dari switch yang saling terhubung, kemudian terhubung ke Internet yang lebih besar. Dengan akses Ethernet, pengguna biasanya memiliki akses 100 Mbps ke sakelar Ethernet, sedangkan server mungkin memiliki akses 1 Gbps atau bahkan 10 Gbps.

Namun, semakin banyak orang yang mengakses Internet secara nirkabel dari laptop, smartphone, tablet, dan perangkat lain (lihat sidebar sebelumnya di “A Array of Devices yang Memusingkan”). Dalam pengaturan LAN nirkabel, pengguna nirkabel mengirim/menerima paket ke/dari titik akses yang terhubung ke jaringan perusahaan (kemungkinan besar termasuk Ethernet kabel), yang pada gilirannya terhubung ke Internet kabel. Pengguna LAN nirkabel biasanya harus berada dalam jarak beberapa puluh meter dari titik akses. Akses LAN nirkabel berdasarkan teknologi IEEE 802.11, yang lebih dikenal dengan WiFi, kini ada di mana-mana—universitas, kantor bisnis, kafe, pelabuhan udara, rumah, dan bahkan di pesawat terbang. Di banyak kota, seseorang dapat berdiri di sudut jalan dan berada dalam jangkauan sepuluh atau dua puluh stasiun pangkalan (untuk peta global stasiun pangkalan 802.11 yang dapat dijelajahi yang telah ditemukan dan dicatat di situs Web oleh orang-orang yang sangat senang melakukannya, hal, lihat [wigle.net 2012]). Seperti dibahas secara rinci di Bab 6, 802.11 saat ini menyediakan laju transmisi bersama hingga 54 Mbps.

Meskipun jaringan akses Ethernet dan WiFi awalnya digunakan dalam pengaturan hadiah (perusahaan, universitas), mereka baru-baru ini menjadi komponen jaringan rumah yang relatif umum. Banyak rumah menggabungkan akses perumahan broadband (yaitu, modem kabel atau DSL) dengan teknologi LAN nirkabel murah ini untuk membuat jaringan rumah yang kuat [Edwards 2011]. Gambar 1.9 menunjukkan jaringan rumah tipikal. Jaringan rumah ini terdiri dari laptop roaming serta PC berkabel; stasiun pangkalan (titik akses nirkabel), yang berkomunikasi dengan PC nirkabel; modem kabel, menyediakan akses broadband ke Internet; dan router, yang menghubungkan antara stasiun pangkalan dan PC stasioner dengan modem kabel. Jaringan ini memungkinkan anggota rumah tangga memiliki akses broadband ke Internet dengan satu anggota menjelajah dari dapur ke halaman belakang hingga ke kamar tidur.

18 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

Gambar 1.9 Sebuah jaringan rumah biasa

Akses Nirkabel Area Luas: 3G dan LTE

Perangkat seperti iPhone, BlackBerry, dan perangkat Android semakin banyak digunakan untuk mengirim email, menjelajahi Web, Tweet, dan mengunduh musik saat dalam perjalanan.

Perangkat ini menggunakan infrastruktur nirkabel yang sama yang digunakan untuk telepon seluler untuk mengirim/menerima paket melalui stasiun pangkalan yang dioperasikan oleh penyedia jaringan seluler. Tidak seperti WiFi, pengguna hanya perlu berada dalam jarak beberapa puluh kilometer (berlawanan dengan beberapa puluh meter) dari stasiun pangkalan.

Perusahaan telekomunikasi telah melakukan investasi besar dalam apa yang disebut nirkabel generasi ketiga (3G), yang menyediakan akses Internet nirkabel area luas packet-switched dengan kecepatan lebih dari 1 Mbps. Tetapi bahkan teknologi akses area luas berkecepatan lebih tinggi—jaringan nirkabel area luas generasi keempat (4G)—sudah digunakan. LTE (untuk “Long-Term Evolution”—kandidat untuk Bad Acronym of the Year Award) berakar pada teknologi 3G, dan berpotensi mencapai kecepatan lebih dari 10 Mbps. Tingkat downstream LTE puluhan Mbps telah dilaporkan dalam penerapan komersial. Kami akan membahas prinsip dasar jaringan nirkabel dan mobilitas, serta teknologi WiFi, 3G, dan LTE (dan banyak lagi!) di Bab 6.

1.2.2 Media Fisik

Pada subbagian sebelumnya, kami memberikan ikhtisar tentang beberapa teknologi akses jaringan yang paling penting di Internet. Saat kami menjelaskan teknologi ini, kami juga menunjukkan media fisik yang digunakan. Misalnya, kami mengatakan bahwa HFC menggunakan kombinasi kabel serat dan kabel koaksial. Kami mengatakan bahwa DSL dan Ethernet menggunakan kabel tembaga. Dan kami mengatakan bahwa jaringan akses seluler menggunakan spektrum radio.

Pada subbagian ini kami memberikan gambaran singkat tentang ini dan media transmisi lainnya yang umum digunakan di Internet.

Untuk mendefinisikan apa yang dimaksud dengan media fisik, mari kita renungkan sedikit tentang kehidupan singkat. Pertimbangkan sedikit perjalanan dari satu sistem akhir, melalui serangkaian tautan dan router, ke sistem akhir lainnya. Bagian yang malang ini ditendang dan ditransmisikan berkali-kali! Sistem ujung sumber pertama-tama mentransmisikan bit, dan tak lama kemudian router pertama dalam rangkaian menerima bit; router pertama kemudian mentransmisikan bit, dan tak lama kemudian router kedua menerima bit; dan seterusnya. Jadi bit kita, saat berjalan dari sumber ke tujuan, melewati serangkaian pasangan pemancar-penerima. Untuk setiap pasangan pemancar-penerima, bit dikirim dengan menyebarkan gelombang elektromagnetik atau pulsa optik melintasi **media fisik**.

Media fisik dapat mengambil banyak bentuk dan bentuk dan tidak harus dari jenis yang sama untuk setiap pasangan pemancar-penerima di sepanjang jalur. Contoh media fisik termasuk kabel tembaga twisted-pair, kabel koaksial, kabel serat optik multimode, spektrum radio terestrial, dan spektrum radio satelit. Media fisik dibagi menjadi dua kategori: **media terpandu** dan **media tidak terarah**. Dengan media terpandu, gelombang dipandu sepanjang media padat, seperti kabel serat optik, kabel tembaga twisted-pair, atau kabel koaksial. Dengan media yang tidak terarah, gelombang merambat di atmosfer dan luar angkasa, seperti di LAN nirkabel atau saluran satelit digital.

Namun sebelum kita membahas karakteristik dari berbagai jenis media, mari kita sampaikan beberapa patah kata tentang biayanya. Biaya sebenarnya dari sambungan fisik (kawat tembaga, kabel serat optik, dan sebagainya) seringkali relatif kecil dibandingkan dengan biaya jaringan lainnya. Secara khusus, biaya tenaga kerja yang terkait dengan pemasangan tautan fisik dapat jauh lebih tinggi daripada biaya material. Untuk itu, banyak pembangun yang memasang kabel twisted pair, serat optik, dan kabel coaxial di setiap ruangan dalam sebuah gedung. Bahkan jika hanya satu media yang digunakan pada awalnya, ada peluang bagus bahwa media lain dapat digunakan dalam waktu dekat, sehingga uang dapat dihemat dengan tidak perlu memasang kabel tambahan di masa mendatang.

Kawat Tembaga Twisted-Pair

Media transmisi terpandu yang paling murah dan paling umum digunakan adalah kawat tembaga pasangan bengkok. Selama lebih dari seratus tahun telah digunakan oleh jaringan telepon. Nyatanya, lebih dari 99 persen sambungan kabel dari handset telepon ke saklar telepon lokal menggunakan kabel tembaga berpilin. Sebagian besar dari kita telah melihat twisted pair di rumah dan lingkungan kerja kita. Twisted pair terdiri dari dua kabel tembaga berinsulasi, masing-masing setebal 1 mm, disusun dalam pola spiral yang teratur. Kabel dipelintir bersama untuk mengurangi gangguan listrik dari pasangan serupa di dekatnya. Biasanya, sejumlah pasangan dibundel bersama dalam kabel dengan membungkus pasangan dalam pelindung pelindung. Sepasang kabel merupakan satu tautan komunikasi. **Unshielded twisted pair (UTP)** biasanya digunakan untuk

20 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

jaringan komputer di dalam gedung, yaitu untuk LAN. Tarif data untuk LAN yang menggunakan twisted pair saat ini berkisar dari 10 Mbps hingga 10 Gbps. Kecepatan data yang dapat dicapai bergantung pada ketebalan kabel dan jarak antara pemancar dan penerima.

Ketika teknologi serat optik muncul pada 1980-an, banyak orang meremehkan twisted pair karena kecepatan bitnya yang relatif rendah. Beberapa orang bahkan merasa bahwa teknologi fiber optic akan sepenuhnya menggantikan twisted pair. Tapi twisted pair tidak menyerah begitu saja. Teknologi twisted-pair modern, seperti kabel kategori 6a, dapat mencapai kecepatan data 10 Gbps untuk jarak hingga seratus meter. Pada akhirnya, twisted pair telah muncul sebagai solusi dominan untuk jaringan LAN berkecepatan tinggi.

Seperti yang telah dibahas sebelumnya, twisted pair juga biasa digunakan untuk akses internet perumahan. Kami melihat bahwa teknologi modem dial-up memungkinkan akses dengan kecepatan hingga 56 kbps melalui twisted pair. Kami juga melihat bahwa teknologi DSL (digital subscriber line) telah memungkinkan pengguna perumahan untuk mengakses Internet dengan kecepatan puluhan Mbps melalui twisted pair (ketika pengguna tinggal dekat dengan modem ISP).

Kawat koaksial

Seperti pasangan bengkok, kabel koaksial terdiri dari dua konduktor tembagga, tetapi kedua konduktor itu konsentris dan bukan paralel. Dengan konstruksi dan insulasi dan pelindung khusus ini, kabel koaksial dapat mencapai tingkat transmisi data yang tinggi. Kabel koaksial cukup umum dalam sistem televisi kabel. Seperti yang kita lihat sebelumnya, sistem televisi kabel baru-baru ini digabungkan dengan modem kabel untuk memberikan pengguna rumahan akses Internet dengan kecepatan puluhan Mbps. Di televisi kabel dan akses Internet kabel, pemancar menggeser sinyal digital ke pita frekuensi tertentu, dan sinyal analog yang dihasilkan dikirim dari pemancar ke satu atau lebih penerima.

Kabel koaksial dapat digunakan sebagai **media bersama yang dipandu**. Khususnya, sejumlah sistem ujung dapat dihubungkan langsung ke kabel, dengan masing-masing sistem ujung menerima apapun yang dikirim oleh sistem ujung lainnya.

Serat optik

Serat optik adalah media tipis dan fleksibel yang mengantarkan gelombang cahaya, dengan masing-masing gelombang mewakili sedikit. Serat optik tunggal dapat mendukung kecepatan bit yang luar biasa, hingga puluhan atau bahkan ratusan gigabit per detik. Mereka kebal terhadap interferensi elektromagnetik, memiliki pelemahan sinyal yang sangat rendah hingga 100 kilometer, dan sangat sulit untuk disadap. Karakteristik ini menjadikan serat optik sebagai media transmisi berpemandu jarak jauh yang lebih disukai, terutama untuk sambungan luar negeri. Banyak jaringan telepon jarak jauh di Amerika Serikat dan di tempat lain sekarang menggunakan serat optik secara eksklusif. Serat optik juga lazim di tulang punggung Internet. Namun, tingginya biaya perangkat optik—seperti pemancar, penerima, dan sakelar—telah menghambat penerapannya untuk transportasi jarak pendek, seperti di LAN atau ke jaringan.

rumah di jaringan akses perumahan. Kecepatan link standar Optical Carrier (OC) berkisar dari 51,8 Mbps hingga 39,8 Gbps; spesifikasi ini sering disebut sebagai OC n, di mana kecepatan link sama dengan $n \times 51,8$ Mbps. Standar yang digunakan saat ini termasuk OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192, OC-768. [Mukherjee 2006, Ramaswamy 2010] memberikan cakupan berbagai aspek jaringan optik.

Saluran Radio Terestrial

Saluran radio membawa sinyal dalam spektrum elektromagnetik. Mereka adalah media yang menarik karena tidak memerlukan kabel fisik untuk dipasang, dapat menembus dinding, menyediakan konektivitas ke pengguna seluler, dan berpotensi membawa sinyal untuk jarak jauh. Karakteristik saluran radio sangat bergantung pada lingkungan propagasi dan jarak yang ditempuh sinyal. Pertimbangan lingkungan menentukan path loss dan shadow fading (yang menurunkan kekuatan sinyal saat sinyal bergerak dalam jarak jauh dan mengelilingi/melalui objek yang menghalangi), multi path fading (karena pantulan sinyal dari objek yang mengganggu), dan interferensi (karena faktor lain). transmisi dan sinyal elektromagnetik).

Saluran radio terestrial dapat secara luas diklasifikasikan menjadi tiga kelompok: yang beroperasi pada jarak yang sangat pendek (misalnya, dengan satu atau dua meter); yang beroperasi di area lokal, biasanya berkisar dari sepuluh hingga beberapa ratus meter; dan yang beroperasi di wilayah yang luas, yang membentang puluhan kilometer. Perangkat pribadi seperti headset tanpa kabel, keyboard, dan perangkat medis beroperasi dalam jarak pendek; teknologi LAN nirkabel yang dijelaskan di Bagian 1.2.1 menggunakan saluran radio area lokal; teknologi akses seluler menggunakan saluran radio area luas. Kami akan membahas saluran radio secara rinci di Bab 6.

Saluran Radio Satelit

Satelit komunikasi menghubungkan dua atau lebih pemancar/penerima gelombang mikro berbasis Bumi, yang dikenal sebagai stasiun bumi. Satelit menerima transmisi pada satu pita frekuensi, meregenerasi sinyal menggunakan pengulang (dibahas di bawah), dan mentransmisikan sinyal pada frekuensi lain. Dua jenis satelit digunakan dalam komunikasi: **satelit geostasioner** dan **satelit orbit rendah (LEO)**.

Satelit geostasioner secara permanen tetap berada di atas titik yang sama di Bumi. Kehadiran stasiun ini dicapai dengan menempatkan satelit di orbit pada ketinggian 36.000 kilometer di atas permukaan bumi. Jarak yang sangat jauh dari stasiun bumi melalui satelit kembali ke stasiun bumi menyebabkan penundaan propagasi sinyal yang substansial sebesar 280 milidetik. Namun demikian, tautan satelit, yang dapat beroperasi dengan kecepatan ratusan Mbps, sering digunakan di area yang tidak memiliki akses DSL atau akses Internet berbasis kabel.

Satelit LEO ditempatkan lebih dekat ke Bumi dan tidak secara permanen berada di atas satu titik di Bumi. Mereka berputar mengelilingi Bumi (seperti halnya Bulan) dan dapat berkomunikasi satu sama lain, serta dengan stasiun bumi. Memberikan terus menerus

22 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

jangkauan ke suatu daerah, banyak satelit perlu ditempatkan di orbit. Saat ini banyak sistem komunikasi ketinggian rendah yang sedang dikembangkan. Halaman web konstelasi satelit Lloyd [Wood 2012] menyediakan dan mengumpulkan informasi tentang sistem konstelasi satelit untuk komunikasi. Teknologi satelit LEO mungkin akan digunakan untuk akses Internet di masa mendatang.

1.3 Inti Jaringan

Setelah mempelajari keunggulan Internet, mari kita mempelajari lebih dalam inti jaringan—jaringan switch paket dan tautan yang menghubungkan sistem akhir Internet. Gambar 1.10 menyoroti inti jaringan dengan garis tebal dan berbayang.

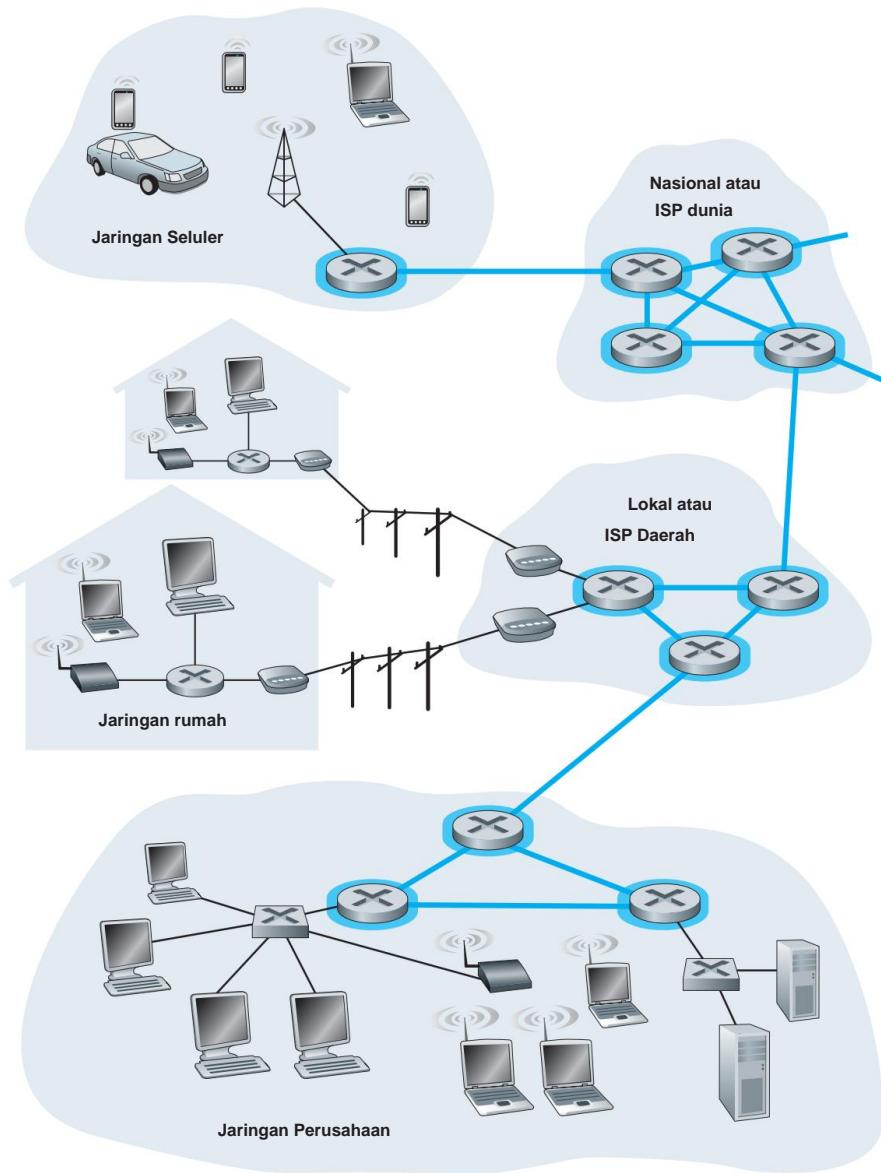
1.3.1 Perpindahan Paket

Dalam aplikasi jaringan, sistem akhir bertukar **pesan** satu sama lain. Mes sages dapat berisi apa pun yang diinginkan oleh perancang aplikasi. Pesan dapat melakukan fungsi kontrol (misalnya, pesan "Hai" dalam contoh jabat tangan kami di Gambar 1.2) atau dapat berisi data, seperti pesan email, gambar JPEG, atau file audio MP3. Untuk mengirim pesan dari sistem ujung sumber ke sistem akhir tujuan, sumber memecah pesan panjang menjadi potongan data yang lebih kecil yang dikenal sebagai **paket**.

Antara sumber dan tujuan, setiap paket berjalan melalui tautan komunikasi dan **sakelar paket** (yang ada dua jenis utama, **router** dan **sakelar lapisan tautan**). Paket ditransmisikan melalui setiap tautan komunikasi dengan laju yang sama dengan laju transmisi *penuh* dari tautan tersebut. Jadi, jika sebuah source end system atau sebuah packet switch mengirimkan sebuah paket L bits melalui sebuah link dengan laju transmisi R bits/sec, maka waktu untuk mentransmisikan paket tersebut adalah L/R detik.

Transmisi Store-and-Forward

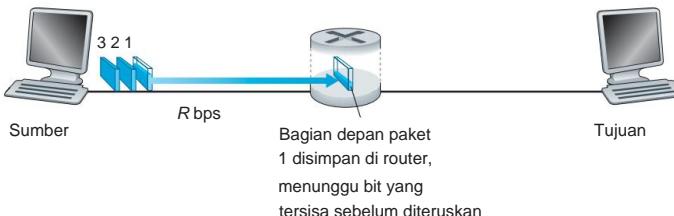
Sebagian besar sakelar paket menggunakan **transmisi simpan-dan-teruskan** pada input ke tautan. Transmisi store-and-forward berarti bahwa packet switch harus menerima seluruh paket sebelum dapat mulai mentransmisikan bit pertama dari paket ke link keluar. Untuk menjelajahi transmisi store-and-forward secara lebih rinci, pertimbangkan jaringan sederhana yang terdiri dari dua sistem akhir yang dihubungkan oleh satu router, seperti yang ditunjukkan pada Gambar 1.11. Sebuah router biasanya memiliki banyak tautan insiden, karena tugasnya adalah mengalihkan paket masuk ke tautan keluar; dalam contoh sederhana ini, router memiliki tugas yang agak sederhana untuk mentransfer paket dari satu tautan (input) ke satu-satunya tautan yang terpasang. Dalam contoh ini, sumber memiliki tiga paket, masing-masing terdiri dari L bit, untuk dikirimkan ke tujuan. Pada snapshot waktu yang ditunjukkan pada Gambar 1.11, sumber telah mengirimkan beberapa paket 1, dan bagian depan paket 1 sudah sampai di router. Karena router menggunakan store-and-forwarding, pada saat ini, router tidak dapat mengirimkan bit yang telah diterimanya; sebagai gantinya, router hanya mampu mentransmisikan bit yang belum diterima.



Gambar 1.10 Inti jaringan

24 BAB 1

• JARINGAN KOMPUTER DAN INTERNET



Gambar 1.11 Perpindahan paket store-and-forward

pertama-tama harus buffer (yaitu, "menyimpan") bit paket. Hanya setelah router menerima semua bit paket barulah router dapat mulai mentransmisikan (yaitu, "meneruskan") paket ke link keluar. Untuk mendapatkan beberapa wawasan tentang transmisi store-and-forward, sekarang mari kita hitung jumlah waktu yang berlalu sejak sumber mulai mengirim paket hingga tujuan menerima seluruh paket. (Di sini kita akan mengabaikan penundaan propagasi—waktu yang dibutuhkan bit untuk melintasi kabel dengan kecepatan mendekati kecepatan cahaya—yang akan dibahas di Bagian 1.4.) Sumber mulai mentransmisikan pada waktu 0; pada waktu L/R detik, sumber telah mengirimkan seluruh paket, dan seluruh paket telah diterima dan disimpan di router (karena tidak ada delay propagasi). Pada saat L/R detik, karena router baru saja menerima seluruh paket, router dapat mulai mengirimkan paket ke link keluar menuju negara tujuan; pada waktu $2L/R$, router telah mengirimkan seluruh paket, dan seluruh paket telah diterima oleh tujuan. Jadi, total delay adalah $2L/R$. Jika switch malah meneruskan bit segera setelah mereka tiba (tanpa terlebih dahulu menerima seluruh paket), maka total penundaan akan menjadi L/R karena bit tidak ditahan di router. Namun, seperti yang akan kita bahas di Bagian 1.4, router perlu menerima, menyimpan, dan memproses seluruh paket sebelum diteruskan.

Sekarang mari kita hitung jumlah waktu yang berlalu sejak sumber mulai mengirim paket pertama hingga tujuan menerima ketiga paket tersebut.

Seperi sebelumnya, pada saat L/R , router mulai meneruskan paket pertama. Tetapi juga pada saat L/R sumber akan mulai mengirim paket kedua, karena baru saja selesai mengirim seluruh paket pertama. Jadi, pada saat $2L/R$, tujuan telah menerima paket pertama dan router telah menerima paket kedua. Demikian pula, pada saat $3L/R$, tujuan telah menerima dua paket pertama dan router telah menerima paket ketiga. Akhirnya, pada saat $4L/R$ tujuan telah menerima ketiga paket!

Mari sekarang pertimbangkan kasus umum pengiriman satu paket dari sumber ke negara tujuan melalui jalur yang terdiri dari N tautan masing-masing dengan kecepatan R (dengan demikian, ada $N-1$ router antara sumber dan tujuan). Menerapkan logika yang sama seperti di atas, kita melihat bahwa penundaan end-to-end adalah:

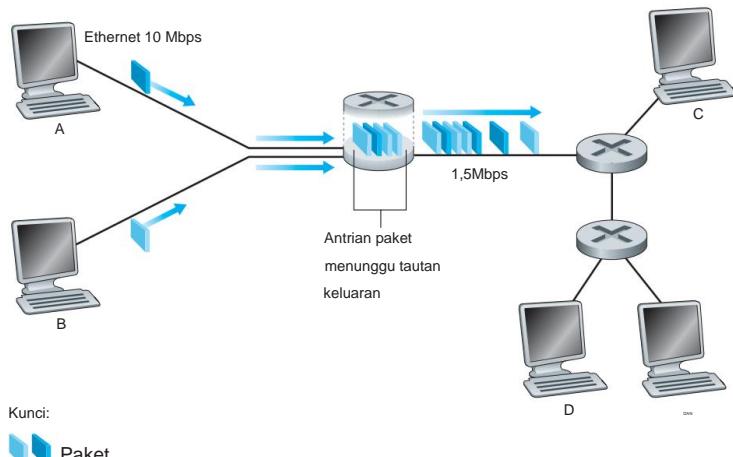
$$\text{dend@to@end} = \frac{L}{R} \quad (1.1)$$

Anda sekarang mungkin ingin mencoba untuk menentukan berapa penundaan untuk paket P yang dikirim melalui serangkaian tautan N .

Keterlambatan Antrian dan Kehilangan Paket

Setiap sakelar paket memiliki banyak tautan yang melekat padanya. Untuk setiap tautan yang terpasang, sakelar paket memiliki **penyangga keluaran** (juga disebut **antrean keluaran**), yang menyimpan paket yang akan dikirim oleh router ke tautan tersebut. Buffer output memainkan peran kunci dalam packet switching. Jika sebuah paket yang datang perlu ditransmisikan ke sebuah link tetapi ternyata link tersebut sibuk dengan transmisi paket lain, paket yang datang harus menunggu di buffer output. Jadi, selain penundaan store-and-forward, paket mengalami **penundaan antrian buffer keluaran**. Penundaan ini bervariasi dan bergantung pada tingkat kemacetan di jaringan. Karena jumlah ruang buffer terbatas, paket yang tiba mungkin menemukan bahwa buffer sudah penuh dengan paket lain yang menunggu untuk ditransmisikan. Dalam hal ini, **packet loss** akan terjadi—apakah paket yang datang atau salah satu paket yang sudah antri akan dibuang.

Gambar 1.12 mengilustrasikan jaringan packet-switched sederhana. Seperti pada Gambar 1.11, paket diwakili oleh lempengan tiga dimensi. Lebar slab mewakili jumlah bit dalam paket. Pada gambar ini, semua paket memiliki lebar yang sama dan karenanya memiliki panjang yang sama. Misalkan Host A dan B mengirim paket ke Host E. Host A dan B pertama-tama mengirim paket mereka melalui tautan Ethernet 10 Mbps ke router pertama. Router kemudian mengarahkan paket ini ke tautan 1,5 Mbps. Jika, selama selang waktu singkat, tingkat kedatangan paket ke router (ketika dikonversi ke bit per detik) melebihi 1,5 Mbps, kongesti akan terjadi di router saat paket mengantri di buffer keluaran link sebelum dikirim ke link. Misalnya, jika Host A dan B masing-masing mengirim semburan lima paket back-to-back pada waktu yang sama, maka sebagian besar paket ini akan menghabiskan waktu menunggu dalam antrian. Situasinya, sebenarnya, sepenuhnya analog dengan banyak situasi sehari-hari — misalnya, ketika kita mengantri ke teller bank atau menunggu di depan loket tol. Kami akan memeriksa penundaan antrian ini secara lebih rinci di Bagian 1.4.



Gambar 1.12 Perpindahan paket

26 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****Tabel Forwarding dan Protokol Routing**

Sebelumnya, kami mengatakan bahwa router mengambil paket yang tiba di salah satu tautan komunikasi yang terpasang dan meneruskan paket itu ke salah satu dari tautan komunikasi yang terpasang. Tapi bagaimana router menentukan tautan mana yang harus diteruskan paketnya? Penerusan paket sebenarnya dilakukan dengan cara yang berbeda di berbagai jenis jaringan komputer. Di sini, kami menjelaskan secara singkat bagaimana hal itu dilakukan di Internet.

Di Internet, setiap sistem akhir memiliki alamat yang disebut alamat IP. Ketika sistem akhir sumber ingin mengirim paket ke sistem akhir tujuan, sumber menyertakan alamat IP tujuan di header paket. Seperti halnya alamat pos, alamat ini memiliki struktur hierarkis. Ketika sebuah paket tiba di router di jaringan, router memeriksa sebagian dari alamat tujuan paket dan mengirimkan paket ke router yang berdekatan. Lebih khusus lagi, setiap router memiliki **tabel penerusan** yang memetakan alamat tujuan (atau bagian dari alamat tujuan) ke tautan keluar router tersebut. Ketika sebuah paket tiba di router, router memeriksa alamatnya dan mencari di tabel penerusannya, menggunakan alamat tujuan ini, untuk menemukan tautan keluar yang sesuai. Router kemudian mengarahkan paket ke tautan keluar ini.

Proses routing end-to-end dianalogikan sebagai pengemudi mobil yang tidak menggunakan peta tetapi lebih memilih untuk menanyakan arah. Misalnya, Joe mengemudi dari Philadelphia ke 156 Lakeside Drive di Orlando, Florida. Joe pertama-tama berkendara ke pom bensin di lingkungannya dan bertanya bagaimana menuju ke 156 Lakeside Drive di Orlando, Florida. Petugas pompa bensin mengekstrak bagian Florida dari alamat tersebut dan memberi tahu Joe bahwa dia harus pergi ke jalan raya antarnegara bagian I-95 Selatan, yang memiliki pintu masuk tepat di sebelah pompa bensin. Dia juga memberi tahu Joe bahwa begitu dia memasuki Florida, dia harus bertanya kepada orang lain di sana. Joe kemudian mengambil I-95 South sampai dia tiba di Jacksonville, Florida, di mana dia menanyakan arah kepada petugas pompa bensin lainnya. Petugas mengekstrak bagian Orlando dari alamat tersebut dan memberi tahu Joe bahwa dia harus melanjutkan I-95 ke Pantai Daytona dan kemudian bertanya kepada orang lain. Di Pantai Daytona, petugas pom bensin lainnya juga mengekstrak bagian Orlando dari alamat tersebut dan memberi tahu Joe bahwa dia harus membawa I-4 langsung ke Orlando. Joe mengambil I-4 dan turun di pintu keluar Orlando. Joe pergi ke petugas pompa bensin lain, dan kali ini petugas mengekstrak bagian Lakeside Drive dari alamat tersebut dan memberi tahu Joe jalan yang harus dia ikuti untuk sampai ke Lakeside Drive. Begitu Joe mencapai Lakeside Drive, dia bertanya kepada seorang anak yang sedang bersepeda bagaimana mencapai tujuannya. Anak itu mengekstrak 156 bagian dari alamat dan menunjuk ke rumah. Joe akhirnya mencapai tujuan akhirnya. Dalam analogi di atas, petugas pom bensin dan anak-anak bersepeda dianalogikan sebagai router.

Kami baru saja mengetahui bahwa router menggunakan alamat tujuan paket untuk mengindeks tabel warding dan menentukan tautan keluar yang sesuai. Namun pernyataan ini menimbulkan pertanyaan lain: Bagaimana tabel penerusan diatur? Apakah mereka dikonfigurasi oleh

menyerahkan setiap router, atau apakah Internet menggunakan prosedur yang lebih otomatis? Masalah ini akan dipelajari secara mendalam di Bab 4. Tetapi untuk membangkitkan selera Anda di sini, kami akan mencatat sekarang bahwa Internet memiliki sejumlah **protokol perutean** khusus yang digunakan untuk mengatur tabel penerusan secara otomatis. Protokol perutean dapat, misalnya, menentukan jalur terpendek dari setiap router ke setiap tujuan dan menggunakan hasil jalur terpendek untuk mengonfigurasi tabel penerusan di router.

Bagaimana Anda benar-benar ingin melihat rute end-to-end yang diambil paket di Internet? Kami sekarang mengundang Anda untuk mengotori tangan Anda dengan berinteraksi dengan program rute Trace. Cukup kunjungi situs www.traceroute.org, pilih sumber di negara tertentu, dan telusuri rute dari sumber tersebut ke komputer Anda. (Untuk pembahasan tentang Traceroute, lihat Bagian 1.4.)

1.3.2 Sakelar Sirkuit

Ada dua pendekatan mendasar untuk memindahkan data melalui jaringan link dan switch: **circuit switching** dan **packet switching**. Setelah membahas jaringan packet switched di subbagian sebelumnya, kita sekarang mengalihkan perhatian kita ke jaringan circuit switched.

Dalam jaringan circuit-switched, sumber daya yang dibutuhkan di sepanjang jalur (buffer, laju transmisi tautan) untuk menyediakan komunikasi antara sistem akhir dicadangkan *selama* durasi sesi komunikasi antara sistem akhir. Dalam jaringan packet-switched, sumber daya ini *tidak* dicadangkan; pesan sesi menggunakan sumber daya sesuai permintaan, dan sebagai konsekuensinya, mungkin harus menunggu (yaitu, antrean) untuk mengakses tautan komunikasi. Sebagai analogi sederhana, pertimbangkan dua restoran, satu yang memerlukan reservasi dan satu lagi yang tidak memerlukan reservasi atau menerimanya. Untuk restoran yang memerlukan reservasi, kami harus melalui kerumitan menelepon sebelum kami meninggalkan rumah. Tetapi ketika kita sampai di restoran, pada prinsipnya kita bisa langsung duduk dan memesan makanan kita. Untuk restoran yang tidak memerlukan reservasi, kita tidak perlu repot untuk memesan meja. Tetapi ketika kita tiba di restoran, kita mungkin harus menunggu meja sebelum kita dapat duduk.

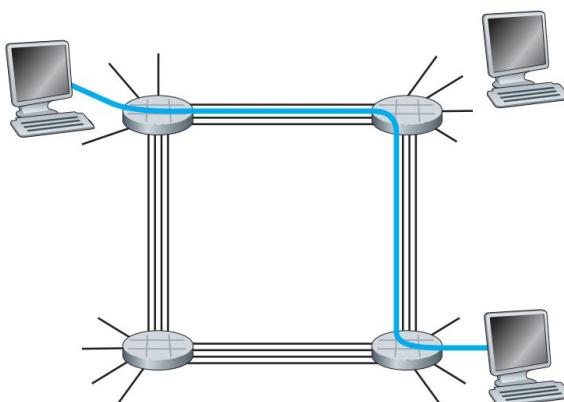
Jaringan telepon tradisional adalah contoh jaringan circuit-switched. Pertimbangkan apa yang terjadi ketika satu orang ingin mengirim informasi (suara atau facsimile) ke orang lain melalui jaringan telepon. Sebelum pengirim dapat mengirim informasi, jaringan harus membuat koneksi antara pengirim dan penerima. Ini adalah koneksi *bonafid* yang sakelar di jalur antara pengirim dan penerima mempertahankan status koneksi untuk koneksi itu. Dalam jargon telepon, hubungan ini disebut *sirkuit*. Ketika jaringan membuat sirkuit, ia juga mencadangkan laju transmisi konstan di tautan kerja jaringan (mewakili sebagian kecil dari kapasitas transmisi setiap tautan) selama durasi koneksi. Karena laju transmisi tertentu telah dicadangkan untuk koneksi pengirim ke penerima ini, pengirim dapat mentransfer data ke penerima dengan laju konstan *yang dijamin*.

Gambar 1.13 mengilustrasikan jaringan circuit-switched. Dalam jaringan ini, keempat sakelar sirkuit dihubungkan oleh empat tautan. Masing-masing sambungan ini memiliki empat sirkuit, sehingga setiap sambungan dapat mendukung empat sambungan secara bersamaan. Host (misalnya, PC dan workstation) masing-masing terhubung langsung ke salah satu sakelar. Ketika dua host ingin berkomunikasi, jaringan membuat **koneksi end-to-end** khusus antara dua host. Jadi, agar Host A dapat berkomunikasi dengan Host B, jaringan harus mencadangkan terlebih dahulu satu sirkuit pada masing-masing dua link. Dalam contoh ini, koneksi end-to-end khusus menggunakan sirkuit kedua pada tautan pertama dan sirkuit keempat pada tautan kedua. Karena setiap sambungan memiliki empat sirkuit, untuk setiap sambungan yang digunakan oleh sambungan ujung ke ujung, sambungan mendapatkan seperempat dari total kapasitas transmisi sambungan selama durasi sambungan. Jadi, misalnya, jika setiap tautan antara sakelar yang berdekatan memiliki laju transmisi 1 Mbps, maka setiap koneksi sakelar sirkuit end-to-end mendapatkan laju transmisi k

Sebaliknya, pertimbangkan apa yang terjadi ketika satu host ingin mengirim paket ke host lain melalui jaringan packet-switched, seperti Internet. Seperti halnya circuit switching, paket ditransmisikan melalui serangkaian link komunikasi. Namun berbeda dengan circuit switching, paket dikirim ke jaringan tanpa memesan sumber daya tautan apa pun. Jika salah satu link macet karena paket lain harus dikirim melalui link pada saat yang sama, maka paket harus menunggu di buffer di sisi pengirim dari link transmisi dan mengalami penundaan. Internet melakukan upaya terbaiknya untuk mengirimkan paket secara tepat waktu, tetapi tidak memberikan jaminan apa pun.

Multiplexing di Circuit-Switched Networks

Sirkuit dalam tautan diimplementasikan dengan **multiplexing pembagian frekuensi (FDM)** atau **multiplexing pembagian waktu (TDM)**. Dengan FDM, spektrum frekuensi dari suatu tautan dibagi di antara koneksi yang dibuat di seluruh tautan.



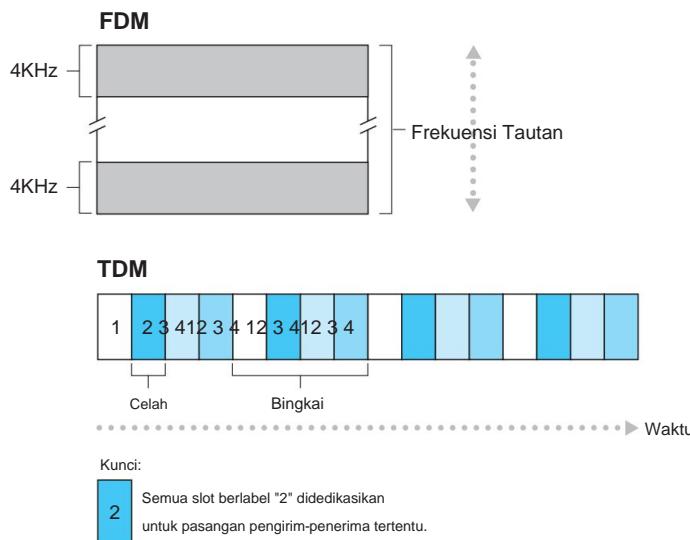
Gambar 1.13 Jaringan circuit-switched sederhana yang terdiri dari empat switch dan empat link

Khususnya, tautan mendedikasikan pita frekuensi untuk setiap sambungan selama sambungan berlangsung. Dalam jaringan telepon, pita frekuensi ini biasanya memiliki lebar 4 kHz (yaitu, 4.000 hertz atau 4.000 siklus per detik). Lebar pita disebut, tidak mengherankan, **bandwidth**. Stasiun radio FM juga menggunakan FDM untuk berbagi spektrum frekuensi antara 88 MHz dan 108 MHz, dengan masing-masing stasiun dialokasikan pita frekuensi tertentu.

Untuk link TDM, waktu dibagi menjadi frame dengan durasi tetap, dan setiap frame dibagi menjadi slot waktu dengan jumlah tetap. Ketika jaringan membuat koneksi melintasi sebuah link, jaringan mendedikasikan satu slot waktu di setiap frame untuk koneksi ini. Slot ini didedikasikan hanya untuk penggunaan koneksi tersebut, dengan satu slot waktu tersedia untuk digunakan (di setiap frame) untuk mengirimkan data koneksi.

Gambar 1.14 mengilustrasikan FDM dan TDM untuk tautan jaringan tertentu yang mendukung hingga empat sirkuit. Untuk FDM, domain frekuensi disegmentasi menjadi empat band, masing-masing bandwidth 4 kHz. Untuk TDM, domain waktu disegmentasi menjadi beberapa frame, dengan empat slot waktu di setiap frame; setiap sirkuit diberi slot khusus yang sama dalam bingkai TDM yang berputar. Untuk TDM, laju transmisi sirkuit sama dengan laju bingkai dikalikan dengan jumlah bit dalam slot. Misalnya, jika tautan mentransmisikan 8.000 frame per detik dan setiap slot terdiri dari 8 bit, maka laju transmisi sirkuit adalah 64 kbps.

Pendukung packet switching selalu berpendapat bahwa circuit switching boros karena sirkuit khusus menganggur selama **periode diam**. Misalnya,



Gambar 1.14 Dengan FDM, setiap rangkaian secara terus-menerus mendapatkan sebagian kecil dari bandwidth. Dengan TDM, setiap sirkuit mendapatkan semua bandwidth secara berkala selama interval waktu yang singkat (yaitu, selama slot)

30 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

ketika satu orang dalam panggilan telepon berhenti berbicara, sumber daya jaringan yang menganggur (pita frekuensi atau slot waktu di tautan di sepanjang rute koneksi) tidak dapat digunakan oleh koneksi lain yang sedang berlangsung. Sebagai contoh lain tentang bagaimana sumber daya ini dapat kurang dimanfaatkan, pertimbangkan ahli radiologi yang menggunakan jaringan circuit-switched untuk mengakses serangkaian sinar-x dari jarak jauh. Ahli radiologi mengatur koneksi, meminta gambar, merenungkan gambar, dan kemudian meminta gambar baru. Sumber daya jaringan dialokasikan untuk koneksi tetapi tidak digunakan (yaitu, terbuang percuma) selama periode kontemplasi radiolog. Pendukung packet switching juga senang menunjukkan bahwa membangun sirkuit end-to-end dan mencadangkan kapasitas transmisi end-to-end itu rumit dan memerlukan perangkat lunak pensinyalan yang rumit untuk mengoordinasikan pengoperasian sakelar di sepanjang jalur end-to-end.

Sebelum kita menyelesaikan diskusi kita tentang circuit switching, mari kita bekerja melalui contoh numerik yang akan memberikan wawasan lebih lanjut tentang topik tersebut. Mari kita pertimbangkan berapa lama untuk mengirim file 640.000 bit dari Host A ke Host B melalui jaringan circuit-switched. Misalkan semua link dalam jaringan menggunakan TDM dengan 24 slot dan memiliki bit rate 1,536 Mbps. Anggap juga dibutuhkan 500 msec untuk membuat sirkuit end-to-end sebelum Host A dapat mulai mengirimkan file. Berapa lama waktu yang dibutuhkan untuk mengirim file? Setiap sirkuit memiliki kecepatan transmisi $(1,536 \text{ Mbps})/24 = 64 \text{ kbps}$, sehingga dibutuhkan $(640.000 \text{ bit})/(64 \text{ kbps}) = 10 \text{ detik}$ untuk mengirimkan file. Untuk 10 detik ini kami menambahkan waktu pembuatan sirkuit, memberikan 10,5 detik untuk mengirim file. Perhatikan bahwa waktu transmisi tidak bergantung pada jumlah link: Waktu transmisi akan menjadi 10 detik jika sirkuit end-to-end melewati satu link atau seratus link. (Penundaan end-to-end aktual juga mencakup penundaan propagasi; lihat Bagian 1.4.)

Pengalihan Paket versus Pengalihan Sirkuit

Setelah menjelaskan switching sirkuit dan switching paket, mari kita bandingkan keduanya. Kritik terhadap packet switching sering berargumen bahwa packet switching tidak cocok untuk layanan real-time (misalnya, panggilan telepon dan panggilan konferensi video) karena penundaan end-to-end yang bervariasi dan tidak dapat diprediksi (terutama karena antrian yang bervariasi dan tidak dapat diprediksi). penundaan). Pendukung packet switching berpendapat bahwa (1) menawarkan pembagian kapasitas transmisi yang lebih baik daripada circuit switching dan (2) lebih sederhana, lebih efisien, dan lebih murah untuk diimplementasikan daripada circuit switching. Pembahasan yang menarik tentang packet switching versus circuit switching adalah [Molinero Fernandez 2002]. Secara umum, orang yang tidak suka repot dengan reservasi restoran lebih suka paket switching daripada circuit switching.

Mengapa packet switching lebih efisien? Mari kita lihat contoh sederhana. Misalkan pengguna berbagi tautan 1 Mbps. Anggap juga bahwa setiap pengguna bergantian antara periode aktivitas, saat pengguna menghasilkan data dengan laju konstan 100 kbps, dan periode tidak aktif, saat pengguna tidak menghasilkan data. Misalkan lebih lanjut bahwa seorang pengguna aktif hanya 10 persen dari waktu (dan iseng minum kopi selama 90 persen sisanya). Dengan pengalihan sirkuit, 100 kbps harus disediakan untuk *setiap* pengguna di

sepanjang waktu. Misalkan, dengan TDM circuit-switched, jika frame satu detik dibagi menjadi 10 slot waktu masing-masing 100 ms, maka setiap pengguna akan dialokasikan satu slot waktu per frame.

Dengan demikian, link circuit-switched hanya dapat mendukung 10 (= 1 Mbps/100 kbps) pengguna secara bersamaan. Dengan pengalihan paket, kemungkinan pengguna tertentu aktif adalah 0,1 (yaitu, 10 persen). Jika ada 35 pengguna, kemungkinan ada 11 atau lebih pengguna aktif secara bersamaan adalah sekitar 0,0004. (Soal Pekerjaan Rumah P8 menguraikan bagaimana probabilitas ini diperoleh.) Ketika ada 10 atau kurang pengguna yang aktif secara bersamaan (yang terjadi dengan probabilitas 0,9996), tingkat kedatangan agregat data kurang dari atau sama dengan 1 Mbps, tingkat output dari tautan. Jadi, ketika ada 10 atau lebih sedikit pengguna aktif, paket pengguna mengalir melalui tautan pada dasarnya tanpa penundaan, seperti halnya dengan pengalihan sirkuit. Ketika ada lebih dari 10 pengguna aktif secara bersamaan, maka tingkat kedatangan agregat paket melebihi kapasitas keluaran tautan, dan antrian keluaran akan mulai bertambah. (Hal ini terus tumbuh sampai tingkat input agregat turun kembali di bawah 1 Mbps, di mana antrian akan mulai berkurang panjangnya.) Karena kemungkinan memiliki lebih dari 10 pengguna aktif secara bersamaan sangat kecil dalam contoh ini, packet switching menyediakan pada dasarnya kinerja yang sama dengan switching sirkuit, tetapi melakukannya sambil memungkinkan lebih dari tiga kali jumlah pengguna.

Sekarang mari kita perhatikan contoh sederhana kedua. Misalkan ada 10 pengguna dan satu pengguna itu tiba-tiba menghasilkan seribu paket 1.000-bit, sementara pengguna lain diam dan tidak menghasilkan paket. Di bawah pengalihan sirkuit TDM dengan 10 slot per frame dan setiap slot terdiri dari 1.000 bit, pengguna aktif hanya dapat menggunakan satu slot waktu per frame untuk mengirimkan data, sedangkan sembilan slot waktu yang tersisa di setiap frame tetap tidak digunakan. Ini akan menjadi 10 detik sebelum semua satu juta bit data pengguna aktif telah dikirim. Dalam kasus pengalihan paket, pengguna aktif dapat terus mengirim paketnya dengan kecepatan tautan penuh 1 Mbps, karena tidak ada pengguna lain yang menghasilkan paket yang perlu dimultipleks dengan paket pengguna aktif. Dalam hal ini, semua data pengguna aktif akan dikirim dalam waktu 1 detik.

Contoh di atas mengilustrasikan dua cara di mana kinerja packet switching bisa lebih baik daripada circuit switching. Mereka juga menyoroti perbedaan krusial antara dua bentuk pembagian laju transmisi tautan di antara banyak aliran data. Circuit switching pra-mengalokasikan penggunaan tautan transmisi kurang memperhatikan permintaan, dengan waktu tautan yang dialokasikan tetapi tidak dibutuhkan menjadi tidak terpakai. Pergantian paket di sisi lain mengalokasikan penggunaan tautan sesuai permintaan. Kapasitas transmisi link akan dibagi berdasarkan paket per paket hanya di antara para pengguna yang memiliki paket yang perlu ditransmisikan melalui link.

Meskipun packet switching dan circuit switching sama-sama lazim di jaringan telekomunikasi saat ini, trennya pasti mengarah ke packet switching. Bahkan banyak jaringan telepon circuit-switched saat ini secara perlahan bermigrasi ke arah packet switching. Secara khusus, jaringan telepon sering menggunakan pengalihan paket untuk porsi panggilan telepon ke luar negeri yang mahal.

1.3.3 Jaringan dari Jaringan

Kita telah melihat sebelumnya bahwa sistem akhir (PC, telepon pintar, server Web, server email, dan sebagainya) terhubung ke Internet melalui ISP akses. ISP akses dapat menyediakan koneksi kabel atau nirkabel, menggunakan rangkaian teknologi akses termasuk DSL, kabel, FTTH, Wi-Fi, dan seluler. Perhatikan bahwa ISP akses tidak harus berupa perusahaan telekomunikasi atau perusahaan kabel; sebaliknya dapat berupa, misalnya, universitas (menyediakan akses Internet untuk mahasiswa, staf, dan fakultas), atau perusahaan (menyediakan akses untuk karyawannya). Tetapi menghubungkan pengguna akhir dan penyedia konten ke ISP akses hanyalah sebagian kecil dari pemecahan teka-teki menghubungkan miliaran sistem akhir yang membentuk Internet. Untuk menyelesaikan teka-teki ini, ISP akses itu sendiri harus saling berhubungan. Ini dilakukan dengan membuat *jaringan dari jaringan*—memahami ungkapan ini adalah kunci untuk memahami Internet.

Selama bertahun-tahun, jaringan dari jaringan yang membentuk Internet telah berkembang menjadi struktur yang sangat kompleks. Banyak dari evolusi ini didorong oleh ekonomi dan kebijakan nasional, bukan oleh pertimbangan kinerja. Untuk memahami struktur jaringan Internet saat ini, mari kita secara bertahap membangun rangkaian struktur jaringan, dengan setiap struktur baru merupakan perkiraan yang lebih baik dari Internet kompleks yang kita miliki saat ini. Ingatlah bahwa tujuan utamanya adalah untuk menghubungkan ISP akses sehingga semua sistem akhir dapat saling mengirim paket. Satu pendekatan naif adalah membuat setiap ISP akses *langsung* terhubung dengan setiap ISP akses lainnya. Desain mesh seperti itu, tentu saja, terlalu mahal untuk ISP akses, karena akan membutuhkan setiap ISP akses untuk memiliki tautan komunikasi terpisah ke masing-masing dari ratusan ribu ISP akses lainnya di seluruh dunia.

Struktur jaringan pertama kami, *Struktur Jaringan 1*, menghubungkan semua ISP akses dengan satu *ISP transit global*. ISP transit global (imajiner) kami adalah jaringan router dan tautan komunikasi yang tidak hanya menjangkau dunia, tetapi juga memiliki setidaknya satu router di dekat masing-masing dari ratusan ribu ISP akses.

Tentu saja, akan sangat mahal bagi ISP global untuk membangun jaringan seluas itu. Agar menguntungkan, secara alami akan membebankan masing-masing ISP akses untuk koneksi, dengan harga yang mencerminkan (tetapi tidak harus berbanding lurus dengan) jumlah lalu lintas yang dipertukarkan oleh ISP akses dengan ISP global. Karena ISP akses membayar ISP transit global, ISP akses dikatakan sebagai **pelanggan** dan ISP transit global dikatakan sebagai **penyedia**.

Sekarang jika beberapa perusahaan membangun dan mengoperasikan ISP transit global yang menguntungkan, maka wajar bagi perusahaan lain untuk membangun ISP transit global mereka sendiri dan bersaing dengan ISP transit global asli. Ini mengarah ke *Struktur Jaringan 2*, yang terdiri dari ratusan ribu ISP akses dan *beberapa* ISP transit global. ISP akses pasti lebih memilih Struktur Jaringan 2 daripada Struktur Jaringan 1 karena mereka sekarang dapat memilih di antara penyedia transit global yang bersaing sebagai fungsi dari harga dan layanan mereka. Perhatikan, bagaimanapun, bahwa ISP transit global

mereka sendiri harus saling terhubung: Jika tidak, ISP akses yang terhubung ke salah satu penyedia transit global tidak akan dapat berkomunikasi dengan ISP akses yang terhubung ke penyedia transit global lainnya.

Struktur Jaringan 2, baru saja dijelaskan, adalah hierarki dua tingkat dengan penyedia transit global berada di tingkat atas dan mengakses ISP di tingkat bawah. Ini mengasumsikan bahwa ISP transit global tidak hanya mampu mendekati setiap ISP akses, tetapi juga merasa diinginkan secara ekonomis untuk melakukannya. Pada kenyataannya, meskipun beberapa ISP memiliki jangkauan global yang mengesankan dan terhubung langsung dengan banyak ISP akses, tidak ada ISP yang hadir di setiap kota di dunia. Sebaliknya, di wilayah tertentu, mungkin ada **ISP regional** yang terhubung dengan ISP akses di wilayah tersebut. Setiap ISP regional kemudian terhubung ke **ISP tier-1**. ISP Tier-1 serupa dengan ISP transit global (imajiner) kami; tetapi ISP tier-1, yang sebenarnya ada, tidak hadir di setiap kota di dunia. Ada sekitar selusin ISP tier-1, termasuk Komunikasi Level 3, AT&T, Sprint, dan NTT. Menariknya, tidak ada grup yang secara resmi memberikan sanksi status tier-1; seperti kata pepatah — jika Anda harus bertanya apakah Anda anggota grup, Anda mungkin tidak.

Kembali ke jaringan jaringan ini, tidak hanya ada beberapa ISP tingkat 1 yang bersaing, mungkin ada beberapa ISP regional yang bersaing di suatu wilayah. Dalam hierarki seperti itu, setiap ISP akses membayar ISP regional yang terhubung, dan setiap ISP regional membayar ISP tier-1 yang terhubung. (ISP akses juga dapat terhubung langsung ke ISP tier-1, dalam hal ini membayar ISP tier-1). Dengan demikian, ada hubungan pelanggan-penyedia pada setiap tingkat hierarki. Perhatikan bahwa ISP tier-1 tidak membayar siapa pun karena mereka berada di puncak hierarki. Untuk lebih memperumit masalah, di beberapa wilayah, mungkin ada ISP regional yang lebih besar (mungkin mencakup seluruh negara) yang terhubung dengan ISP regional yang lebih kecil di wilayah tersebut; ISP regional yang lebih besar kemudian terhubung ke ISP tier-1. Misalnya, di Cina, ada ISP akses di setiap kota, yang terhubung ke ISP provinsi, yang pada gilirannya terhubung ke ISP nasional, yang akhirnya terhubung ke ISP tier-1 [Tian 2012]. Kami mengacu pada hierarki multi-tingkat ini, yang masih merupakan perkiraan kasar dari Internet saat ini, sebagai *Struktur Jaringan 3*.

Untuk membangun jaringan yang lebih menyerupai Internet saat ini, kita harus menambahkan point of presence (PoP), multi-homing, peering, dan Internet exchange point (IXP) ke Struktur Jaringan hierarkis 3. PoP ada di semua tingkatan hierarki, kecuali untuk tingkat bawah (akses ISP). PoP hanyalah sekelompok satu atau lebih router (di lokasi yang sama) di jaringan penyedia tempat ISP pelanggan dapat terhubung ke ISP penyedia. Agar jaringan pelanggan terhubung ke PoP penyedia, ia dapat menyewa tautan berkecepatan tinggi dari penyedia telekomunikasi pihak ketiga untuk menghubungkan langsung salah satu routernya ke router di PoP. Setiap ISP (kecuali untuk ISP tier-1) dapat memilih untuk **multi-home**, yaitu untuk terhubung ke dua atau lebih ISP penyedia. Jadi, misalnya, ISP akses mungkin multi-rumah dengan dua ISP regional, atau mungkin multi-rumah dengan dua ISP regional dan juga dengan ISP tier-1. Demikian pula, ISP regional mungkin multi-rumah dengan beberapa ISP tier-1. Ketika sebuah

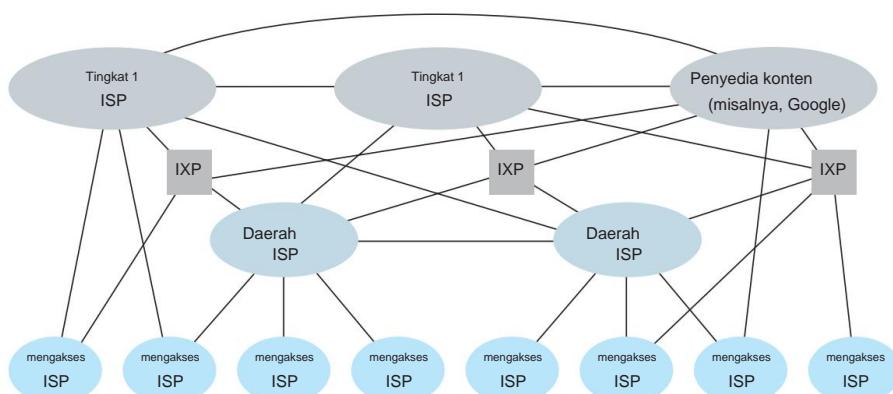
34 BAB 1

• JARINGAN KOMPUTER DAN INTERNET

ISP multi-rumah, ia dapat terus mengirim dan menerima paket ke Internet meskipun salah satu penyedianya mengalami kegagalan.

Seperti yang baru saja kita pelajari, ISP pelanggan membayar ISP penyedia mereka untuk mendapatkan interkoneksi Internet global. Jumlah yang dibayarkan ISP pelanggan kepada ISP penyedia mencerminkan jumlah lalu lintas yang dipertukarkannya dengan penyedia. Untuk mengurangi biaya ini, sepasang ISP terdekat pada tingkat hierarki yang sama dapat **mengintip**, yaitu, mereka dapat langsung menghubungkan jaringan mereka bersama sehingga semua lalu lintas di antara mereka melewati koneksi langsung daripada melalui perantara hulu. Ketika dua ISP peer, biasanya tidak ada penyelesaian, artinya, tidak ada ISP yang membayar yang lain. Seperti disebutkan sebelumnya, ISP tier-1 juga melakukan peer satu sama lain, bebas penyelesaian. Untuk diskusi yang dapat dibaca tentang peering dan hubungan pelanggan-penyedia, lihat [Van der Eijk, 2009]. Sepanjang jalur yang sama ini, perusahaan pihak ketiga dapat membuat **Internet Exchange Point (IXP)** (biasanya di gedung yang berdiri sendiri dengan sakelarnya sendiri), yang merupakan titik pertemuan di mana beberapa ISP dapat saling mengintip. Ada sekitar 300 IXP di Internet saat ini [Agustus 2009]. Kami menyebut ekosistem ini—yang terdiri dari ISP akses, ISP regional, ISP tier-1, PoP, multi-homing, peering, dan IXP—sebagai *Struktur Jaringan 4*.

Akhirnya kita sampai pada *Struktur Jaringan 5*, yang menjelaskan Internet tahun 2012. Struktur Jaringan 5, diilustrasikan pada Gambar 1.15, dibangun di atas Struktur Jaringan 4 dengan menambahkan **jaringan penyedia konten**. Google saat ini adalah salah satu contoh terkemuka dari jaringan penyedia konten semacam itu. Saat tulisan ini dibuat, diperkirakan Google memiliki 30 hingga 50 pusat data yang tersebar di Amerika Utara, Eropa, Asia, Amerika Selatan, dan Australia. Beberapa dari pusat data ini menampung lebih dari seratus ribu server, sementara pusat data lainnya lebih kecil, hanya menampung ratusan server. Semua pusat data Google saling terhubung melalui jaringan TCP/IP pribadi Google, yang menjangkau seluruh dunia namun tetap terpisah dari Internet publik. Yang penting, jaringan pribadi Google saja



Gambar 1.15 Interkoneksi ISP

membawa lalu lintas ke/dari server Google. Seperti yang ditunjukkan pada Gambar 1.15, jaringan pribadi Google mencoba untuk "melewati" tingkat atas Internet dengan peering (settlement free) dengan ISP tingkat rendah, baik dengan langsung terhubung dengan mereka atau dengan terhubung dengan mereka di IXPs [Labovitz 2010]. Namun, karena banyak ISP akses yang hanya dapat dijangkau dengan transit melalui jaringan tingkat-1, jaringan Google juga terhubung ke ISP tingkat-1, dan membayar ISP tersebut untuk lalu lintas yang dipertukarkan dengan mereka. Dengan membuat jaringannya sendiri, penyedia konten tidak hanya mengurangi pembayarannya ke ISP tingkat atas, tetapi juga memiliki kontrol yang lebih besar tentang bagaimana layanannya pada akhirnya dikirimkan ke pengguna akhir. Infrastruktur jaringan Google dijelaskan secara lebih rinci di Bagian 7.2.4.

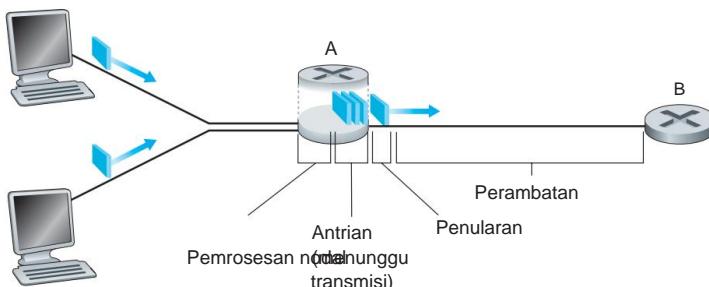
Singkatnya, Internet saat ini—sebuah jaringan dari jaringan—adalah kompleks, terdiri dari selusin atau lebih ISP tingkat-1 dan ratusan ribu ISP tingkat bawah. ISP beragam dalam cakupannya, dengan beberapa mencakup beberapa benua dan samudra, dan lainnya terbatas pada wilayah geografis yang sempit. ISP tingkat rendah terhubung ke ISP tingkat tinggi, dan ISP tingkat tinggi saling terhubung satu sama lain. Pengguna dan penyedia konten adalah pelanggan ISP tingkat rendah, dan ISP tingkat rendah adalah pelanggan ISP tingkat tinggi. Dalam beberapa tahun terakhir, penyedia konten utama juga telah membuat jaringan mereka sendiri dan terhubung langsung ke ISP tingkat rendah jika memungkinkan.

1.4 Delay, Loss, dan Throughput di Jaringan Packet-Switched

Kembali ke Bagian 1.1 kami mengatakan bahwa Internet dapat dilihat sebagai infrastruktur yang menyediakan layanan untuk aplikasi terdistribusi yang berjalan pada sistem akhir. Idealnya, kami ingin layanan Internet dapat memindahkan data sebanyak yang kami inginkan antara dua sistem akhir mana pun, secara instan, tanpa kehilangan data. Sayangnya, ini adalah tujuan yang mulia, yang pada kenyataannya tidak dapat dicapai. Sebaliknya, jaringan komputer perlu membatasi throughput (jumlah data per detik yang dapat ditransfer) antara sistem akhir, menimbulkan penundaan antara sistem akhir, dan benar-benar dapat kehilangan paket. Di satu sisi, sangat disayangkan bahwa hukum fisik realitas menyebabkan keterlambatan dan kerugian serta membatasi throughput. Di sisi lain, karena jaringan komputer memiliki masalah ini, ada banyak isu menarik seputar bagaimana menangani masalah tersebut—lebih dari cukup isu untuk mengisi mata kuliah jaringan komputer dan memotivasi ribuan tesis PhD! Pada bagian ini, kita akan mulai memeriksa dan menghitung delay, loss, dan throughput dalam jaringan komputer.

1.4.1 Tinjauan Delay pada Jaringan Packet-Switched

Ingatlah bahwa sebuah paket dimulai di host (sumber), melewati serangkaian router, dan mengakhiri perjalannya di host lain (tujuan). Saat paket berjalan dari satu node (host atau router) ke node berikutnya (host atau router) di sepanjang jalur ini,



Gambar 1.16 Nodal delay pada router A

paket mengalami beberapa jenis penundaan pada *setiap* node di sepanjang jalur. Delay yang paling penting adalah **delay pemrosesan nodal**, **delay antrian**, **delay transmisi**, dan **delay propagasi**; bersama-sama, penundaan ini terakumulasi untuk memberikan **penundaan nodal total**. Performa banyak aplikasi Internet—seperti pencarian, penelusuran Web, email, peta, pesan instan, dan voice-over-IP—sangat dipengaruhi oleh penundaan jaringan. Untuk memperoleh pemahaman mendalam tentang packet switching dan jaringan komputer, kita harus memahami sifat dan pentingnya penundaan ini.

Jenis Keterlambatan

Mari jelajahi keterlambatan ini dalam konteks Gambar 1.16. Sebagai bagian dari rute end-to-end antara sumber dan tujuan, sebuah paket dikirim dari node upstream melalui router A ke router B. Tujuan kami adalah mengkarakterisasi delay di router A.

Perhatikan bahwa router A memiliki tautan keluar yang mengarah ke router B. Tautan ini didahului oleh antrian (juga dikenal sebagai buffer). Ketika paket tiba di router A dari node upstream, router A memeriksa header paket untuk menentukan tautan keluar yang sesuai untuk paket tersebut dan kemudian mengarahkan paket ke tautan ini. Dalam contoh ini, tautan keluar untuk paket adalah yang mengarah ke router B. Sebuah paket dapat ditransmisikan pada tautan hanya jika tidak ada paket lain yang saat ini sedang dikirim pada tautan tersebut dan jika tidak ada paket lain yang mendahuluinya. antrian; jika link sedang sibuk atau jika ada paket lain yang sudah antri untuk link tersebut, paket yang baru tiba akan bergabung dengan antrian.

Keterlambatan Pemrosesan

Waktu yang diperlukan untuk memeriksa header paket dan menentukan ke mana harus mengarahkan paket adalah bagian dari **penundaan pemrosesan**. Penundaan pemrosesan juga dapat mencakup faktor lain, seperti waktu yang diperlukan untuk memeriksa kesalahan level bit dalam paket yang terjadi saat mentransmisikan bit paket dari node upstream ke router A. Penundaan pemrosesan

di router berkecepatan tinggi biasanya dalam urutan mikrodetik atau kurang. Setelah pemrosesan nodal ini, router mengarahkan paket ke antrian yang mendahului tautan ke router B. (Dalam Bab 4 kita akan mempelajari detail tentang cara kerja router.)

Penundaan Antrian

Di antrian, paket mengalami **penundaan antrian** saat menunggu untuk dikirim ke tautan. Lamanya penundaan antrian dari paket tertentu akan bergantung pada jumlah paket yang tiba lebih awal yang diantrekan dan menunggu untuk ditransmisikan ke link. Jika antrian kosong dan tidak ada paket lain yang sedang dikirim, maka delay antrian paket kita akan menjadi nol. Di sisi lain, jika lalu lintas padat dan banyak paket lain juga menunggu untuk dikirim, penundaan antrian akan lama. Kita akan segera melihat bahwa jumlah paket yang diharapkan ditemukan oleh paket yang datang adalah fungsi dari intensitas dan sifat lalu lintas yang tiba di antrian. Penundaan antrian bisa dalam urutan mikrodetik hingga milidetik dalam praktiknya.

Penundaan Transmisi

Dengan asumsi bahwa paket ditransmisikan dengan cara first-come-first-served, seperti yang biasa terjadi di jaringan packet-switched, paket kita hanya dapat ditransmisikan setelah semua paket yang datang sebelum ditransmisikan. Nyatakan panjang paket dengan L bit, dan nyatakan laju transmisi tautan dari router A ke router B dengan R bit/detik. Misalnya, untuk tautan Ethernet 10 Mbps, tarifnya adalah $R = 10$ Mbps; untuk tautan Ethernet 100 Mbps, kecepatannya adalah $R = 100$ Mbps. Penundaan **transmisi** adalah L/R . Ini adalah jumlah waktu yang diperlukan untuk mendorong (yaitu, mengirimkan) semua bit paket ke dalam tautan. Penundaan transmisi biasanya dalam urutan mikrodetik hingga milidetik dalam praktiknya.

Penundaan Perbanyak

Setelah sedikit didorong ke dalam link, bit tersebut perlu disebarluaskan ke router B. Waktu yang dibutuhkan untuk merambat dari awal link ke router B adalah **delay propagasi**. Bit merambat pada kecepatan propagasi tautan. Kecepatan propagasi tergantung pada media fisik tautan (yaitu, serat optik, kawat tembaga bengkok, dan sebagainya) dan berada dalam kisaran

2 108 meter/detik hingga 3 108 meter/detik

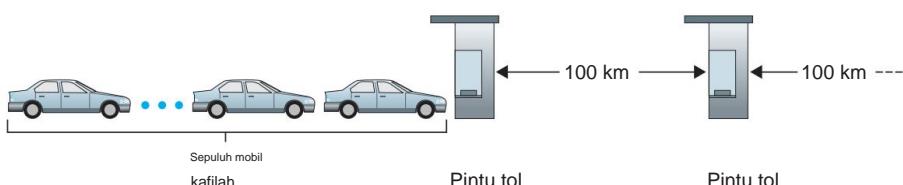
yang sama dengan, atau sedikit kurang dari, kecepatan cahaya. Delay propagasi adalah jarak antara dua router dibagi dengan kecepatan propagasi. Artinya, delay propagasi adalah d/s , dimana d adalah jarak antara router A dan router B dan s adalah kecepatan propagasi dari link. Setelah bit terakhir dari paket menyebar ke node B, bit tersebut dan semua bit sebelumnya dari paket disimpan di router B. Keseluruhan

proses kemudian dilanjutkan dengan router B sekarang melakukan penerusan. Di jaringan area luas, penundaan propagasi berada di urutan milidetik.

Membandingkan Delay Transmisi dan Propagasi

Pendatang baru di bidang jaringan komputer terkadang mengalami kesulitan memahami perbedaan antara delay transmisi dan delay propagasi. Perbedaannya halus tetapi penting. Penundaan transmisi adalah jumlah waktu yang diperlukan router untuk mendorong keluar paket; itu adalah fungsi dari panjang paket dan kecepatan transmisi link, tetapi tidak ada hubungannya dengan jarak antara dua router. Penundaan propagasi, di sisi lain, adalah waktu yang dibutuhkan sedikit untuk menyebar dari satu router ke router berikutnya; itu adalah fungsi dari jarak antara dua router, tetapi tidak ada hubungannya dengan panjang paket atau kecepatan transmisi tautan.

Sebuah analogi dapat memperjelas pengertian tentang penundaan transmisi dan propagasi. Pertimbangkan jalan raya yang memiliki pintu tol setiap 100 kilometer, seperti yang ditunjukkan pada Gambar 1.17. Anda dapat menganggap ruas jalan tol antara gardu tol sebagai penghubung dan gardu tol sebagai router. Misalkan mobil berjalan (yaitu, merambat) di jalan raya dengan kecepatan 100 km/jam (yaitu, ketika sebuah mobil meninggalkan pintu tol, kecepatannya seketika menjadi 100 km/jam dan mempertahankan kecepatan tersebut di antara pintu tol). Misalkan selanjutnya 10 mobil, bepergian bersama sebagai karavan, mengikuti satu sama lain dalam urutan yang tetap. Anda dapat menganggap setiap mobil sebagai sedikit kafilah sebagai satu paket. Juga anggaplah bahwa setiap pintu tol melayani (yaitu, mentransmisikan) sebuah mobil dengan kecepatan satu mobil per 12 detik, dan saat itu larut malam sehingga mobil karavan adalah satu-satunya mobil di jalan raya. Terakhir, misalkan setiap kali mobil pertama dari karavan tiba di pintu tol, ia menunggu di pintu masuk sampai sembilan mobil lainnya tiba dan berbaris di belakangnya. (Dengan demikian seluruh karavan harus disimpan di gardu tol sebelum dapat mulai dijaga.) Waktu yang diperlukan gardu tol untuk mendorong seluruh kafilah ke jalan raya adalah (10 mobil)/(5 mobil/menit) = 2 menit. Kali ini analog dengan penundaan transmisi di router. Waktu yang diperlukan sebuah mobil untuk menempuh perjalanan dari pintu tol satu pintu tol ke pintu tol berikutnya adalah $100 \text{ km}/(100 \text{ km/jam}) = 1 \text{ jam}$. Waktu ini analog dengan delay propagasi. Oleh karena itu, waktu dari saat karavan disimpan di depan gardu tol hingga karavan disimpan di depan gardu tol berikutnya adalah penjumlahan penundaan transmisi dan penundaan propagasi—dalam contoh ini, 62 menit.



Gambar 1.17 Analogi Kafilah

Mari jelajahi analogi ini lebih jauh. Apa yang terjadi jika waktu pelayanan gardu tol untuk karavan lebih besar daripada waktu tempuh mobil antar gardu tol?

Sebagai contoh, misalkan sekarang mobil berjalan dengan kecepatan 1.000 km/jam dan gardu tol melayani mobil dengan kecepatan satu mobil per menit. Kemudian tunda perjalanan antara dua gardu tol adalah 6 menit dan waktu untuk melayani karavan adalah 10 menit. Dalam hal ini, beberapa mobil pertama dalam karavan akan tiba di gardu tol kedua sebelum mobil terakhir dalam karavan tersebut meninggalkan gardu tol pertama. Situasi ini juga muncul dalam jaringan packet-switched — bit pertama dalam sebuah paket dapat tiba di router sementara banyak bit yang tersisa dalam paket masih menunggu untuk ditransmisikan oleh router sebelumnya.

Jika sebuah gambar berbicara seribu kata, maka animasi harus berbicara sejuta kata. Situs Web pendamping untuk buku teks ini menyediakan applet Java interaktif yang mengilustrasikan dengan baik dan membedakan penundaan transmisi dan penundaan propagasi. Pembaca sangat dianjurkan untuk mengunjungi applet tersebut. [Smith 2009] juga memberikan diskusi yang sangat mudah dibaca tentang propagasi, antrian, dan penundaan transmisi.

Jika kita membiarkan dproc, dqueue, dtrans, dan dprop menunjukkan pemrosesan, antrian, transmisi, dan delay propagasi, maka total delay nodal diberikan oleh

$$\text{dnodal} = \text{dproc} + \text{dqueue} + \text{dtrans} + \text{dprop}$$

Kontribusi komponen penundaan ini dapat sangat bervariasi. Misalnya, dprop dapat diabaikan (misalnya, beberapa mikrodetik) untuk tautan yang menghubungkan dua router di kampus universitas yang sama; namun, dprop adalah ratusan milidetik untuk dua router yang saling terhubung oleh tautan satelit geostasioner, dan dapat menjadi istilah dominan dalam dnodal.

Demikian pula, dtrans dapat berkisar dari dapat diabaikan hingga signifikan. Kontribusinya biasanya diabaikan untuk kecepatan transmisi 10 Mbps dan lebih tinggi (misalnya, untuk LAN); namun, bisa ratusan milidetik untuk paket Internet besar yang dikirim melalui tautan modem dial-up berkecepatan rendah. Penundaan pemrosesan, dproc, seringkali dapat diabaikan; namun, ini sangat mempengaruhi throughput maksimum router, yang merupakan kecepatan maksimum router dapat meneruskan paket.

1.4.2 Penundaan Antrian dan Kehilangan Paket

Komponen delay nodal yang paling rumit dan menarik adalah delay antrian, dqueue. Faktanya, penundaan antrian sangat penting dan menarik dalam kerja jaringan komputer sehingga ribuan makalah dan banyak buku telah ditulis tentangnya [Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ros 1995]. Kami hanya memberikan diskusi intuitif tingkat tinggi tentang penundaan antrian di sini; pembaca yang lebih penasaran mungkin ingin menelusuri beberapa buku (atau bahkan akhirnya menulis tesis PhD tentang subjek tersebut!). Berbeda dengan tiga penundaan lainnya (yaitu, dproc, dtrans, dan dprop), penundaan antrian dapat bervariasi dari satu paket ke paket lainnya. Misalnya, jika 10 paket tiba di antrian kosong pada saat yang sama, paket pertama yang dikirimkan tidak akan mengalami penundaan antrian, sedangkan paket terakhir yang dikirimkan akan mengalami penundaan antrian yang relatif besar (sementara menunggu sembilan paket lainnya selesai). Oleh karena itu, kapan

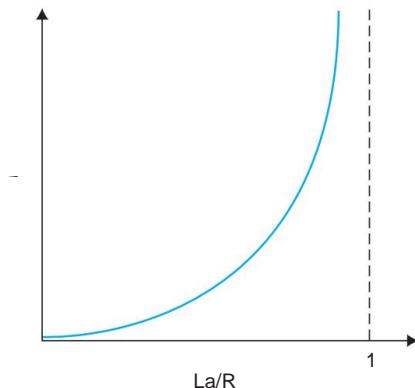
40 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

mencirikan keterlambatan antrian, seseorang biasanya menggunakan ukuran statistik, seperti usia rata-rata penundaan antrian, varian penundaan antrian, dan probabilitas bahwa penundaan antrian melebihi beberapa nilai yang ditentukan.

Kapan penundaan antrian besar dan kapan tidak signifikan? Jawaban atas pertanyaan ini bergantung pada laju lalu lintas yang tiba di antrian, laju transmisi tautan, dan sifat lalu lintas yang datang, yaitu apakah lalu lintas datang secara periodik atau tiba dalam ledakan. Untuk mendapatkan beberapa wawasan di sini, misalkan a menunjukkan tingkat rata-rata di mana paket tiba di antrian (a dalam satuan paket/detik). Ingatlah bahwa R adalah laju transmisi; yaitu, kecepatan (dalam bit/detik) di mana bit didorong keluar dari antrian. Anggap juga, untuk penyederhanaan, bahwa semua paket terdiri dari L bit. Maka kecepatan rata-rata di mana bit tiba di antrian adalah La bit/detik. Akhirnya, asumsikan bahwa antrian sangat besar, sehingga pada dasarnya dapat menampung jumlah bit yang tidak terbatas. Rasio La/R , yang disebut **intensitas lalu lintas**, sering memainkan peran penting dalam memperkirakan tingkat penundaan antrian. Jika $La/R > 1$, maka kecepatan rata-rata di mana bit tiba di antrian melebihi kecepatan di mana bit dapat ditransmisikan dari antrian. Dalam situasi yang tidak menguntungkan ini, antrian akan cenderung bertambah tanpa batas dan penundaan antrian akan mendekati tak terhingga! Oleh karena itu, salah satu aturan emas dalam rekayasa lalu lintas adalah: *Rancang sistem Anda sehingga intensitas lalu lintas tidak lebih dari 1.*

Sekarang pertimbangkan kasus $La/R \leq 1$. Di sini, sifat lalu lintas kedatangan berdampak pada penundaan antrian. Sebagai contoh, jika paket tiba secara periodik—yaitu, satu paket tiba setiap detik L/R —maka setiap paket akan tiba pada antrian kosong dan tidak akan ada penundaan antrian. Di sisi lain, jika paket datang dalam jumlah besar tetapi secara berkala, dapat terjadi penundaan antrian rata-rata yang signifikan. Sebagai contoh, misalkan N paket tiba secara bersamaan setiap $(L/R)/N$ detik. Kemudian paket pertama yang dikirim tidak memiliki delay antrian; paket kedua yang ditransmisikan memiliki delay antrian sebesar L/R detik; dan lebih umum lagi, paket ke- n yang ditransmisikan memiliki penundaan antrian selama $(n-1)L/R$ detik. Kami membiarkannya sebagai latihan bagi Anda untuk menghitung rata-rata penundaan antrian dalam contoh ini.

Dua contoh kedatangan berkala yang dijelaskan di atas sedikit bersifat akademis. Biasanya, proses kedatangan ke antrian adalah *acak*; yaitu, kedatangan tidak mengikuti pola apa pun dan paket-paket dipisahkan oleh jumlah waktu yang acak. Dalam kasus yang lebih realistik ini, besaran La/R biasanya tidak cukup untuk sepenuhnya mencirikan statistik penundaan antrian. Meskipun demikian, ini berguna untuk mendapatkan pemahaman intuitif tentang tingkat penundaan antrian. Secara khusus, jika intensitas lalu lintas mendekati nol, maka kedatangan paket sedikit dan jarang dan tidak mungkin paket yang datang akan menemukan paket lain dalam antrian. Oleh karena itu, penundaan antrian rata-rata akan mendekati nol. Di sisi lain, ketika intensitas lalu lintas mendekati 1, akan ada interval waktu ketika tingkat kedatangan melebihi kapasitas transmisi (karena variasi tingkat kedatangan paket), dan antrian akan terbentuk selama periode waktu ini; ketika tingkat kedatangan kurang dari kapasitas transmisi, panjang antrian akan menyusut. Meskipun demikian, ketika intensitas lalu lintas mendekati 1, panjang antrian rata-rata menjadi semakin besar. Ketergantungan kualitatif tundaan antrian rata-rata terhadap intensitas lalu lintas ditunjukkan pada Gambar 1.18.



Gambar 1.18 Ketergantungan tundaan antrian rata-rata terhadap intensitas lalu lintas

Salah satu aspek penting dari Gambar 1.18 adalah kenyataan bahwa ketika intensitas lalu lintas mendekati 1, rata-rata tundaan antrian meningkat dengan cepat. Persentase peningkatan intensitas yang kecil akan menghasilkan peningkatan persentase yang jauh lebih besar. Mungkin Anda pernah mengalami fenomena ini di jalan raya. Jika Anda secara teratur berkendara di jalan yang biasanya macet, fakta bahwa jalan tersebut biasanya padat berarti intensitas lalu lintasnya mendekati 1. Jika beberapa peristiwa menyebabkan jumlah lalu lintas yang bahkan sedikit lebih besar dari biasanya, penundaan Anda pengalaman bisa sangat besar.

Untuk benar-benar memahami tentang apa itu penundaan antrian, Anda didorong sekali lagi untuk mengunjungi situs Web pendamping, yang menyediakan applet Java interaktif untuk antrian. Jika Anda menyetel tingkat kedatangan paket cukup tinggi sehingga intensitas lalu lintas melebihi 1, Anda akan melihat antrean perlahan bertambah seiring waktu.

Kehilangan Paket

Dalam diskusi kami di atas, kami berasumsi bahwa antrian mampu menampung paket dalam jumlah tak terbatas. Pada kenyataannya sebuah antrian yang mendahului sebuah link memiliki kapasitas yang terbatas, meskipun kapasitas antrian sangat bergantung pada desain dan biaya router.

Karena kapasitas antrian terbatas, penundaan paket tidak benar-benar mendekati tak terhingga saat intensitas lalu lintas mendekati 1. Sebaliknya, sebuah paket dapat tiba untuk menemukan antrian penuh. Tanpa tempat untuk menyimpan paket seperti itu, router akan **menjatuhkan** paket itu; artinya, paket akan **hilang**. Overflow pada antrian ini dapat dilihat lagi di applet Java untuk antrian ketika intensitas lalu lintas lebih besar dari 1.

Dari sudut pandang sistem akhir, kehilangan paket akan terlihat seperti paket yang telah dikirim ke inti jaringan tetapi tidak pernah keluar dari jaringan di tujuan. Fraksi paket yang hilang meningkat seiring dengan meningkatnya intensitas lalu lintas.

Oleh karena itu, kinerja pada sebuah node sering diukur tidak hanya dalam hal keterlambatan, tetapi juga dalam hal probabilitas kehilangan paket. Seperti yang akan kita bahas selanjutnya

42 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

bab, paket yang hilang dapat ditransmisikan ulang secara end-to-end untuk memastikan bahwa semua data pada akhirnya ditransfer dari sumber ke tujuan

1.4.3 Penundaan Ujung ke Ujung

Pembahasan kita sampai saat ini terfokus pada delay nodal, yaitu delay pada satu router. Sekarang mari kita pertimbangkan penundaan total dari sumber ke tujuan. Untuk memahami konsep ini, misalkan ada N router antara host sumber dan host tujuan. Misalkan juga untuk saat jaringan tidak terkoneksi (sehingga penundaan antrian dapat diabaikan), penundaan pemrosesan di setiap router dan di host sumber adalah dproc, laju transmisi dari setiap router dan keluar dari host sumber adalah R bit/detik, dan propagasi pada setiap tautan adalah dprop. Penundaan nodal terakumulasi dan memberikan penundaan end-to-end,

$$\text{end-to-end} = N (\text{dproc} + \text{dtrans} + \text{dprop}) \quad (1.2)$$

dimana, sekali lagi, dtrans = L/R , dimana L adalah ukuran paket. Perhatikan bahwa Persamaan 1.2 adalah generalisasi dari Persamaan 1.1, yang tidak memperhitungkan penundaan pemrosesan dan propagasi. Kita serahkan kepada Anda untuk menggeneralisasikan Persamaan 1.2 untuk kasus keterlambatan heterogen pada node dan adanya penundaan antrian rata-rata pada setiap node.

Traceroute



Untuk merasakan keterlambatan end-to-end dalam jaringan komputer, kita dapat menggunakan program Traceroute. Traceroute adalah program sederhana yang dapat dijalankan di semua host jaringan Inter. Saat pengguna menentukan nama host tujuan, program di host sumber mengirimkan beberapa paket khusus ke tujuan tersebut. Saat paket-paket ini bekerja menuju tujuan, mereka melewati serangkaian router. Ketika sebuah router menerima salah satu dari paket khusus ini, ia mengirimkannya kembali ke sumber pesan singkat yang berisi nama dan alamat router.

Lebih khusus lagi, misalkan ada N router antara sumber dan tujuan. Kemudian sumber akan mengirimkan N paket khusus ke dalam jaringan, dengan setiap paket ditujukan ke tujuan akhir. N paket khusus ini ditandai 1 sampai N , dengan paket pertama ditandai 1 dan paket terakhir ditandai N . Ketika router ke- n menerima paket ke- n bertanda n , router tidak meneruskan paket ke tujuannya, melainkan mengirim pesan kembali ke sumbernya. Ketika host tujuan menerima paket N th, itu juga mengembalikan pesan kembali ke sumbernya. Sumber mencatat waktu yang berlalu antara saat mengirim paket dan saat menerima pesan balasan yang sesuai; itu juga mencatat nama dan alamat router (atau host tujuan) yang mengembalikan pesan. Dengan cara ini, sumber dapat merekonstruksi rute yang diambil oleh paket yang mengalir dari sumber ke tujuan, dan sumber dapat menentukan penundaan perjalanan bolak-balik ke semua router yang mengintervensi. Trace route sebenarnya mengulangi percobaan yang baru saja dijelaskan tiga kali, sehingga sumber benar-benar mengirim 3 • N paket ke tujuan. RFC 1393 menjelaskan Traceroute secara detail.

Berikut adalah contoh keluaran program Traceroute, di mana rute ditelusuri dari host sumber gaia.cs.umass.edu (di University of Massachusetts) ke host cis.poly.edu (di Universitas Politeknik di Brooklyn). Keluaran memiliki enam kolom: kolom pertama adalah nilai n yang dijelaskan di atas, yaitu jumlah router di sepanjang rute; kolom kedua adalah nama router; kolom ketiga adalah alamat router (dalam bentuk xxx.xxx.xxx.xxx); tiga kolom terakhir adalah penundaan bolak-balik untuk tiga percobaan. Jika sumber menerima kurang dari tiga pesan dari router mana pun (karena kehilangan paket di jaringan), Traceroute menempatkan tanda bintang tepat setelah nomor router dan melaporkan kurang dari tiga kali perjalanan bolak-balik untuk router tersebut.

```
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.umass8.11  
(19.222.11) ) 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.Level3.net  
(209.244.160.133) gige9-1-52.hsipaccess1.NewYork1.Level3.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 edu (128.238.32.126) 14.080 ms 13.035  
ms 12.802 ms
```

Pada jejak di atas terdapat sembilan router antara sumber dan tujuan.

Sebagian besar router ini memiliki nama, dan semuanya memiliki alamat. Misalnya, nama Router 3 adalah border4-rt-gi-1-3.gw.umass.edu dan alamatnya 128.119.2.194. Melihat data yang disediakan untuk router yang sama ini, kami melihat bahwa dalam percobaan pertama dari tiga uji coba, penundaan bolak-balik antara sumber dan router adalah 1,03 msec. Penundaan pulang-pergi untuk dua percobaan berikutnya adalah 0,48 dan 0,45 msec. Penundaan bolak-balik ini mencakup semua penundaan yang baru saja dibahas, termasuk penundaan transmisi, penundaan propagasi, penundaan pemrosesan router, dan penundaan antrian. Karena delay antrian bervariasi dengan waktu, delay round-trip paket n yang dikirim ke router n terkadang bisa lebih lama dari delay round-trip paket $n+1$ yang dikirim ke router $n+1$. Memang, kami mengamati fenomena ini pada contoh di atas: penundaan ke Router 6 lebih besar daripada penundaan ke Router 7!

Ingin mencoba Traceroute sendiri? Kami sangat menyarankan Anda mengunjungi <http://www.traceroute.org>, yang menyediakan antarmuka Web ke daftar lengkap sumber untuk pelacakan rute. Anda memilih sumber dan menyediakan nama host untuk tujuan apa pun. Program Traceroute kemudian melakukan semua pekerjaan. Ada sejumlah program perangkat lunak gratis yang menyediakan antarmuka grafis ke Traceroute; salah satu favorit kami adalah PingPlotter [PingPlotter 2012].

Sistem Akhir, Aplikasi, dan Penundaan Lainnya

Selain penundaan pemrosesan, transmisi, dan propagasi, dapat terjadi penundaan tambahan yang signifikan pada sistem akhir. Misalnya, sistem akhir ingin

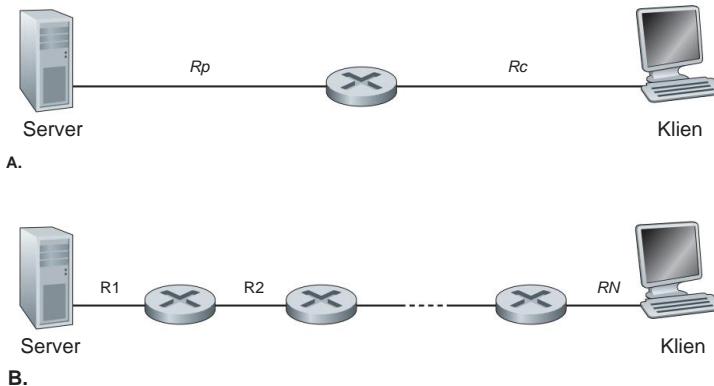
mengirimkan paket ke media bersama (misalnya, seperti dalam skenario WiFi atau modem kabel) dapat *dengan sengaja* menunda pengirimannya sebagai bagian dari protokolnya untuk berbagi media dengan sistem akhir lainnya; kami akan mempertimbangkan protokol tersebut secara rinci di Bab 5. Penundaan penting lainnya adalah penundaan paketisasi media, yang terdapat dalam aplikasi Voice over-IP (VoIP). Di VoIP, pihak pengirim pertama-tama harus mengisi paket dengan pidato digital yang disandikan sebelum mengirimkan paket ke Internet. Kali ini untuk mengisi paket — disebut penundaan paketisasi — bisa menjadi signifikan dan dapat memengaruhi kualitas panggilan VoIP yang dirasakan pengguna. Masalah ini akan dieksplorasi lebih lanjut dalam masalah pekerjaan rumah di akhir bab ini.

1.4.4 Throughput pada Jaringan Komputer

Selain delay dan packet loss, ukuran kinerja penting lainnya dalam jaringan komputer adalah end-to-end throughput. Untuk menentukan throughput, pertimbangkan untuk mentransfer file besar dari Host A ke Host B melalui jaringan komputer. Transfer ini mungkin, misalnya, klip video besar dari satu rekan ke rekan lainnya dalam sistem berbagi file P2P. Throughput **sesaat** pada setiap saat waktu adalah tingkat (dalam bit / detik) di mana Host B menerima file. (Banyak aplikasi, termasuk banyak sistem berbagi file P2P, menampilkan throughput seketika selama pengunduhan di antarmuka pengguna—mungkin Anda telah mengamati ini sebelumnya!) Jika file terdiri dari bit F dan transfer membutuhkan waktu T detik untuk Host B menerima semua F bit, maka **throughput rata-rata** dari transfer file adalah F/T bit/detik. Untuk beberapa aplikasi, seperti telepon Internet, diinginkan untuk memiliki penundaan yang rendah dan throughput seketika secara konsisten di atas ambang batas tertentu (misalnya, lebih dari 24 kbps untuk beberapa aplikasi telepon Internet dan lebih dari 256 kbps untuk beberapa aplikasi video waktunya). Untuk aplikasi lain, termasuk yang melibatkan transfer file, delay tidak kritis, tetapi diinginkan untuk memiliki throughput setinggi mungkin.

Untuk mendapatkan wawasan lebih jauh tentang konsep penting throughput, mari pertimbangkan beberapa contoh. Gambar 1.19(a) menunjukkan dua sistem ujung, server dan klien, dihubungkan oleh dua jalur komunikasi dan sebuah router. Pertimbangkan throughput untuk transfer file dari server ke klien. Biarkan Rs menunjukkan tingkat tautan antara server dan router; dan Rc menunjukkan laju tautan antara router dan klien. Misalkan satu-satunya bit yang dikirim di seluruh jaringan adalah dari server ke klien. Kami sekarang bertanya, dalam skenario ideal ini, apa throughput server ke klien? Untuk menjawab pertanyaan ini, kita mungkin menganggap bit sebagai *fluida* dan hubungan komunikasi sebagai *pipa*. Jelas, server tidak dapat memompa bit melalui tautannya dengan kecepatan lebih cepat dari Rs bps; dan router tidak dapat meneruskan bit dengan kecepatan lebih cepat dari Rc bps. Jika $Rs < Rc$, maka bit yang dipompa oleh server akan "mengalir" langsung melalui router dan sampai ke klien dengan kecepatan Rs bps, memberikan throughput Rs bps. Sebaliknya, jika $Rc < Rs$, maka router tidak akan dapat meneruskan bit ~~secepatnya~~ kecepatanya. Balasannya hanya akan meninggalkan

1.4 • DELAY, LOSS, DAN THROUGHPUT DALAM JARINGAN PACKET-SWITCHED 45



Gambar 1.19 Throughput untuk transfer file dari server ke client

throughput end-to-end dari R_c . (Perhatikan juga bahwa jika bit terus tiba di router dengan kecepatan R_s , ditentukan bahwa R_s adalah bottleneck. Pada jaringan ini, R_s yang patut)

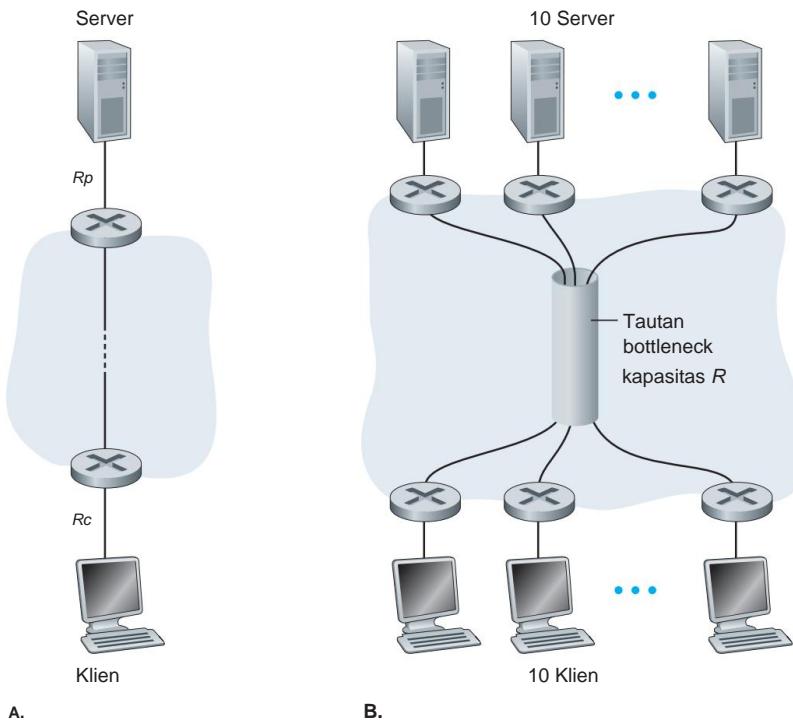
Jadi, untuk jaringan dua-link sederhana ini, throughputnya adalah $\min\{R_c, R_s\}$, yaitu laju transmisi dari **link bottleneck**. Setelah menentukan throughput, sekarang kita dapat memperkirakan waktu yang diperlukan untuk mentransfer file besar bit F dari server ke klien sebagai $F/\min\{R_s, R_c\}$. Untuk contoh khusus, misalkan Anda mengunduh file MP3 dengan $F = 32$ juta bit, server memiliki kecepatan transmisi $R_s = 2$ Mbps, dan Anda memiliki tautan akses $R_c = 1$ Mbps. Waktu yang diperlukan untuk mentransfer file adalah 32 detik. Tentu saja, ekspresi untuk throughput dan waktu transfer ini hanyalah perkiraan, karena tidak memperhitungkan penundaan penyimpanan dan penerusan dan pemrosesan serta masalah protokol.

Gambar 1.19(b) sekarang menunjukkan sebuah jaringan dengan N link antara server dan klien, dengan tingkat transmisi N link menjadi R_1, R_2, \dots, R_N . Menerapkan analisis yang sama seperti untuk jaringan dua-tautan, kami menemukan bahwa throughput untuk transfer file dari server ke klien adalah $\min\{R_1, R_2, \dots, R_N\}$, yang sekali lagi merupakan laju misi trans dari bottleneck menghubungkan sepanjang jalur antara server dan klien.

Sekarang pertimbangkan contoh lain yang dimotivasi oleh Internet saat ini. Gambar 1.20(a) menunjukkan dua sistem akhir, server dan klien, yang terhubung ke jaringan komputer. Pertimbangkan throughput untuk transfer file dari server ke klien. Server terhubung ke jaringan dengan link akses rate R_s dan klien terhubung ke jaringan dengan link akses rate R_c . Sekarang misalkan semua tautan di inti jaringan komunikasi memiliki tingkat transmisi yang sangat tinggi, jauh lebih tinggi daripada R_s dan R_c . Memang, saat ini, inti dari Internet menyediakan sambungan berkecepatan tinggi yang mengalami sedikit kemacetan. Juga anggaplah bahwa satu-satunya bit yang dikirim di seluruh jaringan adalah dari server ke klien. Karena inti dari jaringan komputer seperti pipa lebar dalam contoh ini, kecepatan bit dapat mengalir

46 BAB 1

• JARINGAN KOMPUTER DAN INTERNET



Gambar 1.20 Throughput end-to-end: (a) Klien mendownload file dari server; (b) 10 klien mengunduh dengan 10 server

dari sumber ke tujuan lagi minimum Rs dan Rc , yaitu, $\text{throughput} = \min\{Rs, Rc\}$. Oleh karena itu, faktor pembatas throughput di Internet saat ini biasanya adalah jaringan akses.

Sebagai contoh terakhir, perhatikan Gambar 1.20(b) di mana terdapat 10 server dan 10 klien yang terhubung ke inti jaringan komputer. Dalam contoh ini, ada 10 unduhan simultan yang terjadi, melibatkan 10 pasang klien-server. Misalkan 10 unduhan ini adalah satu-satunya lalu lintas di jaringan saat ini. Seperti yang ditunjukkan pada gambar, ada tautan di inti yang dilalui oleh 10 unduhan.

Nyatakan R untuk laju transmisi dari tautan ini R . Misalkan semua tautan akses server memiliki laju yang sama Rs , selain tautan ini. Klien tentu tidak jauh lebih besar dari umum link rate R —jauh lebih besar dari Rs , Rc , dan R . Sekarang kita bertanya, apa throughput dari download? Jelas, ~~link rate betar daripada R , Rc dan Rs karena~~ throughput untuk setiap sekali lagi akan menjadi $\min\{Rs, Rc\}$. Tetapi bagaimana jika laju tautan umum memiliki urutan yang sama dengan Rs dan Rc ? Apa yang akan menjadi throughput dalam kasus ini? Mari kita lihat secara spesifik

contoh. Misalkan $Rs = 2 \text{ Mbps}$, $Rc = 1 \text{ Mbps}$, $R = 5 \text{ Mbps}$, dan tautan umum membagi kecepatan transmisinya secara merata di antara 10 unduhan. Kemudian hambatan untuk setiap unduhan tidak lagi di jaringan akses, tetapi sekarang menjadi tautan bersama di inti, yang hanya menyediakan setiap unduhan dengan throughput 500 kbps.

Dengan demikian, throughput end-to-end untuk setiap unduhan kini dikurangi menjadi 500 kbps.

Contoh pada Gambar 1.19 dan Gambar 1.20(a) menunjukkan bahwa throughput tergantung pada tingkat transmisi dari link dimana data mengalir. Kami melihat bahwa ketika tidak ada lalu lintas lain yang mengintervensi, throughput dapat dengan mudah didekati sebagai laju transmisi minimum di sepanjang jalur antara sumber dan tujuan. Contoh pada Gambar 1.20(b) menunjukkan bahwa secara umum throughput tidak hanya bergantung pada laju transmisi dari link di sepanjang jalur, tetapi juga pada lalu lintas yang mengintervensi. Secara khusus, link dengan kecepatan transmisi yang tinggi mungkin tetap menjadi link bottleneck untuk transfer file jika banyak aliran data lainnya juga melewati link tersebut. Kami akan memeriksa throughput di jaringan komputer lebih dekat dalam masalah pekerjaan rumahan dan di bab selanjutnya.

1.5 Lapisan Protokol dan Model Layanannya

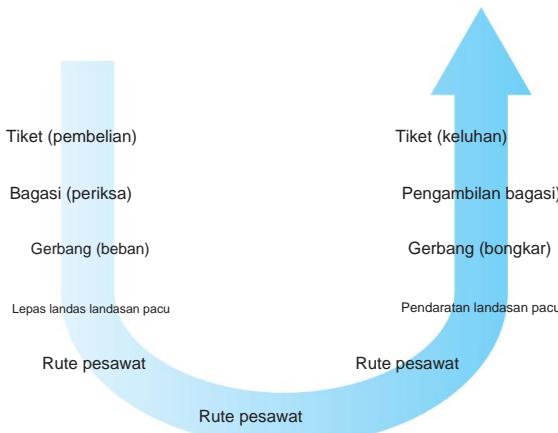
Dari diskusi kita sejauh ini, terlihat jelas bahwa Internet adalah sistem yang sangat rumit. Kita telah melihat bahwa ada banyak bagian di Internet: banyak aplikasi dan protokol, berbagai jenis sistem akhir, sakelar paket, dan berbagai jenis media tingkat tautan. Mengingat kerumitan yang sangat besar ini, apakah ada harapan untuk mengatur arsitektur jaringan, atau setidaknya diskusi kita tentang arsitektur jaringan?

Untungnya, jawaban untuk kedua pertanyaan itu adalah ya.

1.5.1 Arsitektur Berlapis

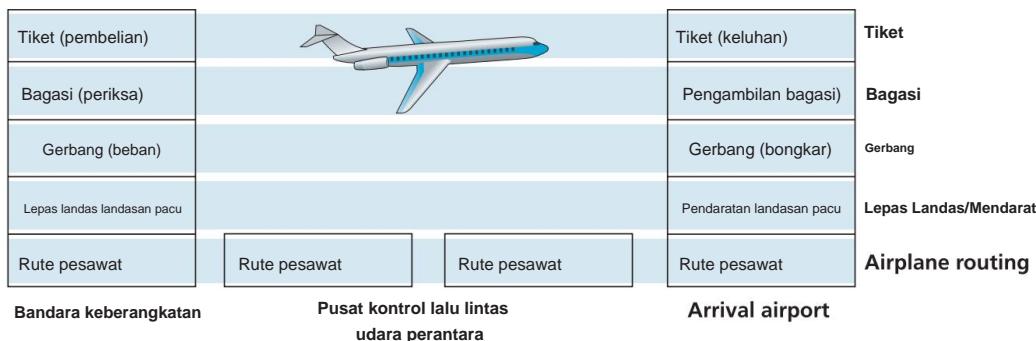
Sebelum mencoba mengatur pemikiran kita tentang arsitektur Internet, mari kita lihat analogi manusia. Sebenarnya, kita berurusan dengan sistem yang rumit sepanjang waktu dalam kehidupan kita sehari-hari. Bayangkan jika seseorang meminta Anda menjelaskan, misalnya, sistem penerbangan. Bagaimana Anda menemukan struktur untuk menggambarkan sistem kompleks ini yang memiliki agen tiket, pemeriksa bagasi, petugas gerbang, pilot, pesawat terbang, kontrol lalu lintas udara, dan sistem rute penerbangan di seluruh dunia? Salah satu cara untuk mendeskripsikan sistem ini mungkin dengan mendeskripsikan rangkaian tindakan yang Anda lakukan (atau tindakan lain yang dilakukan untuk Anda) saat Anda terbang dengan maskapai penerbangan. Anda membeli tiket, memeriksa tas, pergi ke gerbang, dan akhirnya dimuat ke pesawat. Pesawat lepas landas dan diarahkan ke tujuannya. Setelah pesawat Anda mendarat, Anda turun di gerbang dan mengambil tas Anda. Jika perjalannya buruk, Anda mengeluh tentang penerbangan ke agen tiket (tidak mendapatkan apa-apa atas usaha Anda). Skenario ini ditunjukkan pada Gambar 1.21.

Sudah, kita bisa melihat beberapa analogi di sini dengan jaringan komputer: Anda dikirim dari sumber ke tujuan oleh maskapai; sebuah paket dikirim dari

48 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****Gambar 1.21** Melakukan perjalanan pesawat: tindakan

host sumber ke host tujuan di Internet. Tapi ini bukan analogi yang kita kejar. Kami sedang mencari beberapa *struktur* pada Gambar 1.21. Melihat Gambar 1.21, kami mencatat bahwa ada fungsi tiket di setiap ujungnya; ada juga fungsi bagasi untuk penumpang yang sudah ditilang, dan fungsi gerbang untuk penumpang yang sudah ditilang dan yang sudah diperiksa bagasinya. Untuk penumpang yang sudah berhasil melewati gate (yaitu penumpang yang sudah ditilang, pemeriksaan bagasi, dan melewati gate), terdapat fungsi lepas landas dan mendarat, dan selama dalam penerbangan terdapat fungsi routing pesawat. Ini menunjukkan bahwa kita dapat melihat fungsionalitas pada Gambar 1.21 secara *horizontal*, seperti yang ditunjukkan pada Gambar 1.22.

Gambar 1.22 telah membagi fungsionalitas penerbangan menjadi beberapa lapisan, memberikan kerangka kerja di mana kita dapat mendiskusikan perjalanan penerbangan. Perhatikan bahwa setiap lapisan, digabungkan dengan

**Gambar 1.22** Pelapisan horizontal dari fungsi penerbangan

lapisan di bawahnya, mengimplementasikan beberapa fungsi, beberapa *layanan*. Pada lapisan tiket dan di bawahnya, transfer maskapai-counter-to-airline-counter seseorang dilakukan.

Pada lapisan bagasi dan di bawahnya, transfer bagasi-check-to-bagasi-klaim seseorang dan tas dilakukan.

Perhatikan bahwa lapisan bagasi menyediakan layanan ini hanya untuk orang yang sudah memiliki tiket. Pada lapisan gerbang, transfer keberangkatan-gerbang-ke-kedatangan seseorang dan tas dilakukan. Pada lapisan lepas landas/pendaratan, transfer orang dari landasan pacu ke landasan pacu dan tasnya dilakukan. Setiap lapisan menyediakan layanannya dengan (1) melakukan tindakan tertentu di dalam lapisan itu (misalnya, di lapisan gerbang, memuat dan menurunkan orang dari pesawat terbang) dan dengan (2) menggunakan layanan lapisan langsung di bawahnya (misalnya, di lapisan gerbang, menggunakan layanan transfer penumpang landasan pacu ke landasan dari lapisan lepas landas/pendaratan).

Arsitektur berlapis memungkinkan kita untuk membahas bagian spesifik yang terdefinisi dengan baik dari sistem yang besar dan kompleks. Penyederhanaan ini sendiri memiliki nilai yang cukup besar dengan menyediakan modularitas, membuatnya lebih mudah untuk mengubah implementasi layanan yang disediakan oleh layer. Selama lapisan menyediakan layanan yang sama ke lapisan di atasnya, dan menggunakan layanan yang sama dari lapisan di bawahnya, sisa sistem tetap tidak berubah saat penerapan lapisan diubah.

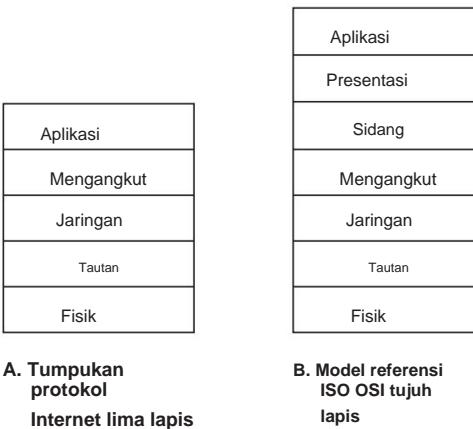
(Perhatikan bahwa mengubah penerapan layanan sangat berbeda dengan mengubah layanan itu sendiri!) Misalnya, jika fungsi gerbang diubah (misalnya, agar orang naik dan turun berdasarkan ketinggian), sisa sistem penerbangan akan tetap tidak berubah karena lapisan gerbang masih memberikan fungsi yang sama (mengangkut dan menurunkan orang); itu hanya mengimplementasikan fungsi itu dengan cara yang berbeda setelah perubahan. Untuk sistem besar dan kompleks yang terus diperbarui, kemampuan untuk mengubah implementasi layanan tanpa memengaruhi komponen lain dari sistem adalah keuntungan penting lainnya dari layering.

Pelapisan Protokol

Tapi cukup tentang maskapai penerbangan. Sekarang mari kita alihkan perhatian kita ke protokol jaringan. Untuk menyediakan struktur pada desain protokol jaringan, perancang jaringan mengatur protokol—and perangkat keras dan perangkat lunak jaringan yang mengimplementasikan protokol—berlapis- **lapis**. Setiap protokol milik salah satu lapisan, seperti setiap fungsi dalam arsitektur penerbangan pada Gambar 1.22 milik lapisan. Kami sekali lagi tertarik pada **layanan** yang ditawarkan lapisan ke lapisan di atasnya — yang disebut **model layanan** lapisan. Sama seperti dalam kasus contoh maskapai kami, setiap lapisan menyediakan layanannya dengan (1) melakukan tindakan tertentu dalam lapisan itu dan dengan (2) menggunakan layanan lapisan langsung di bawahnya. Misalnya, layanan yang disediakan oleh lapisan n dapat mencakup pengiriman pesan yang andal dari satu ujung jaringan ke ujung lainnya.

Ini mungkin diimplementasikan dengan menggunakan layanan pengiriman pesan tepi-ke-tepi yang tidak dapat diandalkan dari lapisan n , dan menambahkan fungsionalitas lapisan n untuk mendekripsi dan mengirim ulang pesan yang hilang.

Lapisan protokol dapat diimplementasikan dalam perangkat lunak, perangkat keras, atau kombinasi keduanya. Protokol lapisan aplikasi—seperti HTTP dan SMTP—hampir

50 BAB 1**• JARINGAN KOMPUTER DAN INTERNET****Gambar 1.23** Tumpukan protokol Internet (a) dan model referensi OSI (b)

selalu diimplementasikan dalam perangkat lunak pada sistem akhir; begitu juga protokol transport-layer. Karena lapisan fisik dan lapisan data link bertanggung jawab untuk menangani komunikasi melalui tautan tertentu, mereka biasanya diimplementasikan dalam kartu antarmuka jaringan (misalnya, kartu antarmuka Ethernet atau WiFi) yang terkait dengan tautan yang diberikan. Lapisan jaringan sering kali merupakan implementasi campuran dari perangkat keras dan perangkat lunak. Perhatikan juga bahwa sama seperti fungsi dalam arsitektur berlapis maskapai didistribusikan di antara berbagai bandara dan pusat kontrol penerbangan yang membentuk sistem, demikian pula protokol lapisan n didistribusikan *di antara* sistem akhir, sakelar paket, dan komponen lain yang membentuk jaringan. Artinya, sering ada bagian dari protokol layer n di setiap komponen jaringan ini.

Pelapisan protokol memiliki keunggulan konseptual dan struktural [RFC 3439]. Seperti yang telah kita lihat, layering menyediakan cara terstruktur untuk membahas komponen sistem. Modularitas membuatnya lebih mudah untuk memperbarui komponen sistem. Kami menyebutkan, bagaimanapun, bahwa beberapa peneliti dan insinyur jaringan sangat menentang layering [Wakeman 1992]. Salah satu kelemahan potensial dari pelapisan adalah bahwa satu lapisan dapat menduplikasi fungsionalitas lapisan bawah. Sebagai contoh, banyak tumpukan protokol menyediakan pemulihian kesalahan pada basis per-tautan dan basis end-to-end. Kelemahan potensial kedua adalah fungsionalitas pada satu lapisan mungkin memerlukan informasi (misalnya, nilai stempel waktu) yang hanya ada di lapisan lain; ini melanggar tujuan pemisahan lapisan.

Ketika disatukan, protokol dari berbagai lapisan disebut **tumpukan protokol**. Tumpukan protokol Internet terdiri dari lima lapisan: lapisan fisik, tautan, jaringan, transportasi, dan aplikasi, seperti yang ditunjukkan pada Gambar 1.23(a). Jika Anda memeriksa Daftar Isi, Anda akan melihat bahwa kami telah menyusun buku ini secara kasar menggunakan lapisan tumpukan protokol Internet. Kami mengambil **pendekatan top-down**, pertama menutupi lapisan aplikasi dan kemudian melanjutkan ke bawah.

Lapisan Aplikasi

Lapisan aplikasi adalah tempat aplikasi jaringan dan protokol lapisan aplikasinya berada. Lapisan aplikasi Internet mencakup banyak protokol, seperti protokol HTTP (yang menyediakan permintaan dan transfer dokumen Web), SMTP (yang menyediakan transfer pesan email), dan FTP (yang menyediakan transfer file antara sistem dua ujung). Kita akan melihat bahwa fungsi jaringan tertentu, seperti terjemahan nama ramah manusia untuk sistem akhir Internet seperti www.ietf.org ke alamat jaringan 32-bit, juga dilakukan dengan bantuan protokol lapisan aplikasi khusus. , yaitu, sistem nama domain (DNS). Kita akan melihat di Bab 2 bahwa sangat mudah untuk membuat dan menerapkan protokol lapisan aplikasi baru kita sendiri.

Protokol lapisan aplikasi didistribusikan melalui beberapa sistem akhir, dengan aplikasi di satu sistem akhir menggunakan protokol untuk bertukar paket informasi dengan aplikasi di sistem akhir lainnya. Kami akan merujuk paket informasi ini pada lapisan aplikasi sebagai **pesan**.

Lapisan Transportasi

Lapisan transport Internet mengangkut pesan lapisan aplikasi antara titik akhir aplikasi. Di Internet ada dua protokol transport, TCP dan UDP, keduanya dapat mengangkut pesan lapisan aplikasi. TCP menyediakan layanan berorientasi koneksi untuk aplikasinya. Layanan ini mencakup jaminan pengiriman pesan lapisan aplikasi ke tujuan dan kontrol aliran (yaitu, pencocokan kecepatan pengirim/penerima). TCP juga memecah pesan-pesan panjang menjadi segmen-segmen yang lebih pendek dan menyediakan mekanisme kontrol kongesti, sehingga sumber membatasi laju transmisinya ketika jaringan padat. Protokol UDP menyediakan layanan tanpa koneksi ke aplikasinya. Ini adalah layanan tanpa embel-embel yang tidak memberikan keandalan, tanpa kontrol aliran, dan tanpa kontrol kemacetan. Dalam buku ini, kita akan mengacu pada paket transport-layer sebagai sebuah **segmen**.

Lapisan Jaringan

Lapisan jaringan Internet bertanggung jawab untuk memindahkan paket lapisan jaringan yang dikenal sebagai **datagram** dari satu host ke host lainnya. Protokol lapisan transpor Internet (TCP atau UDP) di host sumber melewati segmen lapisan transpor dan alamat tujuan ke lapisan jaringan, sama seperti Anda memberikan surat dengan alamat tujuan kepada layanan pos. Lapisan jaringan kemudian menyediakan layanan pengiriman segmen ke lapisan transport di host tujuan.

Lapisan jaringan Internet mencakup Protokol IP yang terkenal, yang mendefinisikan bidang-bidang dalam datagram serta bagaimana sistem akhir dan router bekerja di bidang-bidang ini. Hanya ada satu protokol IP, dan semua komponen Internet yang memiliki lapisan kerja jaringan harus menjalankan protokol IP. Lapisan jaringan Internet juga berisi protokol perutean yang menentukan rute yang diambil oleh datagram antara sumber dan

52 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

tujuan. Internet memiliki banyak protokol perutean. Seperti yang kita lihat di Bagian 1.3, Internet adalah jaringan dari jaringan, dan di dalam jaringan, administrator jaringan dapat menjalankan protokol perutean yang diinginkan. Meskipun lapisan jaringan berisi protokol IP dan banyak protokol perutean, sering kali hanya disebut sebagai lapisan IP, yang mencerminkan fakta bahwa IP adalah lem yang mengikat Internet bersama.

Lapisan Tautan

Lapisan jaringan Internet merutekan datagram melalui serangkaian router antara sumber dan tujuan. Untuk memindahkan paket dari satu node (host atau router) ke node berikutnya dalam rute, network layer bergantung pada layanan dari link layer. Secara khusus, pada setiap node, lapisan jaringan meneruskan datagram ke lapisan tautan, yang mengirimkan datagram ke node berikutnya di sepanjang rute. Pada node berikutnya, lapisan tautan meneruskan datagram ke lapisan jaringan.

Layanan yang disediakan oleh lapisan tautan bergantung pada protokol lapisan tautan khusus yang digunakan di atas tautan. Misalnya, beberapa protokol lapisan tautan menyediakan pengiriman yang andal, dari node transmisi, melalui satu tautan, ke node penerima. Perhatikan bahwa layanan pengiriman andal ini berbeda dari layanan pengiriman TCP yang andal, yang menyediakan pengiriman andal dari satu sistem akhir ke sistem lainnya. Contoh protokol lapisan tautan termasuk Ethernet, WiFi, dan protokol jaringan akses kabel DOCSIS. Karena datagram biasanya perlu melintasi beberapa tautan untuk melakukan perjalanan dari sumber ke tujuan, sebuah datagram dapat ditangani oleh protokol lapisan tautan yang berbeda pada tautan yang berbeda di sepanjang rutenya. Sebagai contoh, sebuah datagram dapat ditangani oleh Ethernet pada satu link dan oleh PPP pada link berikutnya. Lapisan jaringan akan menerima layanan yang berbeda dari masing-masing protokol lapisan tautan yang berbeda. Dalam buku ini, kami akan mengacu pada paket lapisan tautan sebagai **bingkai**.

Lapisan fisik

Sementara tugas lapisan tautan adalah memindahkan seluruh bingkai dari satu elemen jaringan ke elemen jaringan yang berdekatan, tugas lapisan fisik adalah memindahkan bit individu *di* dalam bingkai dari satu simpul ke simpul berikutnya. Protokol pada lapisan ini sekali lagi bergantung pada tautan dan selanjutnya bergantung pada media transmisi sebenarnya dari tautan tersebut (misalnya, kabel tembaga twisted-pair, serat optik mode tunggal). Sebagai contoh, Ethernet memiliki banyak protokol lapisan fisik: satu untuk kabel tembaga twisted-pair, satu lagi untuk kabel koaksial, satu lagi untuk serat, dan seterusnya. Dalam setiap kasus, sedikit dipindahkan melintasi tautan dengan cara yang berbeda.

Model OSI

Setelah membahas tumpukan protokol Internet secara rinci, kami harus menyebutkan bahwa itu bukan satu-satunya tumpukan protokol. Secara khusus, pada akhir 1970-an, Organisasi Internasional untuk Standardisasi (ISO) mengusulkan agar jaringan komputer

disusun sekitar tujuh lapisan, yang disebut model Open System Interconnection (OSI) [ISO 2012]. Model OSI terbentuk ketika protokol yang akan menjadi protokol Internet masih dalam masa pertumbuhan, dan hanyalah salah satu dari banyak rangkaian protokol berbeda yang sedang dikembangkan; pada kenyataannya, penemu model OSI asli mungkin tidak memikirkan Internet saat membuatnya. Namun demikian, dimulai pada akhir 1970-an, banyak pelatihan dan kursus universitas mengikuti mandat ISO dan menyelenggarakan kursus di sekitar model tujuh lapis. Karena dampak awalnya pada pendidikan jaringan, model tujuh lapis terus bertahan di beberapa buku teks dan kursus pelatihan jaringan.

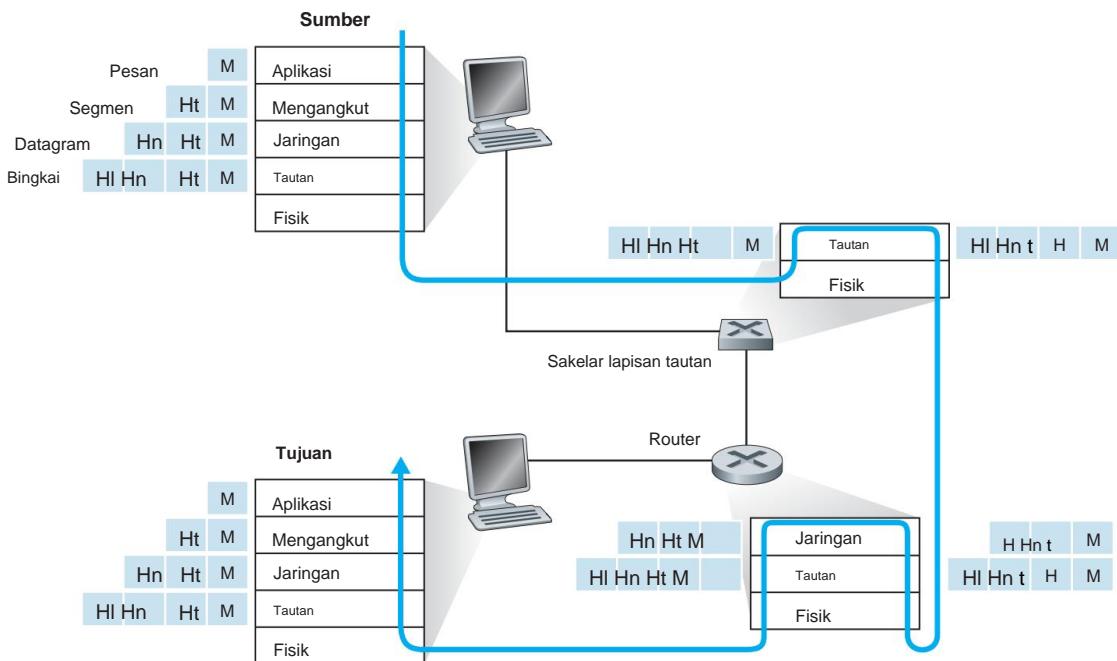
Tujuh lapisan model referensi OSI, ditunjukkan pada Gambar 1.23(b), adalah: lapisan aplikasi, lapisan presentasi, lapisan sesi, lapisan transport, lapisan jaringan, lapisan data link, dan lapisan fisik. Fungsionalitas dari lima lapisan ini kira-kira sama dengan rekan Internet mereka yang bernama serupa. Jadi, mari pertimbangkan dua lapisan tambahan yang ada dalam model referensi OSI—lapisan presentasi dan lapisan sesi. Peran lapisan presentasi adalah untuk menyediakan layanan yang memungkinkan aplikasi yang berkomunikasi untuk menginterpretasikan arti dari data yang dipertukarkan.

Layanan ini termasuk kompresi data dan enkripsi data (yang cukup jelas) serta deskripsi data (yang, seperti yang akan kita lihat di Bab 9, membebaskan aplikasi dari kekhawatiran tentang format internal di mana data direpresentasikan/disimpan—format yang mungkin berbeda dari satu komputer ke komputer lainnya). Lapisan sesi menyediakan batasan dan sinkronisasi pertukaran data, termasuk sarana untuk membangun pos pemeriksaan dan skema pemulihian.

Fakta bahwa Internet kekurangan dua lapisan yang ditemukan dalam model referensi OSI menimbulkan beberapa pertanyaan menarik: Apakah layanan yang disediakan oleh lapisan ini tidak penting? Bagaimana jika aplikasi *membutuhkan* salah satu dari layanan ini? Jawaban Internet untuk kedua pertanyaan ini sama—terserah pengembang aplikasi. Terserah pengembang aplikasi untuk memutuskan apakah suatu layanan itu penting, dan jika layanan itu *penting*, terserah pengembang aplikasi untuk membangun fungsionalitas itu ke dalam aplikasi.

1.5.2 Enkapsulasi

Gambar 1.24 menunjukkan jalur fisik di mana data mengambil tumpukan protokol sistem akhir pengiriman, naik turun tumpukan protokol dari switch lapisan tautan dan router, dan kemudian naik tumpukan protokol pada sistem akhir penerima. Seperti yang akan kita bahas nanti di buku ini, router dan switch lapisan tautan keduanya adalah switch paket. Mirip dengan sistem akhir, router dan sakelar lapisan tautan mengatur perangkat keras dan perangkat lunak jaringan mereka ke dalam lapisan. Tetapi router dan sakelar lapisan tautan tidak mengimplementasikan *semua* lapisan dalam tumpukan protokol; mereka biasanya hanya menerapkan lapisan bawah. Seperti yang ditunjukkan pada Gambar 1.24, link-layer switch mengimplementasikan layer 1 dan 2; router mengimplementasikan lapisan 1 sampai 3. Ini berarti, misalnya, router Internet mampu mengimplementasikan protokol IP (protokol lapisan 3), sementara sakelar lapisan tautan tidak. Kita akan melihat nanti bahwa meskipun sakelar lapisan tautan tidak mengenali alamat IP, mereka



Gambar 1.24 Host, router, dan sakelar lapisan tautan; masing-masing berisi kumpulan lapisan yang berbeda, yang mencerminkan perbedaan fungsionalitasnya

mampu mengenali alamat layer 2, seperti alamat Ethernet. Perhatikan bahwa host mengimplementasikan kelima lapisan; ini konsisten dengan pandangan bahwa arsitektur Internet menempatkan sebagian besar kerumitan di tepi jaringan.

Gambar 1.24 juga mengilustrasikan konsep **enkapsulasi yang penting**. Di host pengirim, **pesan lapisan aplikasi** (M pada Gambar 1.24) diteruskan ke lapisan transport. Dalam kasus yang paling sederhana, lapisan transport mengambil pesan dan menambahkan informasi tambahan (disebut informasi header lapisan transport, *Ht* pada Gambar 1.24) yang akan digunakan oleh lapisan transport sisi penerima. Pesan application-layer dan informasi header transport-layer bersama-sama membentuk **segmen transport-layer**. Dengan demikian, segmen lapisan transport mengenkapsulasi pesan lapisan aplikasi. Informasi tambahan mungkin termasuk informasi yang memungkinkan lapisan transport sisi penerima untuk mengirimkan pesan ke aplikasi yang sesuai, dan bit deteksi kesalahan yang memungkinkan penerima untuk menentukan apakah bit dalam pesan telah diubah dalam rute. Lapisan transport kemudian meneruskan segmen ke lapisan jaringan, yang menambahkan informasi header lapisan jaringan (*Hn* pada Gambar 1.24) seperti alamat sistem akhir sumber dan tujuan,

membuat **datagram lapisan jaringan**. Datagram kemudian diteruskan ke lapisan tautan, yang (tentu saja!) akan menambahkan informasi tajuk lapisan tautannya sendiri dan membuat bingkai **lapisan tautan**. Jadi, kita melihat bahwa pada setiap lapisan, sebuah paket memiliki dua jenis field: field header dan **field payload**. Muatan biasanya berupa paket dari lapisan di atasnya.

Analogi yang berguna di sini adalah pengiriman memo antar kantor dari satu kantor cabang korporat ke kantor cabang lainnya melalui layanan pos umum. Misalkan Alice yang berada di satu kantor cabang ingin mengirim memo ke Bob yang berada di kantor cabang lain. *Memo* itu analog dengan *pesan lapisan aplikasi*. Alice memasukkan memo itu ke dalam amplop antar-kantor dengan nama Bob dan departemen tertulis di bagian depan amplop. Amplop *antar-kantor* serupa dengan *segmen lapisan transportasi*—yang berisi informasi header (nama Bob dan nomor departemen) dan merangkum pesan lapisan aplikasi (memo). Ketika ruang surat kantor cabang pengirim menerima amplop antarkantor, amplop antarkantor dimasukkan ke dalam amplop lain lagi, yang sesuai untuk pengiriman melalui layanan pos umum. Ruang surat pengirim juga menuliskan alamat pos kantor cabang pengirim dan penerima pada amplop pos. Di sini, *amplop pos* dianalogikan dengan *datagram*—ia merangkum segmen lapisan transpor (amplop antar kantor), yang merangkum pesan asli (memo). Layanan pos mengirimkan amplop pos ke ruang surat kantor cabang penerima. Di sana, proses de-enkapsulasi dimulai. Ruang surat mengekstrak memo antar kantor dan meneruskannya ke Bob. Akhirnya, Bob membuka amplop dan mengeluarkan memo itu.

Proses enkapsulasi bisa lebih kompleks dari yang dijelaskan di atas. Misalnya, sebuah pesan besar dapat dibagi menjadi beberapa segmen lapisan transport (yang masing-masing dapat dibagi menjadi beberapa datagram lapisan jaringan). Di ujung penerima, segmen seperti itu kemudian harus direkonstruksi dari datagram penyusunnya.

1.6 Jaringan Diserang

Internet telah menjadi misi penting bagi banyak institusi saat ini, termasuk perusahaan besar dan kecil, universitas, dan lembaga pemerintah. Banyak orang juga mengandalkan Internet untuk banyak aktivitas profesional, sosial, dan pribadi mereka.

Namun di balik semua utilitas dan kegembiraan ini, ada sisi gelap, sisi di mana "orang jahat" berusaha mengacaukan kehidupan kita sehari-hari dengan merusak komputer yang terhubung ke Internet, melanggar privasi kita, dan membuat layanan Internet yang kita gunakan tidak dapat dioperasikan. bergantung.

Bidang keamanan jaringan adalah tentang bagaimana orang jahat dapat menyerang jaringan komputer dan tentang bagaimana kita, calon ahli dalam jaringan komputer, dapat

mempertahankan jaringan dari serangan tersebut, atau lebih baik lagi, merancang arsitektur baru yang kebal terhadap serangan semacam itu. Mengingat frekuensi dan ragam serangan yang ada serta ancaman serangan baru dan masa depan yang lebih merusak, keamanan jaringan telah menjadi topik sentral dalam bidang jaringan komputer.

Salah satu fitur dari buku teks ini adalah membawa masalah keamanan jaringan ke permukaan.

Karena kami belum memiliki keahlian dalam jaringan komputer dan protokol Internet, kami akan mulai di sini dengan mensurvei beberapa masalah terkait keamanan yang lebih lazim saat ini. Ini akan membangkitkan selera kita untuk diskusi yang lebih substansial di bab-bab mendatang. Jadi kita mulai di sini hanya dengan bertanya, apa yang salah?

Bagaimana jaringan komputer rentan? Apa saja jenis serangan yang lebih umum saat ini?

Orang jahat dapat memasukkan malware ke host Anda melalui Internet

Kami menyambungkan perangkat ke Internet karena kami ingin menerima/mengirim data dari/ke Internet. Ini mencakup semua jenis hal bagus, termasuk halaman Web, pesan email, MP3, panggilan telepon, video langsung, hasil mesin pencari, dan seterusnya.

Namun, sayangnya, seiring dengan semua hal bagus itu, muncullah hal-hal berbahaya—secara kolektif dikenal sebagai **malware**—yang juga dapat masuk dan menginfeksi perangkat kita.

Setelah perangkat lunak perusak menginfeksi perangkat kami, ia dapat melakukan segala macam hal licik, termasuk menghapus file kami; menginstal spyware yang mengumpulkan informasi pribadi kita, seperti nomor jaminan sosial, kata sandi, dan penekanan tombol, lalu mengirimkannya (melalui jaringan Inter, tentu saja!) kembali ke orang jahat. Host kami yang dikompromikan juga dapat terdaftar dalam jaringan ribuan perangkat yang sama-sama dikompromikan, secara kolektif dikenal sebagai **botnet**, yang dikendalikan dan dimanfaatkan oleh orang-orang jahat untuk distribusi email spam atau serangan denial-of-service yang didistribusikan (segera akan dibahas) terhadap host yang ditargetkan.

Sebagian besar malware di luar sana saat ini **mereplikasi dirinya sendiri**: setelah menginfeksi satu host, dari host tersebut ia mencari akses ke host lain melalui Internet, dan dari host yang baru terinfeksi, ia mencari akses ke lebih banyak host. Dengan cara ini, malware yang mereplikasi diri dapat menyebar dengan cepat secara eksponensial. Malware dapat menyebar dalam bentuk virus atau worm. **Virus** adalah malware yang memerlukan beberapa bentuk interaksi pengguna untuk menginfeksi perangkat pengguna. Contoh klasiknya adalah lampiran email yang berisi kode berbahaya yang dapat dieksekusi. Jika pengguna menerima dan membuka lampiran tersebut, pengguna secara tidak sengaja menjalankan malware di perangkat. Biasanya, virus email semacam itu menggandakan diri: setelah dieksekusi, virus dapat mengirim pesan identik dengan lampiran berbahaya yang identik, misalnya, ke setiap penerima di buku alamat pengguna. **Cacing** adalah malware yang dapat memasuki perangkat tanpa interaksi pengguna yang eksplisit. Misalnya, pengguna mungkin menjalankan aplikasi jaringan yang rentan tempat penyerang dapat mengirimkan malware. Dalam beberapa kasus, tanpa campur tangan pengguna, aplikasi dapat menerima malware dari Internet dan

menjalankannya, membuat worm. Worm di perangkat yang baru terinfeksi kemudian memindai Internet, mencari host lain yang menjalankan aplikasi jaringan rentan yang sama. Ketika menemukan host lain yang rentan, ia mengirimkan salinan dirinya ke host tersebut.

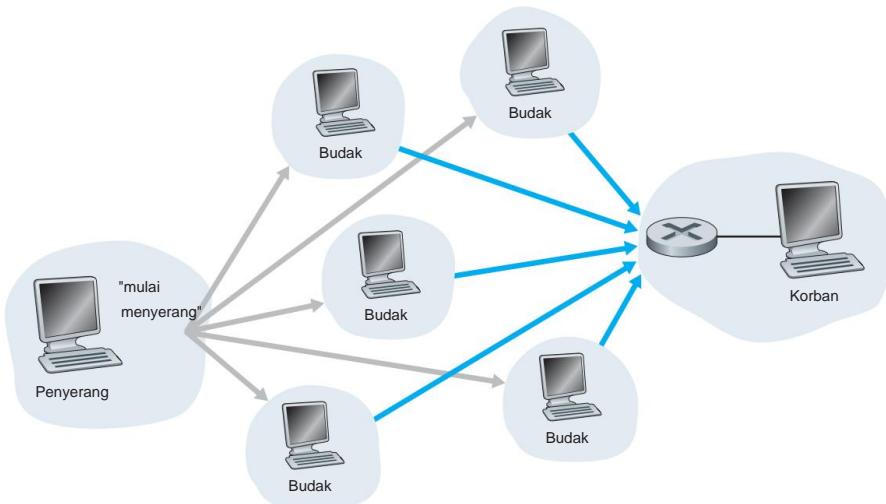
Saat ini, malware, menyebar luas dan mahal untuk dilawan. Saat Anda mengerjakan buku teks ini, kami mendorong Anda untuk memikirkan pertanyaan berikut: Apa yang dapat dilakukan perancang jaringan komputer untuk melindungi perangkat yang tersambung ke Internet dari serangan malware?

Orang jahat dapat menyerang server dan infrastruktur jaringan

Ancaman keamanan kelas luas lainnya dikenal sebagai **serangan denial-of-service (DoS)**. Seperti namanya, serangan DoS membuat jaringan, host, atau infrastruktur lain tidak dapat digunakan oleh pengguna yang sah. Server web, server e-mail, server DNS (dibahas di Bab 2), dan jaringan institusi semuanya dapat terkena serangan DoS. Serangan DoS Internet sangat umum, dengan ribuan serangan DoS terjadi setiap tahun [Moore 2001; Mirkovic 2005]. Sebagian besar serangan DoS Internet termasuk dalam salah satu dari tiga kategori:

- *Serangan kerentanan.* Ini melibatkan pengiriman beberapa pesan yang dibuat dengan baik ke aplikasi yang rentan atau sistem operasi yang berjalan pada host yang ditargetkan. Jika urutan paket yang tepat dikirim ke aplikasi atau sistem operasi yang rentan, layanan dapat berhenti atau, lebih buruk lagi, host dapat macet.
- *Bandwidth banjir.* Penyerang mengirimkan banjir paket ke host yang ditargetkan—begitu banyak paket sehingga tautan akses target tersumbat, mencegah paket yang sah mencapai server.
- *Sambungan banjir.* Penyerang membuat sejumlah besar koneksi TCP setengah terbuka atau terbuka penuh (koneksi TCP dibahas di Bab 3) di host target. Tuan rumah bisa menjadi sangat macet dengan koneksi palsu ini sehingga berhenti menerima koneksi yang sah.

Sekarang mari kita jelajahi serangan bandwidth-flooding lebih terinci. Mengingat diskusi analisis penundaan dan kerugian kami di Bagian 1.4.2, terbukti bahwa jika server memiliki tingkat akses R bps, maka penyerang perlu mengirimkan lalu lintas dengan kecepatan sekitar R bps untuk menyebabkan kerusakan. Jika R sangat besar, satu sumber serangan mungkin tidak dapat menghasilkan lalu lintas yang cukup untuk merusak server. Selain itu, jika semua lalu lintas berasal dari satu sumber, router upstream dapat mendeteksi serangan dan memblokir semua lalu lintas dari sumber tersebut sebelum lalu lintas mendekati server. Dalam serangan **DoS (DDoS) terdistribusi**, yang diilustrasikan pada Gambar 1.25, penyerang mengontrol beberapa sumber dan mengatur lalu lintas ledakan sumber ke target. Dengan pendekatan ini, tingkat lalu lintas agregat di semua sumber yang dikendalikan harus kira-kira R untuk melumpuhkannya



Gambar 1.25 Serangan denial-of-service terdistribusi

melayani. Serangan DDoS yang memanfaatkan botnet dengan ribuan host adalah hal yang umum terjadi saat ini [Mirkovic 2005]. Serangan DDoS jauh lebih sulit untuk dideteksi dan dipertahankan daripada serangan DoS dari satu host.

Kami mendorong Anda untuk mempertimbangkan pertanyaan berikut saat Anda mempelajari buku ini: Apa yang dapat dilakukan oleh perancang jaringan komputer untuk mempertahankan diri dari serangan DoS? Kita akan melihat bahwa diperlukan pertahanan yang berbeda untuk ketiga jenis serangan DoS.

Orang jahat bisa mengendus paket

Banyak pengguna saat ini mengakses Internet melalui perangkat nirkabel, seperti laptop yang terhubung ke WiFi atau perangkat genggam dengan koneksi Internet seluler (dibahas di Bab 6). Sementara akses Internet di mana-mana sangat nyaman dan memungkinkan aplikasi baru yang luar biasa untuk pengguna seluler, itu juga menciptakan kerentanan keamanan yang besar—dengan menempatkan penerima pasif di sekitar pemancar nirkabel, penerima tersebut dapat memperoleh salinan dari setiap paket yang ditransmisikan! Paket ini dapat berisi semua jenis informasi sensitif, termasuk kata sandi, nomor jaminan sosial, rahasia dagang, dan pesan pribadi pribadi. Penerima pasif yang merekam salinan setiap paket yang lewat disebut **packet sniffer**.

Sniffer juga dapat digunakan di lingkungan kabel. Di lingkungan siaran kabel, seperti di banyak LAN Ethernet, packet sniffer dapat memperoleh salinan paket siaran yang dikirim melalui LAN. Seperti yang dijelaskan pada Bagian 1.2, teknologi akses kabel juga menyiarakan paket sehingga rentan terhadap penyadapan. Selain itu, orang jahat yang memperoleh akses ke router akses institusi atau tautan akses ke Internet dapat melakukannya

dapat menanam sniffer yang membuat salinan dari setiap paket yang pergi ke/dari organisasi.

Paket yang diendus kemudian dapat dianalisis secara offline untuk informasi sensitif.

Perangkat lunak pengendus paket tersedia secara bebas di berbagai situs Web dan sebagai produk komersial. Profesor yang mengajar kursus jaringan telah dikenal untuk menugaskan latihan laboratorium yang melibatkan penulisan paket-sniffing dan program rekonstruksi data lapisan aplikasi. Memang, lab Wireshark [Wireshark 2012] yang terkait dengan teks ini (lihat pengantar lab Wireshark di akhir bab ini) menggunakan packet sniffer yang persis seperti itu!

Karena packet sniffer bersifat pasif—yaitu, mereka tidak menyuntikkan paket ke saluran—mereka sulit dideteksi. Jadi, ketika kita mengirim paket ke saluran nirkabel, kita harus menerima kemungkinan bahwa orang jahat mungkin merekam salinan paket kita. Seperti yang mungkin sudah Anda duga, beberapa pertahanan terbaik melawan packet sniffing melibatkan kriptografi. Kami akan memeriksa kriptografi yang berlaku untuk keamanan kerja jaringan di Bab 8.

Orang jahat bisa menyamar sebagai seseorang yang Anda percayai

Sangat mudah (*Anda* akan memiliki pengetahuan untuk melakukannya segera setelah Anda membaca teks ini!) untuk membuat paket dengan alamat sumber arbitrer, isi paket, dan alamat tujuan dan kemudian mengirimkan paket buatan tangan ini ke Internet , yang dengan patuh akan meneruskan paket ke tujuannya. Bayangkan penerima yang tidak menaruh curiga (misalkan router Internet) yang menerima paket seperti itu, menganggap alamat sumber (salah) sebagai benar, dan kemudian melakukan beberapa perintah yang disematkan dalam konten paket (katakanlah memodifikasi tabel penerusannya). Kemampuan untuk menyuntikkan paket ke Internet dengan alamat sumber palsu dikenal sebagai **spoofing IP**, dan hanyalah salah satu dari banyak cara di mana satu pengguna dapat menyamar sebagai pengguna lain.

Untuk mengatasi masalah ini, kita memerlukan *autentikasi titik akhir*, yaitu mekanisme yang memungkinkan kita menentukan dengan pasti apakah suatu pesan berasal dari tempat yang menurut kita berasal. Sekali lagi, kami mendorong Anda untuk berpikir tentang bagaimana hal ini dapat dilakukan untuk aplikasi dan protokol jaringan saat Anda mempelajari bab-bab dalam buku ini. Kami akan mengeksplorasi mekanisme otentifikasi titik akhir di Bab 8.

Sebagai penutup bagian ini, ada baiknya mempertimbangkan bagaimana Internet menjadi tempat yang tidak aman. Jawabannya, pada intinya, adalah bahwa Internet pada awalnya dirancang seperti itu, berdasarkan model “sekelompok pengguna yang saling percaya yang terikat pada jaringan transparan” [Blumenthal 2001]—sebuah model di mana (menurut definisi) tidak perlu keamanan. Banyak aspek arsitektur Internet asli yang sangat mencerminkan gagasan saling percaya ini. Misalnya, kemampuan untuk satu pengguna untuk mengirim paket ke pengguna lain adalah default daripada kemampuan yang diminta/diberikan, dan identitas pengguna diambil pada nilai nominal yang dinyatakan, daripada diautentikasi secara default.

Tetapi Internet saat ini tentu saja tidak melibatkan “pengguna yang saling percaya”. Meskipun demikian, pengguna saat ini masih perlu berkomunikasi ketika mereka tidak perlu saling percaya, mungkin ingin berkomunikasi secara anonim, dapat berkomunikasi secara tidak langsung melalui pihak ketiga (misalnya, cache Web, yang akan kita pelajari di Bab 2, atau

60 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

agen pendukung mobilitas, yang akan kita pelajari di Bab 6), dan mungkin tidak mempercayai perangkat keras, perangkat lunak, dan bahkan udara yang mereka gunakan untuk berkomunikasi. Kami sekarang memiliki banyak tantangan terkait keamanan di hadapan kami saat kami melanjutkan buku ini: Kami harus mencari pertahanan terhadap sniffing, penyamaran titik akhir, serangan man-in-the-middle, serangan DDoS, malware, dan banyak lagi. Kita harus ingat bahwa komunikasi di antara pengguna yang saling dipercaya adalah pengecualian, bukan aturannya. Selamat datang di dunia jaringan komputer modern!

1.7 Sejarah Jaringan Komputer dan internet

Bagian 1.1 sampai 1.6 menyajikan ikhtisar teknologi jaringan komputer dan Internet. Anda harus cukup tahu sekarang untuk mengesankan keluarga dan teman Anda! Namun, jika Anda benar-benar ingin menjadi terkenal di pesta koktail berikutnya, Anda harus memercikkan wacana Anda dengan berita menarik tentang sejarah menarik dari jaringan Inter [Segaller 1998].

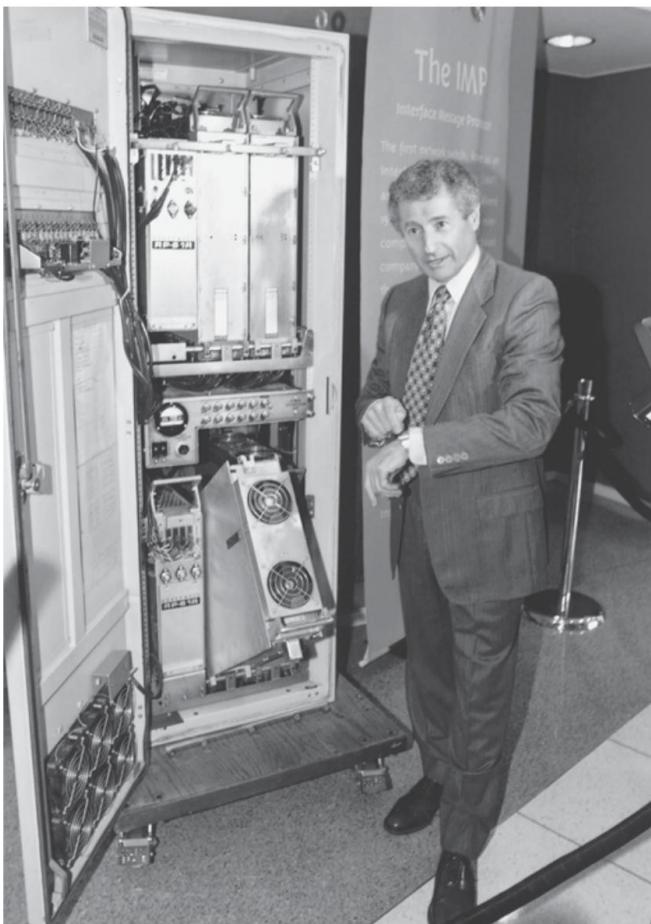
1.7.1 Perkembangan Packet Switching: 1961–1972

Bidang jaringan komputer dan Internet saat ini menelusuri permulaannya kembali ke awal 1960-an, ketika jaringan telepon adalah jaringan komunikasi yang dominan di dunia. Ingat dari Bagian 1.3 bahwa jaringan telepon menggunakan pengalihan sirkuit untuk mengirimkan informasi dari pengirim ke penerima—pilihan yang tepat mengingat bahwa suara ditransmisikan pada kecepatan konstan antara pengirim dan penerima. Mengingat semakin pentingnya komputer pada awal 1960-an dan munculnya komputer timeshared, mungkin wajar untuk mempertimbangkan bagaimana menghubungkan komputer bersama sehingga mereka dapat dibagi di antara pengguna yang tersebar secara geografis. Lalu lintas yang dihasilkan oleh pengguna seperti itu cenderung *meledak*—interval aktivitas, seperti pengiriman perintah ke komputer jarak jauh, diikuti dengan periode tidak aktif saat menunggu balasan atau saat memikirkan respons yang diterima.

Tiga kelompok penelitian di seluruh dunia, masing-masing menyadari pekerjaan orang lain [Leiner 1998], mulai menemukan packet switching sebagai alternatif yang efisien dan kuat untuk circuit switching. Karya pertama yang dipublikasikan tentang teknik packet-switching adalah karya Leonard Kleinrock [Kleinrock 1961; Kleinrock 1964], kemudian menjadi mahasiswa pascasarjana di MIT. Dengan menggunakan teori antrian, karya Kleinrock secara elegan mendemonstrasikan keefektifan pendekatan packet-switching untuk sumber lalu lintas yang meledak. Pada tahun 1964, Paul Baran [Baran 1964] di Institut Rand telah mulai menyelidiki penggunaan pengalihan paket untuk suara yang aman melalui jaringan militer, dan di Laboratorium Fisik Nasional di Inggris, Donald Davies dan Roger Scantlebury juga mengembangkan gagasan mereka tentang perpindahan paket.

1.7 • SEJARAH JARINGAN KOMPUTER DAN INTERNET 61

Pekerjaan di MIT, Rand, dan NPL meletakkan dasar untuk jaringan Internet hari ini. Tetapi Internet juga memiliki sejarah panjang tentang sikap ayo-bangun-dan-tunjukkan-itu yang juga sudah ada sejak tahun 1960-an. JCR Licklider [DEC 1990] dan Lawrence Roberts, keduanya rekan kerja Kleinrock di MIT, memimpin program ilmu komputer di Advanced Research Projects Agency (ARPA) di Amerika Serikat. Roberts menerbitkan rencana keseluruhan untuk ARPAnet [Roberts 1967], jaringan komputer pertama yang dialihkan paket dan nenek moyang langsung dari Internet publik saat ini. Pada Hari Buruh tahun 1969, saklar paket pertama dipasang di UCLA di bawah pengawasan Kleinrock, dan tiga saklar paket tambahan dipasang segera sesudahnya di Stanford Research Institute (SRI), UC Santa Bar Bara, dan Universitas Utah (Gambar 1.26). Prekursor pemula untuk



Gambar 1.26 Pergantian paket awal

62 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

Internet adalah empat node besar pada akhir tahun 1969. Kleinrock mengingat penggunaan pertama jaringan untuk melakukan login jarak jauh dari UCLA ke SRI, merusak sistem [Kleinrock 2004].

Pada tahun 1972, ARPAnet telah berkembang menjadi sekitar 15 node dan diberikan demonstrasi publik pertamanya oleh Robert Kahn. Protokol host-ke-host pertama antara sistem akhir ARPAnet, yang dikenal sebagai protokol kontrol jaringan (NCP), telah diselesaikan [RFC 001]. Dengan tersedianya protokol end-to-end, aplikasi sekarang dapat ditulisi. Ray Tomlinson menulis program email pertama pada tahun 1972.

1.7.2 Jaringan Proprietary dan Internetworking: 1972–1980

ARPAnet awal adalah satu jaringan tertutup. Untuk berkomunikasi dengan host ARPAnet, seseorang harus benar-benar terhubung ke IMP ARPAnet lain. Pada awal hingga pertengahan 1970-an, jaringan packet-switching tambahan yang berdiri sendiri selain ARPAnet muncul: ALOHAnet, jaringan gelombang mikro yang menghubungkan universitas di pulau Hawaii [Abramson 1970], serta paket-satelit DARPA [RFC 829] dan jaringan paket-radio [Kahn 1978]; Telenet, jaringan pengalihan paket komersial BBN berdasarkan teknologi ARPAnet; Cyclades, jaringan pengalihan paket Prancis yang dipelopori oleh Louis Pouzin [Think 2012]; Jaringan pembagian waktu seperti Tymnet dan jaringan Layanan Informasi GE, antara lain, pada akhir 1960-an dan awal 1970-an [Schwartz 1977]; SNA IBM (1969–1974), yang paralel dengan pekerjaan ARPAnet [Schwartz 1977].

Jumlah jaringan pun bertambah. Dengan melihat ke belakang secara sempurna, kita dapat melihat bahwa sudah waktunya untuk mengembangkan arsitektur yang mencakup untuk menghubungkan jaringan bersama. Pekerjaan perintis pada jaringan interkoneksi (di bawah kapal sponsor Defense Advanced Research Projects Agency (DARPA)), pada dasarnya menciptakan jaringan *dari jaringan*, dilakukan oleh Vinton Cerf dan Robert Kahn [Cerf 1974]; istilah *internetting* diciptakan untuk menggambarkan pekerjaan ini.

Prinsip-prinsip arsitektur ini diwujudkan dalam TCP. Namun, versi awal TCP sangat berbeda dari TCP saat ini. Versi awal TCP menggabungkan pengiriman data berurutan yang andal melalui transmisi ulang sistem akhir (masih menjadi bagian dari TCP saat ini) dengan fungsi penerusan (yang saat ini dilakukan oleh IP). Eksperimen awal dengan TCP, dikombinasikan dengan pengakuan akan pentingnya layanan transportasi end-to-end yang tidak dapat diandalkan, tidak dikontrol aliran, untuk aplikasi seperti paket suara, menyebabkan pemisahan IP dari TCP dan pengembangan protokol UDP. Tiga protokol Internet utama yang kita lihat hari ini—TCP, UDP, dan IP—secara konseptual sudah ada pada akhir tahun 1970-an.

Selain penelitian terkait Internet DARPA, banyak kegiatan jaringan penting lainnya sedang berlangsung. Di Hawaii, Norman Abramson sedang mengembangkan ALOHAnet, sebuah jaringan radio berbasis paket yang memungkinkan beberapa lokasi terpencil di Kepulauan Hawaii untuk berkomunikasi satu sama lain. Protokol ALOHA

1.7 • SEJARAH JARINGAN KOMPUTER DAN INTERNET 63

[Abramson 1970] adalah protokol multi-akses pertama, memungkinkan pengguna yang terdistribusi secara geografis untuk berbagi satu media komunikasi siaran (frekuensi radio). Metcalfe dan Boggs dibangun di atas kerja protokol multi-akses Abramson ketika mereka mengembangkan protokol Ethernet [Metcalfe 1976] untuk jaringan siaran bersama berbasis kabel. Menariknya, protokol Ethernet Metcalfe dan Boggs dimotivasi oleh kebutuhan untuk menghubungkan banyak PC, printer, dan disk bersama [Perkins 1994]. Dua puluh lima tahun yang lalu, jauh sebelum revolusi PC dan ledakan jaringan, Metcalfe dan Boggs meletakkan dasar LAN PC saat ini.

1.7.3 Proliferasi Jaringan: 1980–1990

Pada akhir tahun 1970-an, sekitar dua ratus host terhubung ke ARPAnet. Pada akhir 1980-an jumlah host yang terhubung ke Internet publik, sebuah konfederasi jaringan yang terlihat seperti Internet saat ini, akan mencapai seratus ribu. Tahun 1980-an akan menjadi masa pertumbuhan yang luar biasa.

Sebagian besar pertumbuhan itu dihasilkan dari beberapa upaya berbeda untuk membuat jaringan komputer yang menghubungkan universitas bersama. BITNET menyediakan transfer email dan file di antara beberapa universitas di Timur Laut. CSNET (jaringan ilmu komputer) dibentuk untuk menghubungkan peneliti universitas yang tidak memiliki akses ke ARPAnet. Pada tahun 1986, NSFNET dibuat untuk menyediakan akses ke pusat superkomputer yang disponsori NSF. Dimulai dengan kecepatan backbone awal 56 kbps, backbone NSFNET akan berjalan pada 1,5 Mbps pada akhir dekade ini dan akan berfungsi sebagai backbone utama yang menghubungkan jaringan regional.

Dalam komunitas ARPAnet, banyak bagian terakhir dari arsitektur Internet saat ini yang jatuh ke tempatnya. 1 Januari 1983 melihat penyebaran resmi TCP/IP sebagai protokol host standar baru untuk ARPAnet (menggantikan protokol NCP). Transisi [RFC 801] dari NCP ke TCP/IP adalah peristiwa hari bendera—semua host diminta untuk mentransfer ke TCP/IP pada hari itu. Pada akhir 1980-an, ekstensi penting dibuat untuk TCP untuk mengimplementasikan kontrol kongesti berbasis host [Jacobson 1988]. DNS, yang digunakan untuk memetakan antara nama Internet yang dapat dibaca manusia (misalnya, gaia.cs.umass.edu) dan alamat IP 32-bitnya, juga dikembangkan [RFC 1034].

Sejalan dengan perkembangan ARPAnet ini (yang sebagian besar merupakan upaya AS), pada awal 1980-an Prancis meluncurkan proyek Minitel, sebuah rencana ambisius untuk membawa jaringan data ke rumah semua orang. Disponsori oleh pemerintah Prancis, sistem Minitel terdiri dari jaringan pengalih paket publik (berdasarkan rangkaian protokol X.25), server Minitel, dan terminal murah dengan modem berkecepatan rendah bawaan. Minitel menjadi sukses besar pada tahun 1984 ketika pemerintah Prancis memberikan terminal Minitel gratis kepada setiap rumah tangga Prancis yang menginginkannya. Situs Minitel menyertakan situs gratis—seperti situs direktori telepon—serta situs pribadi, yang memungut biaya berdasarkan penggunaan dari

setiap pengguna. Pada puncaknya pada pertengahan 1990-an, ia menawarkan lebih dari 20.000 layanan, mulai dari home banking hingga database penelitian khusus. Minitel berada di sebagian besar rumah Prancis 10 tahun sebelum kebanyakan orang Amerika pernah mendengar tentang Internet.

1.7.4 Ledakan Internet: Tahun 1990-an

Tahun 1990-an diantar dengan sejumlah peristiwa yang melambangkan evolusi lanjutan dan komersialisasi Internet yang akan segera tiba. ARPAnet, nenek moyang Internet, tidak ada lagi. Pada tahun 1991, NSFNET mencabut larangan penggunaan NSFNET untuk tujuan komersial. NSFNET sendiri akan dinonaktifkan pada tahun 1995, dengan lalu lintas tulang punggung Internet dibawa oleh Penyedia Layanan Internet komersial.

Peristiwa utama tahun 1990-an adalah munculnya aplikasi World Wide Web, yang membawa Internet ke rumah dan bisnis jutaan orang di seluruh dunia. Web berfungsi sebagai platform untuk mengaktifkan dan menyebarkan ratusan aplikasi baru yang kita terima hari ini, termasuk pencarian (mis., Google dan Bing), perdagangan Internet (mis., Amazon dan eBay) dan jejaring sosial (mis., Facebook).

Web ditemukan di CERN oleh Tim Berners-Lee antara tahun 1989 dan 1991 [Berners-Lee 1989], berdasarkan ide yang berasal dari karya sebelumnya tentang hypertext dari tahun 1940-an oleh Vannevar Bush [Bush 1945] dan sejak tahun 1960-an oleh Ted Nelson [Xanadu 2012]. Berners-Lee dan rekan-rekannya mengembangkan versi awal HTML, HTTP, server Web, dan browser—empat komponen utama Web. Sekitar akhir tahun 1993 ada sekitar dua ratus server Web yang beroperasi, kumpulan server ini hanyalah pertanda dari apa yang akan datang. Pada saat ini beberapa peneliti sedang mengembangkan browser Web dengan antarmuka GUI, termasuk Marc Andreessen, yang bersama Jim Clark, membentuk Mosaic Communications, yang kemudian menjadi Netscape Communications Corporation [Cusumano 1998; Quitner 1998]. Pada tahun 1995, mahasiswa menggunakan browser Netscape untuk menjelajahi Web setiap hari. Sekitar waktu ini perusahaan—besar dan kecil—mulai mengoperasikan server Web dan melakukan transaksi perdagangan melalui Web. Pada tahun 1996, Microsoft mulai membuat browser, yang memulai perang browser antara Netscape dan Microsoft, yang dimenangkan oleh Microsoft beberapa tahun kemudian [Cusumano 1998].

Paruh kedua tahun 1990-an adalah periode pertumbuhan dan inovasi yang luar biasa untuk Internet, dengan perusahaan-perusahaan besar dan ribuan pemula menciptakan produk dan layanan Internet. Pada akhir milenium, Internet mendukung ratusan aplikasi populer, termasuk empat aplikasi mematikan:

- Email, termasuk lampiran dan email yang dapat diakses Web •
- Web, termasuk penjelajahan Web dan perdagangan Internet

1.7 • SEJARAH JARINGAN KOMPUTER DAN INTERNET 65

- Pesan instan, dengan daftar kontak •

Berbagi file MP3 peer-to-peer, dipelopori oleh Napster

Menariknya, dua killer application pertama berasal dari komunitas riset, sedangkan dua yang terakhir diciptakan oleh beberapa pengusaha muda.

Periode dari tahun 1995 hingga 2001 merupakan perjalanan roller-coaster untuk Internet di pasar keuangan. Bahkan sebelum mereka untung, ratusan startup Internet melakukan penawaran umum perdana dan mulai diperdagangkan di pasar saham. Banyak perusahaan dihargai dalam miliaran dolar tanpa memiliki aliran pendapatan yang signifikan. Saham Internet ambruk pada tahun 2000–2001, dan banyak perusahaan rintisan tutup.

Namun demikian, sejumlah perusahaan muncul sebagai pemenang besar di ruang Internet, termasuk Microsoft, Cisco, Yahoo, e-Bay, Google, dan Amazon.

1.7.5 Milenium Baru

Inovasi dalam jaringan komputer terus berlanjut dengan pesat. Kemajuan sedang dibuat di semua lini, termasuk penyebaran router yang lebih cepat dan kecepatan transmisi yang lebih tinggi baik di jaringan akses maupun di tulang punggung jaringan. Tetapi opsi pengembangan berikut patut mendapat perhatian khusus:

- Sejak awal milenium, kita telah melihat penggelaran agresif akses Internet broadband ke rumah—tidak hanya modem kabel dan DSL tetapi juga fiber ke rumah, seperti yang dibahas di Bagian 1.2. Akses Internet berkecepatan tinggi ini telah menyiapkan panggung untuk banyak aplikasi video, termasuk distribusi video buatan pengguna (misalnya, YouTube), streaming film dan acara televisi sesuai permintaan (misalnya, Netflix), dan multi-konferensi video -orang (misalnya, Skype).
- Meningkatnya keberadaan jaringan WiFi publik berkecepatan tinggi (54 Mbps dan lebih tinggi) dan akses Internet kecepatan menengah (hingga beberapa Mbps) melalui jaringan telepon seluler 3G dan 4G tidak hanya memungkinkan untuk tetap terhubung secara konstan sambil bergerak, tetapi juga mengaktifkan aplikasi khusus lokasi baru. Jumlah perangkat nirkabel yang terhubung ke Internet melampaui jumlah perangkat kabel pada tahun 2011. Akses nirkabel berkecepatan tinggi ini telah mengatur panggung untuk kemunculan cepat komputer genggam (iPhone, Android, iPad, dan sebagainya), yang menikmati akses konstan dan tanpa kabel ke Internet.
- Jejaring sosial online, seperti Facebook dan Twitter, telah menciptakan jaringan orang yang masif di atas Internet. Banyak pengguna Internet saat ini "hidup" terutama di dalam Facebook. Melalui API mereka, jejaring sosial online membuat formulir plat untuk aplikasi jaringan baru dan permainan terdistribusi.

66 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

- Sebagaimana dibahas dalam Bagian 1.3.3, penyedia layanan online, seperti Google dan Microsoft, telah menerapkan jaringan pribadi mereka yang luas, yang tidak hanya menghubungkan bersama pusat data mereka yang didistribusikan secara global, tetapi juga digunakan untuk mem-bypass Internet sebanyak mungkin oleh mengintip langsung dengan ISP tingkat bawah. Akibatnya, Google memberikan hasil pencarian dan akses email hampir seketika, seolah-olah pusat data mereka berjalan di dalam komputer sendiri.
- Banyak perusahaan perdagangan Internet sekarang menjalankan aplikasi mereka di "awan"—seperti di EC2 Amazon, di Mesin Aplikasi Google, atau di Microsoft Azure. Banyak perusahaan dan universitas juga memigrasikan aplikasi internet mereka (misalnya, email dan hosting web) ke cloud. Perusahaan cloud tidak hanya menyediakan lingkungan komputasi dan penyimpanan aplikasi yang dapat diskalakan, tetapi juga menyediakan akses implisit aplikasi ke jaringan pribadi berkinerja tinggi mereka.

1.8 Ringkasan

Dalam bab ini kita telah membahas banyak sekali materi! Kami telah melihat berbagai perangkat keras dan perangkat lunak yang membentuk Internet pada khususnya dan jaringan komputer pada umumnya. Kami mulai di tepi jaringan, melihat sistem dan aplikasi akhir, dan layanan transportasi yang disediakan untuk aplikasi yang berjalan di sistem akhir. Kami juga melihat teknologi lapisan tautan dan media fisik yang biasanya ditemukan di jaringan akses. Kami kemudian masuk lebih dalam ke dalam jaringan, ke dalam inti jaringan, mengidentifikasi packet switching dan circuit switching sebagai dua pendekatan dasar untuk mengangkut data melalui jaringan telekomunikasi, dan kami memeriksa kekuatan dan kelemahan masing-masing pendekatan. Kami juga memeriksa struktur Internet global, mempelajari bahwa Internet adalah jaringan dari jaringan. Kami melihat bahwa struktur hierarkis Internet, yang terdiri dari ISP tingkat atas dan bawah, memungkinkannya berkembang hingga mencakup ribuan jaringan.

Pada bagian kedua dari bab pengantar ini, kami memeriksa beberapa topik utama dalam bidang jaringan komputer. Kami pertama-tama memeriksa penyebab keterlambatan, melalui put dan packet loss di jaringan packet-switched. Kami mengembangkan model kuantitatif sederhana untuk transmisi, propagasi, dan penundaan antrian serta untuk throughput; kita akan menggunakan model keterlambatan ini secara ekstensif dalam soal-soal pekerjaan rumah di sepanjang buku ini. Selanjutnya kita memeriksa layering protokol dan model layanan, prinsip arsitektur utama dalam jaringan yang juga akan kita rujuk kembali di seluruh buku ini. Kami juga mensurvei beberapa serangan keamanan yang lebih lazim di zaman Internet. Kami menyelesaikan pengantar jaringan dengan sejarah singkat jaringan komputer. Bab pertama itu sendiri merupakan kursus mini dalam jaringan komputer.

Jadi, kita memang telah membahas banyak hal di bab pertama ini! Jika Anda sedikit kewalahan, jangan khawatir. Dalam bab-bab berikutnya kita akan meninjau kembali semua ide ini, membahasnya dengan lebih mendetail (itu janji, bukan ancaman!). Pada titik ini, kami harap Anda meninggalkan bab ini dengan intuisi yang masih berkembang untuk setiap bagian

yang membentuk jaringan, perintah kosa kata jaringan yang masih berkembang (jangan malu untuk merujuk kembali ke bab ini), dan keinginan yang terus tumbuh untuk mempelajari lebih lanjut tentang jaringan. Itulah tugas di depan kita untuk sisa buku ini.

Road-Mapping Buku Ini

Sebelum memulai perjalanan apa pun, Anda harus selalu melihat peta jalan agar terbiasa dengan jalan utama dan persimpangan yang ada di depan. Untuk perjalanan yang akan kita mulai, tujuan akhirnya adalah pemahaman mendalam tentang bagaimana, apa, dan mengapa jaringan komputer. Peta jalan kami adalah urutan bab dari buku ini:

1. Jaringan Komputer dan Internet 2.
 - Lapisan Aplikasi 3. Lapisan Transport 4.
 - Lapisan Jaringan 5. Lapisan Tautan dan Jaringan Area Lokal 6. Jaringan Nirkabel dan Seluler 7. Jaringan Multimedia 8.
 - Keamanan di Jaringan Komputer 9.
- Manajemen Jaringan

Bab 2 sampai 5 adalah empat bab inti dari buku ini. Anda harus memperhatikan bahwa bab-bab ini diatur di sekitar empat lapisan teratas dari tumpukan protokol Internet lima lapis, satu bab untuk setiap lapisan. Perhatikan lebih lanjut bahwa perjalanan kita akan dimulai di bagian atas tumpukan protokol Internet, yaitu, lapisan aplikasi, dan akan berjalan ke bawah. Alasan di balik perjalanan top-down ini adalah setelah kami memahami aplikasinya, kami dapat memahami layanan jaringan yang diperlukan untuk mendukung aplikasi ini. Selanjutnya, kita dapat memeriksa berbagai cara di mana layanan tersebut dapat diimplementasikan oleh arsitektur jaringan. Meliputi aplikasi lebih awal sehingga memberikan motivasi untuk sisa teks.

Paruh kedua buku ini—Bab 6 sampai 9—memperbesar empat topik yang sangat penting (dan agak independen) dalam jaringan komputer modern. Dalam Bab 6, kita mempelajari jaringan nirkabel dan seluler, termasuk LAN nirkabel (termasuk WiFi dan Bluetooth), jaringan telepon seluler (termasuk GSM, 3G, dan 4G), dan mobilitas (dalam jaringan IP dan GSM). Dalam Bab 7 (Jaringan Multimedia) kami memeriksa aplikasi audio dan video seperti telepon Internet, konferensi video, dan streaming media tersimpan. Kami juga melihat bagaimana jaringan packet-switched dapat dirancang untuk memberikan kualitas layanan yang konsisten untuk aplikasi audio dan video. Dalam Bab 8 (Keamanan di Jaringan Komputer), pertama-tama kita melihat dasar-dasar enkripsi dan keamanan jaringan, dan kemudian kita memeriksa bagaimana teori dasar diterapkan dalam berbagai konteks Internet.

Bab terakhir (Manajemen Jaringan) mengkaji isu-isu utama dalam manajemen jaringan serta protokol Internet utama yang digunakan untuk manajemen jaringan.

68 BAB 1

• JARINGAN KOMPUTER DAN INTERNET



Masalah dan Pertanyaan Pekerjaan Rumah

Bab 1 Tinjau Pertanyaan

BAGIAN 1.1

- R1. Apa perbedaan antara host dan sistem akhir? Daftar beberapa berbeda jenis sistem akhir. Apakah server Web merupakan sistem akhir?
- R2. Kata *protokol* sering digunakan untuk menggambarkan hubungan diplomatik. Bagaimana Wikipedia menggambarkan protokol diplomatik?
- R3. Mengapa standar penting untuk protokol?

BAGIAN 1.2

- R4. Sebutkan enam teknologi akses. Klasifikasikan masing-masing sebagai akses rumah, perusahaan akses, atau akses nirkabel area luas.
- R5. Apakah tingkat transmisi HFC didedikasikan atau dibagikan di antara pengguna? Apakah tabrakan mungkin terjadi di saluran HFC hilir? Mengapa atau mengapa tidak?
- R6. Cantumkan teknologi akses perumahan yang tersedia di kota Anda. Untuk setiap jenis akses, berikan tarif downstream, tarif upstream, dan harga bulanan yang diiklankan.
- R7. Berapa tingkat transmisi LAN Ethernet?
- R8. Apa saja media fisik yang dapat dijalankan oleh Ethernet?
- R9. Modem dial-up, HFC, DSL dan FTTH semuanya digunakan untuk akses perumahan. Untuk masing-masing teknologi akses ini, berikan kisaran kecepatan transmisi dan beri komentar apakah kecepatan transmisi dibagi atau didedikasikan.
- R10. Jelaskan teknologi akses Internet nirkabel paling populer saat ini. Kompare dan kontras mereka.

BAGIAN 1.3

- R11. Misalkan ada tepat satu packet switch antara host pengirim dan host penerima. Tingkat transmisi antara host pengirim dan switch dan antara switch dan host penerima masing-masing adalah R1 dan R2 . Asumsikan bahwa switch menggunakan store-and-forward packet switching, berapa total delay end-to-end untuk mengirim paket dengan panjang L? (Abaikan antrian, penundaan propagasi, dan penundaan pemrosesan.)
- R12. Apa keunggulan jaringan circuit-switched dibandingkan jaringan packet-switched? Apa keunggulan yang dimiliki TDM dibandingkan FDM dalam jaringan circuit-switched?
- R13. Misalkan pengguna berbagi tautan 2 Mbps. Juga misalkan setiap pengguna mentransmisikan secara terus-menerus pada 1 Mbps saat mentransmisikan, tetapi setiap pengguna hanya mentransmisikan 20 persen dari waktu. (Lihat pembahasan multiplexing statistik di Bagian 1.3.)

SOAL DAN PERTANYAAN PR 69

- A. Ketika switching sirkuit digunakan, berapa banyak pengguna yang dapat didukung?
- B. Untuk sisa masalah ini, misalkan packet switching digunakan. Mengapa pada dasarnya tidak ada penundaan antrian sebelum tautan jika dua atau lebih sedikit pengguna mengirim pada waktu yang sama? Mengapa akan ada penundaan antrian jika tiga pengguna mengirimkan pada waktu yang sama?
- C. Temukan probabilitas yang dikirim oleh pengguna tertentu. D.

Misalkan sekarang ada tiga pengguna. Temukan probabilitas bahwa pada waktu tertentu, ketiga pengguna melakukan transmisi secara bersamaan. Temukan fraksi waktu di mana antrian tumbuh.

R14. Mengapa dua ISP pada tingkat hierarki yang sama sering saling mengintip lainnya? Bagaimana IXP menghasilkan uang?

R15. Beberapa penyedia konten telah membuat jaringan mereka sendiri. Jelaskan jaringan Google. Apa yang memotivasi penyedia konten untuk membuat jaringan ini?

BAGIAN 1.4

R16. Pertimbangkan untuk mengirim paket dari host sumber ke host tujuan melalui rute tetap. Buat daftar komponen delay dalam end-to-end delay. Manakah dari penundaan ini yang konstan dan mana yang variabel?

R17. Kunjungi applet Penundaan Transmisi Versus Propagasi di situs Web pendamping. Di antara rate, delay propagasi, dan ukuran paket yang tersedia, temukan kombinasi yang mana pengirim selesai mengirim sebelum bit pertama paket mencapai penerima. Temukan kombinasi lain yang bit pertama dari paket mencapai penerima sebelum pengirim selesai mengirim.

R18. Berapa lama waktu yang diperlukan paket dengan panjang 1.000 byte untuk menyebar melalui tautan dengan jarak 2.500 km, kecepatan propagasi $2.5 \cdot 10^8$ m/s, dan kecepatan transmisi R Mbps? Secara lebih umum, berapa lama waktu yang dibutuhkan sebuah paket dengan panjang L untuk merambat melalui sebuah link dengan jarak d , kecepatan rambat s , dan laju transmisi R bps? Apakah penundaan ini tergantung pada panjang paket? Apakah penundaan ini tergantung pada tingkat transmisi?

R19. Misalkan Host A ingin mengirim file besar ke Host B. Jalur dari Host A ke Host B memiliki tiga link, dengan tarif $R_1 = 500$ kbps, $R_2 = 2$ Mbps, dan $R_3 = 1$ Mbps. A. Dengan asumsi tidak ada lalu lintas lain dalam jaringan, berapakah throughput untuk pemindahan berkas?

B. Misalkan file tersebut berukuran 4 juta byte. Membagi ukuran file dengan throughput, kira-kira berapa lama waktu yang dibutuhkan untuk mentransfer file ke Host B? C. Ulangi (a) dan (b), tetapi sekarang dengan R_2 dikurangi menjadi 100 kbps.

R20. Misalkan sistem akhir A ingin mengirim file besar ke sistem akhir B. Pada tingkat yang sangat tinggi, jelaskan bagaimana sistem akhir A membuat paket dari file tersebut. Kapan

70 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

salah satu paket ini tiba di sebuah packet switch, informasi apa di dalam paket yang digunakan oleh switch untuk menentukan tautan ke mana paket tersebut diteruskan? Mengapa pengalihan paket di Internet dianalogikan dengan mengemudi dari satu kota ke kota lain dan menanyakan arah di sepanjang jalan?

R21. Kunjungi applet Antrian dan Rugi di situs Web pendamping. Berapa tingkat emisi maksimum dan tingkat transmisi minimum? Dengan tarif tersebut, berapa intensitas lalu lintasnya? Jalankan applet dengan tarif ini dan tentukan berapa lama waktu yang dibutuhkan untuk terjadinya kehilangan paket. Kemudian ulangi percobaan untuk kedua kalinya dan tentukan lagi berapa lama waktu yang diperlukan untuk terjadinya packet loss.

Apakah nilainya berbeda? Mengapa atau mengapa tidak?

BAGIAN 1.5

R22. Buat daftar lima tugas yang dapat dilakukan oleh lapisan. Mungkinkah satu (atau lebih) tugas ini dapat dilakukan oleh dua (atau lebih) lapisan?

R23. Apa saja lima lapisan dalam tumpukan protokol Internet? Apa kepala sekolah tanggung jawab masing-masing lapisan ini?

R24. Apa itu pesan lapisan aplikasi? Segmen lapisan transportasi? Datagram lapisan jaringan? Bingkai lapisan tautan?

R25. Lapisan mana dalam tumpukan protokol Internet yang diproses oleh router? Yang lapisan apakah proses peralihan tautan-lapisan? Lapisan mana yang diproses oleh host?

BAGIAN 1.6

R26. Apa perbedaan antara virus dan worm?

R27. Jelaskan bagaimana botnet dapat dibuat, dan bagaimana botnet dapat digunakan untuk DDoS menyerang.

R28. Misalkan Alice dan Bob saling mengirim paket melalui jaringan komputer. Misalkan Trudy memposisikan dirinya dalam jaringan sehingga dia dapat menangkap semua paket yang dikirim oleh Alice dan mengirimkan apapun yang dia inginkan ke Bob; dia juga dapat menangkap semua paket yang dikirim oleh Bob dan mengirimkan apapun yang dia inginkan ke Alice. Sebutkan beberapa hal jahat yang dapat dilakukan Trudy dari posisi ini.

**Masalah**

P1. Rancang dan jelaskan protokol tingkat aplikasi yang akan digunakan antara anjungan tunai mandiri dan komputer terpusat bank. Protokol Anda harus memungkinkan verifikasi kartu dan kata sandi pengguna, saldo akun (yang disimpan di komputer terpusat) untuk ditanyakan, dan penarikan akun dilakukan (yaitu, uang dicairkan ke pengguna). Milikmu

entitas protokol harus dapat menangani kasus yang terlalu umum di mana tidak ada cukup uang di akun untuk menutup penarikan. Tentukan protokol Anda dengan mencantumkan pesan yang dipertukarkan dan tindakan yang diambil oleh anjungan tunai mandiri atau komputer terpusat bank pada pengiriman dan penerimaan pesan. Buat sketsa operasi protokol Anda untuk kasus penarikan sederhana tanpa kesalahan, menggunakan diagram yang serupa dengan yang ada di Gambar 1.2.

Nyatakan secara eksplisit asumsi yang dibuat oleh protokol Anda tentang layanan transportasi end-to-end yang mendasarinya.

P2. Persamaan 1.1 memberikan formula untuk delay end-to-end dari pengiriman satu paket dengan panjang L melewati N link dengan laju transmisi R . Generalisasikan formula ini untuk mengirim P paket seperti itu secara berurutan melalui N link.

P3. Pertimbangkan aplikasi yang mentransmisikan data pada kecepatan tetap (misalnya, pengirim menghasilkan unit data N -bit setiap k unit waktu, di mana k kecil dan tetap). Juga, ketika aplikasi seperti itu dimulai, itu akan terus berjalan untuk jangka waktu yang relatif lama. Jawablah pertanyaan-pertanyaan berikut, dengan singkat membenarkan jawaban Anda: a. Apakah jaringan packet-switched atau jaringan circuit-switched lebih

cocok untuk aplikasi ini? Mengapa?

B. Misalkan sebuah jaringan packet-switched digunakan dan satu-satunya lalu lintas di jaringan ini berasal dari aplikasi seperti yang dijelaskan di atas. Lebih jauh lagi, asumsikan bahwa jumlah tarif data aplikasi kurang dari kapasitas setiap tautan. Apakah diperlukan suatu bentuk pengendalian kemacetan? Mengapa?

P4. Pertimbangkan jaringan circuit-switched pada Gambar 1.13. Ingatlah bahwa ada 4 sirkuit di setiap tautan. Beri label keempat sakelar A, B, C, dan D, searah jarum jam.

A. Berapa jumlah maksimum koneksi simultan yang bisa masuk kemajuan pada satu waktu dalam jaringan ini?
 B. Misalkan semua koneksi berada di antara sakelar A dan C. Berapa jumlah maksimum koneksi simultan yang dapat berlangsung? C. Misalkan kita ingin membuat empat sambungan antara sakelar A dan C, dan empat sambungan lainnya antara sakelar B dan D. Bisakah kita merutekan panggilan ini melalui empat tautan untuk mengakomodasi delapan sambungan?

P5. Tinjau analogi mobil-karavan di Bagian 1.4. Asumsikan kecepatan propagasi 100 km/jam.

A. Anggaplah kafilah tersebut menempuh jarak 150 km, dimulai dari depan satu gardu tol, melewati gardu tol kedua, dan berakhir tepat setelah gardu tol ketiga. Apa itu penundaan ujung ke ujung?
 B. Ulangi (a), sekarang asumsikan ada delapan mobil dalam karavan sering.

72 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

VideoNote
Menjelajahi penundaan
propagasi dan penundaan
transmisi

P6. Masalah mendasar ini mulai mengeksplorasi delay propagasi dan delay transmisi, dua konsep sentral dalam jaringan data. Pertimbangkan dua host, A dan B, dihubungkan oleh satu tautan dengan kecepatan R bps. Misalkan dua host dipisahkan oleh m meter, dan misalkan kecepatan propagasi sepanjang link adalah s meter/detik. Host A akan mengirimkan paket berukuran L bit ke Host B. a. Nyatakan delay propagasi, dprop, dalam satuan m dan tentukan waktu

transmisi paket, dtrans, dalam satuan L

dan R .

- C. Mengabaikan penundaan pemrosesan dan antrian, dapatkan ekspresi untuk akhir penundaan sampai akhir.
- D. Misalkan Host A mulai mentransmisikan paket pada waktu $t = 0$. Pada waktu $t = dtrans$, dimana bit terakhir dari paket tersebut? e. Misalkan dprop lebih besar dari dtrans. Pada saat $t = dtrans$, dimana adalah bit pertama dari paketnya?
- F. Misalkan dprop kurang dari dtrans. Pada saat $t = dtrans$, dimana adalah bit pertama dari paketnya?
- G. Misalkan $s = 2,5 \cdot 108$, $L = 120$ bit, dan $R = 56$ kbps. Cari jarak m sehingga dprop sama dengan dtrans.

P7. Dalam masalah ini, kami mempertimbangkan pengiriman suara real-time dari Host A ke Host B melalui jaringan packet-switched (VoIP). Host A mengonversi suara analog menjadi aliran bit 64 kbps digital dengan cepat. Host A kemudian mengelompokkan bit ke dalam paket 56-byte. Ada satu link antara Host A dan B; tingkat transmisinya adalah 2 Mbps dan penundaan propagasinya adalah 10 msec. Segera setelah Host A mengumpulkan paket, ia mengirimkannya ke Host B. Segera setelah Host B menerima seluruh paket, ia mengubah bit paket menjadi sinyal analog. Berapa banyak waktu yang berlalu sejak bit dibuat (dari sinyal analog asli di Host A) hingga bit didekodekan (sebagai bagian dari sinyal analog di Host B)?

P8. Misalkan pengguna berbagi tautan 3 Mbps. Misalkan juga setiap pengguna membutuhkan 150 kbps saat mentransmisikan, tetapi setiap pengguna hanya mentransmisikan 10 persen dari waktu. (Lihat pembahasan tentang packet switching versus circuit switching di Bagian 1.3.) a. Ketika switching sirkuit digunakan, berapa banyak pengguna yang dapat didukung? B. Untuk sisa masalah ini, misalkan packet switching digunakan. Temukan probabilitas yang dikirim oleh pengguna tertentu.

- C. Misalkan ada 120 pengguna. Temukan probabilitas bahwa pada waktu tertentu, tepat n pengguna melakukan transmisi secara bersamaan. (*Petunjuk:* Gunakan distribusi binomial.) d. Temukan probabilitas bahwa ada 21 pengguna atau lebih yang melakukan transmisi serentak.

P9. Pertimbangkan pembahasan di Bagian 1.3 tentang packet switching versus circuit switching di mana contoh disediakan dengan tautan 1 Mbps. Pengguna menghasilkan data dengan kecepatan 100 kbps saat sibuk, tetapi sibuk menghasilkan data hanya dengan probabilitas $p = 0,1$. Misalkan link 1 Mbps diganti dengan link 1 Gbps. A. Apa N , jumlah maksimum pengguna yang dapat didukung secara bersamaan di bawah circuit switching?

B. Sekarang pertimbangkan pengalihan paket dan populasi pengguna dari M pengguna. Berikan rumus (dalam hal p , M , N) untuk probabilitas bahwa lebih dari N pengguna mengirim data.

P10. Pertimbangkan paket dengan panjang L yang dimulai pada sistem akhir A dan berjalan melalui tiga tautan ke sistem akhir tujuan. Ketiga tautan ini dihubungkan oleh dua sakelar paket. Biarkan di , si , dan Ri menunjukkan panjang, kecepatan propaga si, dan laju transmisi paket untuk $i = 1, 2, 3$,

prok. Dengan asumsi tidak ada penundaan antrian, dalam hal di , si , Ri , ($i = 1,2,3$), dan L , berapa total delay end-to-end untuk paket tersebut? Misalkan sekarang paket adalah 1.500 byte, kecepatan propagasi pada ketiga link adalah $2,5 \cdot 108$ m/s, tingkat transmisi dari ketiga link adalah 2 Mbps, penundaan pemrosesan packet switch adalah 3 msec, panjang link pertama adalah 5.000 km, panjang mata rantai kedua 4.000 km, dan panjang mata rantai terakhir 1.000 km. Untuk nilai-nilai ini, berapa penundaan ujung ke ujung?

P11. Pada soal di atas, misalkan $R1 = R2 = R3 = R$ dan dproc = 0. Selanjutnya suppose packet switch tidak menyimpan-dan-meneruskan paket tetapi sebaliknya segera mentransmisikan setiap bit yang diterimanya sebelum menunggu seluruh paket tiba. Apa itu penundaan ujung ke ujung?

P12. Sakelar paket menerima paket dan menentukan tautan keluar ke mana paket harus diteruskan. Ketika paket tiba, satu paket lain setengah selesai ditransmisikan pada tautan keluar ini dan empat paket lainnya sedang menunggu untuk dikirim. Paket ditransmisikan dalam urutan kedatangan.

Misalkan semua paket adalah 1.500 byte dan kecepatan tautannya adalah 2 Mbps. Apa delay antrian untuk paket? Secara lebih umum, berapa penundaan antrian ketika semua paket memiliki panjang L , laju transmisi adalah R , x bit dari paket yang sedang dikirim telah dikirim, dan n paket sudah dalam antrian?

P13. (a) Misalkan N paket tiba secara bersamaan ke sebuah link di mana tidak ada paket yang sedang dikirim atau antri. Setiap paket memiliki panjang L dan link tersebut memiliki laju transmisi R . Berapakah delay antrian rata-rata untuk N paket?

(b) Sekarang misalkan N paket tersebut tiba ke link setiap detik LN/R . Berapa delay antrian rata-rata sebuah paket?

74 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

P14. Pertimbangkan penundaan antrian di buffer router. Biarkan saya menunjukkan intensitas lalu lintas; yaitu, $I = La/R$. Anggap delay antrian berbentuk $IL/R(1 - I)$ untuk $I < 1$.

- Berikan rumus untuk penundaan total, yaitu penundaan antrian ditambah penundaan transmisi.
- Plot delay total sebagai fungsi dari L/R .

P15. Biarkan a menunjukkan tingkat paket yang tiba di tautan dalam paket/detik, dan biarkan \bar{y} menunjukkan tingkat transmisi tautan dalam paket/detik. Berdasarkan rumus untuk penundaan total (yaitu, penundaan antrian ditambah penundaan transmisi) yang diperoleh dari masalah sebelumnya, turunkan rumus untuk penundaan total dalam bentuk a dan \bar{y} .

P16. Pertimbangkan buffer router sebelum tautan keluar. Dalam soal ini, Anda akan menggunakan rumus Little, rumus terkenal dari teori antrian. Biarkan N menunjukkan jumlah rata-rata paket dalam buffer ditambah paket yang sedang dikirim. Biarkan a menunjukkan tingkat paket yang tiba di tautan. Biarkan d menunjukkan rata-rata total penundaan (yaitu, penundaan antrian ditambah penundaan transmisi) yang dialami oleh sebuah paket. Rumus Little adalah $N = a \cdot d$. Misalkan rata-rata buffer berisi 10 paket, dan rata-rata delay antrian paket adalah 10 msec. Kecepatan transmisi tautan adalah 100 paket/detik. Dengan menggunakan rumus Little, berapa rata-rata tingkat kedatangan paket, dengan asumsi tidak ada paket yang hilang?

P17. A. Generalisasikan Persamaan 1.2 di Bagian 1.4.3 untuk laju pemrosesan yang heterogen, laju transmisi, dan penundaan propagasi.

- Ulangi (a), tetapi sekarang anggap juga ada penundaan antrian rata-rata dqueue di setiap node.

P18. Lakukan Traceroute antara sumber dan tujuan di benua yang sama pada tiga jam berbeda dalam sehari. A.

Temukan rata-rata dan deviasi standar dari penundaan bolak-balik pada masing-masing dari tiga jam.

- Temukan jumlah router di jalur pada masing-masing dari tiga jam. Melakukan jalur berubah selama setiap jam?
- Cobalah untuk mengidentifikasi jumlah jaringan ISP yang dilewati paket Traceroute melalui dari sumber ke tujuan. Router dengan nama yang mirip dan/atau alamat IP yang serupa harus dianggap sebagai bagian dari ISP yang sama. Dalam eksperimen Anda, apakah penundaan terbesar terjadi pada antarmuka peering antara ISP yang berdekatan?
- Ulangi langkah di atas untuk sumber dan tujuan di berbagai benua. Bandingkan hasil intra-benua dan antar-benua.

P19. (a) Kunjungi situs www.traceroute.org dan lakukan traceroute dari dua kota berbeda di Prancis ke host tujuan yang sama di Amerika Serikat. Berapa banyak tautan yang sama di kedua traceroutes? Apakah tautan transatlantiknya sama?



VideoNote

Menggunakan Traceroute untuk menemukan jalur jaringan dan mengukur keterlambatan jaringan

- (b) Ulangi (a) tetapi kali ini pilih satu kota di Prancis dan kota lain di dalamnya Jerman.
- (c) Pilih sebuah kota di Amerika Serikat, dan lakukan pelacakan rute ke dua host, masing-masing di kota berbeda di Cina. Berapa banyak tautan yang umum di kedua traceroutes? Apakah kedua traceroutes menyimpang sebelum mencapai China?

P20. Pertimbangkan contoh throughput yang sesuai dengan Gambar 1.20(b). Sekarang misalkan ada M pasangan klien-server daripada 10. Nyatakan Rs , Rc , dan R untuk tarif tautan server, tautan klien, dan tautan jaringan. Asumsikan semua link lain memiliki kapasitas yang melimpah dan tidak ada lalu lintas lain di jaringan selain lalu lintas yang dihasilkan oleh pasangan klien-server M . Turunkan ekspresi umum untuk throughput dalam bentuk Rs , Rc , R , dan M .

P21. Perhatikan Gambar 1.19(b). Sekarang misalkan ada M jalur antara server dan klien. Tidak ada dua jalur yang berbagi tautan apa pun. Jalur k ($k = 1, \dots, M$) terdiri dari N link dengan laju transmisi Rk . Rk menggunakan satu jalur untuk mengirim data ke klien, berpantulan kembali ke server. Jika server hanya bisa dapat menggunakan semua jalur M untuk mengirim data, berapa throughput maksimum yang dapat dicapai server?

P22. Perhatikan Gambar 1.19(b). Misalkan setiap link antara server dan klien memiliki probabilitas kehilangan paket p , dan probabilitas kehilangan paket untuk link ini adalah independen. Berapa probabilitas bahwa sebuah paket (dikirim oleh server) berhasil diterima oleh penerima? Jika sebuah paket hilang dalam jalur dari server ke klien, maka server akan mengirimkan kembali paket tersebut. Rata-rata, berapa kali server mengirimkan ulang paket agar klien berhasil menerima paket?

P23. Perhatikan Gambar 1.19(a). Asumsikan bahwa kita mengetahui link bottleneck sepanjang jalur dari server ke klien adalah link pertama dengan kecepatan Rs bit/detik. Misalkan kita mengirim sepasang paket bolak-balik dari server ke klien, dan tidak ada lalu lintas lain di jalur ini. Asumsikan setiap paket berukuran L bit, dan kedua link memiliki delay propagasi yang sama d a. Berapa waktu antar kedatangan paket di tempat tujuan? Artinya, berapa lama waktu berlalu dari saat bit terakhir dari paket pertama tiba hingga bit terakhir dari paket kedua tiba? B. Sekarang misalkan bahwa link kedua adalah link bottleneck (yaitu, $Rc < Rs$). Apakah mungkin paket kedua mengantre di antrian input dari tautan kedua? Menjelaskan. Sekarang misalkan server mengirimkan paket kedua T detik setelah mengirim paket pertama. Seberapa besar T harus memastikan tidak ada antrian sebelum tautan kedua? Menjelaskan.

P24. Misalkan Anda ingin segera mengirimkan data 40 terabyte dari Boston ke Los Angeles. Anda telah menyediakan tautan khusus 100 Mbps untuk transfer data. Apakah Anda lebih suka mengirimkan data melalui tautan ini atau menggunakan FedEx melalui pengiriman malam? Menjelaskan.

76 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

P25. Anggaplah dua host, A dan B, dipisahkan sejauh 20.000 kilometer dan dihubungkan dengan direct link sebesar $R = 2$ Mbps. Misalkan kecepatan propagasi melalui link adalah 2,5 108 meter/detik.

A. Hitung hasil kali bandwidth-delay, $R \cdot d_{\text{prop}}$. B.

Pertimbangkan untuk mengirim file 800.000 bit dari Host A ke Host B. Misalkan file dikirim secara terus menerus sebagai satu pesan besar. Berapa jumlah maksimum bit yang akan ada di tautan pada waktu tertentu? C. Berikan interpretasi produk bandwidth-delay. D. Berapa lebar (dalam meter) dari bit di tautan? Apakah lebih panjang dari satu kaki
lapangan bola?

e. Turunkan persamaan umum untuk lebar bit dalam kaitannya dengan kecepatan propagasi s , laju transmisi R , dan panjang link m .

P26. Mengacu pada soal P25, misalkan kita dapat memodifikasi R . Untuk nilai R berapa lebar sedikit selama panjang link?

P27. Pertimbangkan masalah P25 tetapi sekarang dengan tautan $R =$

1 Gbps. A. Hitung hasil kali bandwidth-delay, $R \cdot d_{\text{prop}}$. B.

Pertimbangkan untuk mengirim file 800.000 bit dari Host A ke Host B. Misalkan file tersebut dikirim secara terus menerus sebagai satu pesan besar. Berapa jumlah maksimum bit yang akan ada di tautan pada waktu tertentu?

C. Berapa lebar (dalam meter) dari bit di tautan?

P28. Rujuk kembali ke soal P25.

A. Berapa lama pengiriman file, dengan asumsi dikirim terus menerus? B. Misalkan sekarang file tersebut dipecah menjadi 20 paket dengan masing-masing paket berisi 40.000 bit. Misalkan setiap paket diakui oleh penerima dan waktu transmisi paket pengakuan dapat diabaikan. Terakhir, asumsikan bahwa pengirim tidak dapat mengirim paket sampai yang sebelumnya diakui. Berapa lama waktu yang dibutuhkan untuk mengirim file? C. Bandingkan hasil dari (a) dan (b).

P29. Misalkan ada link gelombang mikro 10 Mbps antara satelit geostasioner dan stasiun pangkalannya di Bumi. Setiap menit satelit mengambil foto digital dan mengirimkannya ke stasiun pangkalan. Asumsikan kecepatan propagasi 2,4 108 meter/detik. A. Apa delay propagasi dari link? B. Apa produk bandwidth-delay, $R \cdot d_{\text{prop}}$? C.

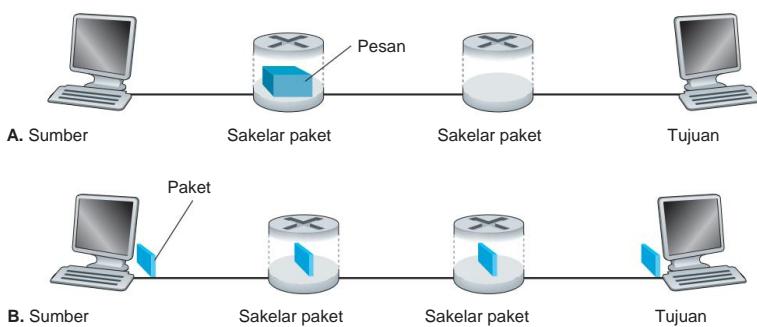
Biarkan x menunjukkan ukuran foto. Berapakah nilai minimum x agar sambungan gelombang mikro terus menerus dipancarkan?

P30. Pertimbangkan analogi perjalanan maskapai dalam pembahasan kita tentang layering di Bagian 1.5, dan penambahan header ke unit data protokol saat mengalir ke bawah

tumpukan protokol. Apakah ada gagasan yang setara tentang informasi tajuk yang ditambahkan ke penumpang dan bagasi saat mereka turun dari tumpukan protokol maskapai?

P31. Dalam jaringan packet-switched modern, termasuk Internet, host sumber membagi pesan lapisan aplikasi yang panjang (misalnya, gambar atau file musik) ke dalam paket yang lebih kecil dan mengirimkan paket ke jaringan. Penerima kemudian memasang kembali paket-paket itu kembali ke pesan aslinya. Kami menyebut proses ini sebagai *segmentasi pesan*. Gambar 1.27 mengilustrasikan pengangkutan pesan end-to-end dengan dan tanpa segmentasi pesan. Pertimbangkan pesan yang panjangnya $8 \cdot 10^6$ bit yang akan dikirim dari sumber ke tujuan pada Gambar 1.27. Misalkan setiap tautan pada gambar adalah 2 Mbps. Abaikan propagasi, antrian, dan penundaan pemrosesan. A. Pertimbangkan mengirim pesan dari sumber ke tujuan *tanpa* segmentasi pesan. Berapa lama waktu yang diperlukan untuk memindahkan pesan dari host sumber ke sakelar paket pertama? Perlu diingat bahwa setiap switch menggunakan store-and-forward packet switching, berapa total waktu untuk memindahkan pesan dari host sumber ke host tujuan? B. Sekarang misalkan pesan tersebut tersegmentasi menjadi 800 paket, dengan masing-masing paket panjangnya 10.000 bit. Berapa lama untuk memindahkan paket pertama dari host sumber ke switch pertama? Ketika paket pertama dikirim dari switch pertama ke switch kedua, paket kedua dikirim dari host sumber ke switch pertama. Pada jam berapa paket kedua akan diterima sepenuhnya pada saklar pertama?

C. Berapa lama waktu yang diperlukan untuk memindahkan file dari host sumber ke host tujuan saat segmentasi pesan digunakan? Bandingkan hasil ini dengan jawaban Anda di bagian (a) dan komentar.



Gambar 1.27 Transportasi pesan end-to-end: (a) tanpa segmentasi pesan; (b) dengan segmentasi pesan

78 BAB 1**• JARINGAN KOMPUTER DAN INTERNET**

D. Selain mengurangi delay, apa alasan menggunakan segmentasi pesan? e. Diskusikan kelebihan dan kelemahan segmentasi pesan.

P32. Berekspresikanlah dengan applet Segmentasi Pesan di situs Web buku. Apakah penundaan applet sesuai dengan penundaan masalah sebelumnya?

Bagaimana delay propagasi link mempengaruhi keseluruhan end-to-end delay untuk packet switching (dengan segmentasi pesan) dan untuk message switching?

P33. Pertimbangkan untuk mengirim file besar bit F dari Host A ke Host B. Ada tiga link (dan dua switch) antara A dan B, dan link tersebut tidak macet (yaitu, tidak ada penundaan antrian). Host A mengelompokkan file ke dalam segmen masing-masing S bit dan menambahkan 80 bit header ke setiap segmen, membentuk paket $L = 80 + S$ bit. Setiap link memiliki tingkat transmisi R bps. Temukan nilai S yang meminimalkan penundaan pemindahan file dari Host A ke Host B. Abaikan penundaan propagasi.

P34. Skype menawarkan layanan yang memungkinkan Anda melakukan panggilan telepon dari PC ke telepon biasa. Artinya, panggilan suara harus melewati Internet dan melalui jaringan telepon. Diskusikan bagaimana hal ini dapat dilakukan.



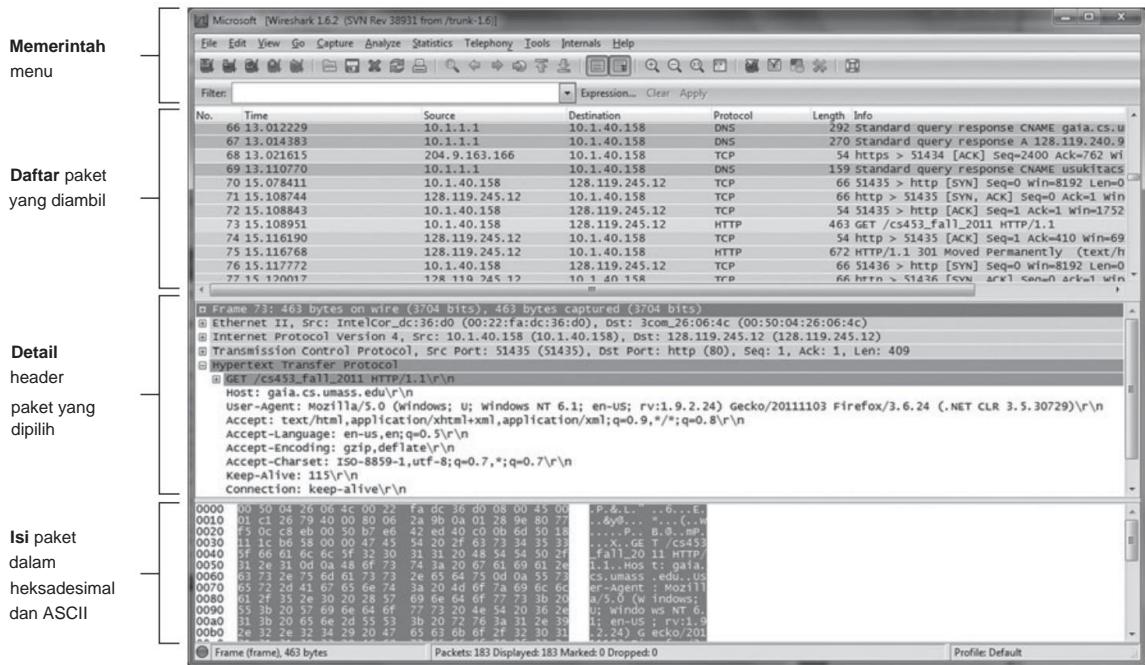
Laboratorium Wireshark

"Katakan padaku dan aku lupa. Tunjukkan padaku dan aku ingat. Libatkan saya dan saya mengerti."
— pepatah Cina

Pemahaman seseorang tentang protokol jaringan seringkali dapat sangat diperoleh dengan melihatnya beraksi dan bermain-main dengannya—mengamati urutan pesan yang dipertukarkan antara dua entitas protokol, menyelidiki detail operasi protokol, menyebabkan protokol melakukan tindakan tertentu, dan mengamati tindakan ini dan konsekuensinya. Ini dapat dilakukan dalam skenario simulasi atau dalam lingkungan jaringan nyata seperti Internet. Applet Java di situs Web buku teks mengambil pendekatan pertama. Di lab Wireshark, kami akan menggunakan pendekatan terakhir. Anda dapat menjalankan aplikasi kerja jaringan dalam berbagai skenario menggunakan komputer di meja Anda, di rumah, atau di lab. Anda akan mengamati protokol jaringan di komputer Anda, berinteraksi dan bertukar pesan dengan entitas protokol yang dijalankan di tempat lain di Internet.

Dengan demikian, Anda dan komputer Anda akan menjadi bagian integral dari lab langsung ini. Anda akan mengamati—and Anda akan belajar—dengan melakukan.

Alat dasar untuk mengamati pesan yang dipertukarkan antara entitas protokol pelaksana disebut **packet sniffer**. Seperti namanya, packet sniffer secara pasif menyalin (mengendus) pesan yang dikirim dari dan diterima oleh komputer Anda; itu juga menampilkan konten dari berbagai bidang protokol dari pesan yang ditangkap ini. Tangkapan layar dari paket sniffer Wireshark ditunjukkan pada Gambar 1.28. Wireshark adalah paket sniffer gratis yang berjalan di Windows, Linux/Unix, dan Mac.



Gambar 1.28 Tangkapan layar Wireshark (tangkapan layar Wireshark dicetak ulang dengan izin dari Wireshark Foundation.)

komputer. Di seluruh buku teks, Anda akan menemukan lab Wireshark yang memungkinkan Anda menjelajahi sejumlah protokol yang dipelajari di bab ini. Di lab Wireshark pertama ini, Anda akan mendapatkan dan menginstal salinan Wireshark, mengakses situs Web, dan menangkap serta memeriksa pesan protokol yang dipertukarkan antara browser Web Anda dan server Web.

Anda dapat menemukan detail lengkap tentang lab Wireshark pertama ini (termasuk instruksi tentang cara mendapatkan dan menginstal Wireshark) di situs Web <http://www.awl.com/kurose-ross>.

WAWANCARA DENGAN...

Leonard Kleinrock

Leonard Kleinrock adalah profesor ilmu komputer di University of California, Los Angeles. Pada tahun 1969, komputernya di UCLA menjadi simpul pertama Internet. Ciptaannya tentang prinsip-prinsip packet-switching pada tahun 1961 menjadi teknologi di belakang Internet. Dia menerima gelar BEE dari City College of New York (CCNY) dan master serta PhD di bidang teknik elektro dari MIT.



Apa yang membuat Anda memutuskan untuk berspesialisasi dalam teknologi jaringan/internet?

Sebagai mahasiswa PhD di MIT pada tahun 1959, saya melihat sekeliling dan menemukan bahwa sebagian besar teman sekelas saya sedang melakukan penelitian di bidang teori informasi dan teori pengkodean. Di MIT, ada peneliti hebat, Claude Shannon, yang telah meluncurkan bidang ini dan telah memecahkan sebagian besar masalah penting. Masalah penelitian yang tersisa sulit dan konsekuensinya kurang. Jadi saya memutuskan untuk memulai di area baru yang belum pernah terpikirkan oleh orang lain. Ingatlah bahwa di MIT saya dikelilingi oleh banyak komputer, dan jelas bagi saya bahwa mesin-mesin ini akan segera perlu berkomunikasi satu sama lain. Pada saat itu, tidak ada cara yang efektif bagi mereka untuk melakukannya, jadi saya memutuskan untuk mengembangkan teknologi yang memungkinkan terciptanya jaringan data yang efisien dan andal.

Apa pekerjaan pertama Anda di industri komputer? Apa artinya itu?

Saya pergi ke sesi malam di CCNY dari tahun 1951 hingga 1957 untuk mendapatkan gelar sarjana saya di bidang teknik elektro. Pada siang hari, saya bekerja pertama sebagai teknisi dan kemudian sebagai insinyur di sebuah perusahaan elektronik industri kecil bernama Photobell. Selama di sana, saya memperkenalkan teknologi digital ke lini produk mereka. Pada dasarnya, kami menggunakan perangkat fotolistrik untuk mendeteksi keberadaan benda tertentu (kotak, orang, dll.) dan penggunaan sirkuit yang kemudian dikenal sebagai multivibrator *bistabil* adalah jenis teknologi yang kami butuhkan untuk membawa pemrosesan digital ke bidang ini. deteksi. Sirkuit ini kebetulan menjadi blok bangunan untuk komputer, dan kemudian dikenal sebagai *flip-flop* atau *sakelar* dalam bahasa sehari-hari.

Apa yang terlintas dalam pikiran Anda ketika Anda mengirim pesan host-to-host pertama (dari UCLA ke Stanford Research Institute)?

Terus terang, kami tidak tahu pentingnya acara itu. Kami belum menyiapkan pesan khusus yang penting secara historis, seperti yang dilakukan oleh begitu banyak penemu di masa lalu (Samuel Morse dengan "Apa yang telah Tuhan tempa." atau Alexander Graham Bell dengan "Watson, kemarilah! Aku menginginkanmu." atau Neal Armstrong dengan "Itu satu langkah kecil untuk seorang pria, satu lompatan besar bagi umat manusia.") Orang-orang itu *cerdas!* Mereka memahami media dan hubungan masyarakat. Yang ingin kami lakukan hanyalah masuk ke komputer SRI. Jadi kami mengetik "L", yang diterima dengan benar, kami mengetik "o" yang diterima, dan kemudian kami mengetik "g" yang menyebabkan komputer host SRI

menabrak! Nah, ternyata pesan kami adalah pesan terpendek dan mungkin paling profetik yang pernah ada, yaitu "Lo!" seperti dalam "Lihat dan lihatlah!"

Awal tahun itu, saya dikutip dalam siaran pers UCLA yang mengatakan bahwa setelah jaringan aktif dan berjalan, akan mungkin untuk mendapatkan akses ke utilitas komputer dari rumah dan kantor kita semudah kita mendapatkan akses ke sambungan listrik dan telepon. Jadi visi saya saat itu adalah Internet akan ada di mana-mana, selalu aktif, selalu tersedia, siapa pun dengan perangkat apa pun dapat terhubung dari lokasi mana pun, dan tidak akan terlihat. Namun, saya tidak pernah mengantisipasi bahwa ibu saya yang berusia 99 tahun akan menggunakan Internet—and memang demikian!

Apa visi Anda untuk masa depan jaringan?

Bagian yang mudah dari visi ini adalah memprediksi infrastruktur itu sendiri. Saya mengantisipasi bahwa kita melihat penerapan komputasi nomaden, perangkat seluler, dan ruang pintar yang cukup besar. Memang, ketersediaan perangkat komputasi dan komunikasi yang ringan, murah, berperforma tinggi, dan portabel (ditambah keberadaan Internet di mana-mana) telah memungkinkan kita menjadi pengembara. Komputasi nomaden mengacu pada teknologi yang memungkinkan pengguna akhir yang melakukan perjalanan dari satu tempat ke tempat lain untuk mendapatkan akses ke layanan Internet secara transparan, ke mana pun mereka bepergian dan perangkat apa pun yang mereka bawa atau dapatkan aksesnya. Bagian tersulit dari visi ini adalah memprediksi aplikasi dan layanan, yang secara konsisten mengejutkan kami dengan cara yang dramatis (email, teknologi pencarian, world-wide-web, blog, jejaring sosial, pembuatan pengguna, dan berbagi musik, foto, dan video, dll). Kami berada di ambang kelas baru aplikasi seluler yang mengejutkan dan inovatif yang dikirimkan ke perangkat genggam kami.

Langkah selanjutnya akan memungkinkan kita untuk keluar dari dunia bawah dunia maya ke dunia fisik ruang cerdas. Lingkungan kita (meja, dinding, kendaraan, jam tangan, ikat pinggang, dan sebagainya) akan menjadi hidup dengan teknologi, melalui aktuator, sensor, logika, pemrosesan, penyimpanan, kamera, mikrofon, speaker, layar, dan komunikasi. Teknologi tertanam ini akan memungkinkan lingkungan kita menyediakan layanan IP yang kita inginkan. Ketika saya masuk ke sebuah ruangan, ruangan itu akan tahu saya masuk. Saya akan dapat berkomunikasi dengan lingkungan saya secara alami, seperti dalam bahasa Inggris lisan; permintaan saya akan menghasilkan balasan yang menampilkan halaman Web kepada saya dari pajangan dinding, melalui kacamata saya, sebagai ucapan, hologram, dan sebagainya.

Melihat sedikit lebih jauh, saya melihat masa depan jaringan yang mencakup addi berikut komponen kunci nasional. Saya melihat agen perangkat lunak cerdas yang disebarluaskan di seluruh jaringan yang fungsinya untuk menambang data, menindaklanjuti data tersebut, mengamati tren, dan menjalankan tugas secara dinamis dan adaptif. Saya melihat lebih banyak lalu lintas jaringan yang dihasilkan tidak begitu banyak oleh manusia, tetapi oleh perangkat yang disematkan ini dan agen perangkat lunak cerdas ini. Saya melihat kumpulan besar sistem pengaturan mandiri yang mengendalikan jaringan yang luas dan cepat ini. Saya melihat sejumlah besar informasi melintas di jaringan ini secara instan dengan informasi ini menjalani pemrosesan dan penyaringan yang sangat besar. Internet pada dasarnya akan menjadi sistem saraf global yang meresap. Saya melihat semua hal ini dan lebih banyak lagi saat kita bergerak cepat melewati abad kedua puluh satu.

Siapa orang yang telah menginspirasi Anda secara profesional?

Sejauh ini, Claude Shannon dari MIT, seorang peneliti brilian yang memiliki kemampuan untuk menghubungkan ide matematikanya dengan dunia fisik dengan cara yang sangat intuitif. Dia berada di komite tesis PhD saya.

Apakah Anda punya saran untuk siswa memasuki bidang jaringan / Internet?

Internet dan semua yang dimungkinkannya merupakan perbatasan baru yang luas, penuh dengan tantangan luar biasa. Ada ruang untuk inovasi besar. Jangan terkekang oleh teknologi saat ini. Jangkau dan bayangkan apa yang bisa terjadi dan kemudian wujudkan.



BAB

2

Aplikasi Lapisan

Aplikasi jaringan adalah *raisons d'être* dari jaringan komputer—jika kita tidak dapat membayangkan aplikasi yang berguna, tidak akan ada kebutuhan untuk protokol jaringan yang mendukung aplikasi ini. Sejak dimulainya Internet, banyak aplikasi yang berguna dan menghibur memang telah dibuat. Aplikasi ini telah menjadi kekuatan pendorong di balik kesuksesan Internet, memotivasi orang-orang di rumah, sekolah, pemerintah, dan bisnis untuk menjadikan Internet sebagai bagian integral dari aktivitas sehari-hari mereka.

Aplikasi Internet termasuk aplikasi berbasis teks klasik yang menjadi populer pada tahun 1970-an dan 1980-an: email teks, akses jarak jauh ke komputer, transfer file, dan newsgroup. Mereka termasuk *aplikasi pembunuh* pada pertengahan 1990-an, World Wide Web, yang mencakup penjelajahan Web, pencarian, dan perdagangan elektronik. Itu termasuk pesan instan dan berbagi file P2P, dua aplikasi pembunuh yang diperkenalkan pada akhir milenium. Sejak tahun 2000, kami telah melihat ledakan aplikasi suara dan video populer, termasuk: voice-over-IP (VoIP) dan konferensi video melalui IP seperti Skype; distribusi video buatan pengguna seperti YouTube; dan film sesuai permintaan seperti Netflix. Selama periode yang sama ini kami juga telah melihat munculnya game online multi-pemain yang sangat menarik, termasuk Second Life dan World of Warcraft. Dan baru-baru ini, kita telah melihat munculnya generasi baru aplikasi jejaring sosial, seperti Facebook dan Twitter, yang telah menciptakan jaringan manusia yang menarik di atas jaringan router dan tautan komunikasi Internet. Jelas, belum ada perlambatan baru

84 BAB 2 • LAPISAN APLIKASI

dan aplikasi Internet yang menarik. Mungkin beberapa dari pembaca teks ini akan menciptakan aplikasi Internet pembunuh generasi berikutnya!

Dalam bab ini kita mempelajari aspek konseptual dan implementasi aplikasi jaringan. Kita mulai dengan mendefinisikan konsep lapisan aplikasi utama, termasuk layanan jaringan yang dibutuhkan oleh aplikasi, klien dan server, proses, dan antarmuka lapisan transport. Kami memeriksa beberapa aplikasi jaringan secara rinci, termasuk Web, e-mail, DNS, dan distribusi file peer-to-peer (P2P) (Bab 8 berfokus pada aplikasi multimedia, termasuk video streaming dan VoIP). Kami kemudian membahas pengembangan aplikasi jaringan, melalui TCP dan UDP. Secara khusus, kami mempelajari API soket dan menelusuri beberapa aplikasi server-klien sederhana dengan Python. Kami juga memberikan beberapa tugas pemrograman soket yang menyenangkan dan menarik di akhir bab.

Lapisan aplikasi adalah tempat yang sangat baik untuk memulai studi protokol kami. Ini tanah yang akrab. Kami mengenal banyak aplikasi yang bergantung pada protokol yang akan kami pelajari. Ini akan memberi kita pemahaman yang baik tentang semua protokol dan akan memperkenalkan kita pada banyak masalah yang sama yang akan kita lihat lagi ketika kita mempelajari protokol trans port, jaringan, dan lapisan tautan.

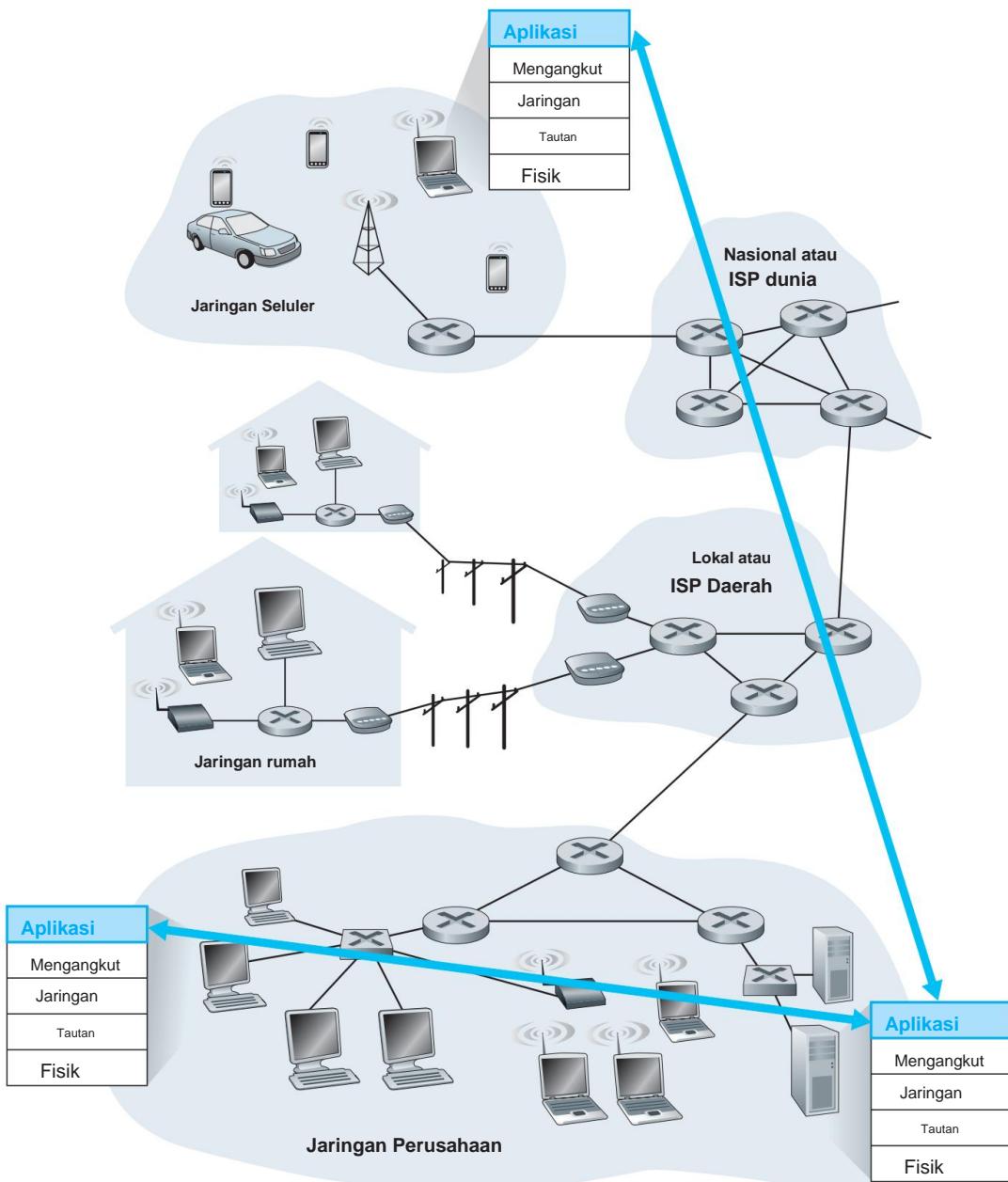
2.1 Prinsip Aplikasi Jaringan

Misalkan Anda memiliki ide untuk aplikasi jaringan baru. Mungkin aplikasi ini akan menjadi layanan yang luar biasa bagi umat manusia, atau akan menyenangkan profesor Anda, atau akan memberi Anda kekayaan besar, atau hanya menyenangkan untuk dikembangkan. Apa pun motivasinya, sekarang mari kita periksa bagaimana Anda mengubah ide menjadi aplikasi jaringan dunia nyata.

Inti dari pengembangan aplikasi jaringan adalah menulis program yang berjalan pada sistem akhir yang berbeda dan berkomunikasi satu sama lain melalui jaringan. Misalnya, di aplikasi Web ada dua program berbeda yang berkomunikasi satu sama lain: program browser berjalan di host pengguna (desktop, laptop, tablet, smartphone, dan sebagainya); dan program server Web berjalan di host server Web. Sebagai contoh lain, dalam sistem file-sharing P2P terdapat program di setiap host yang berpartisipasi dalam komunitas file-sharing. Dalam hal ini, program di berbagai host mungkin serupa atau identik.

Jadi, saat mengembangkan aplikasi baru Anda, Anda perlu menulis perangkat lunak yang akan berjalan di banyak sistem akhir. Perangkat lunak ini dapat ditulis, misalnya, dalam bahasa C, Java, atau Python. Yang penting, Anda tidak perlu menulis perangkat lunak yang berjalan di perangkat inti jaringan, seperti router atau saklar lapisan tautan. Bahkan jika Anda ingin menulis perangkat lunak aplikasi untuk perangkat inti jaringan ini, Anda tidak akan dapat melakukannya. Seperti yang kita pelajari di Bab 1, dan seperti yang ditunjukkan sebelumnya di Gambar 1.24, perangkat inti jaringan tidak berfungsi pada lapisan aplikasi melainkan berfungsi pada lapisan bawah —khususnya pada lapisan jaringan dan di bawahnya. Desain dasar ini—yaitu, membatasi perangkat lunak aplikasi ke sistem akhir—seperti yang ditunjukkan pada Gambar 2.1, telah memfasilitasi pengembangan dan penerapan yang cepat dari rangkaian aplikasi jaringan yang luas.

2.1 • PRINSIP APLIKASI JARINGAN 85



Gambar 2.1 Komunikasi untuk aplikasi jaringan terjadi antara sistem akhir pada lapisan aplikasi

86 BAB 2 • LAPISAN APLIKASI

2.1.1 Arsitektur Aplikasi Jaringan

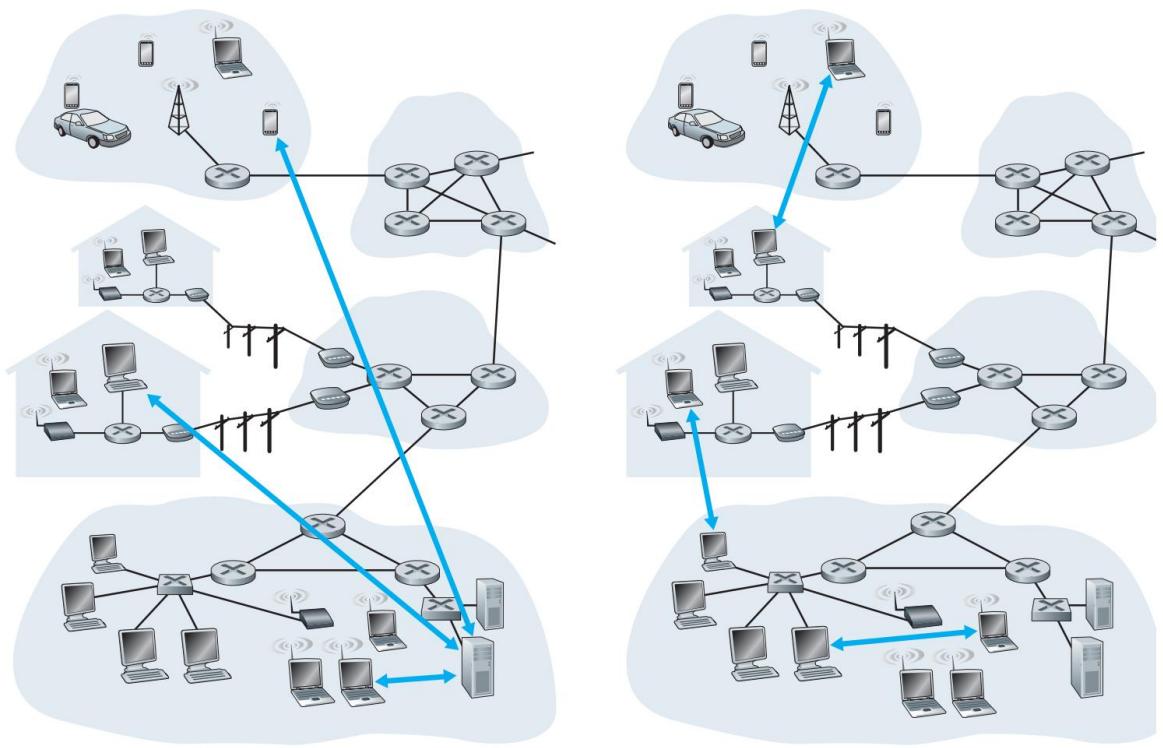
Sebelum mendalami pengkodean perangkat lunak, Anda harus memiliki rencana arsitektur yang luas untuk aplikasi Anda. Perlu diingat bahwa arsitektur aplikasi jelas berbeda dari arsitektur jaringan (misalnya, arsitektur Internet lima lapis yang dibahas dalam Bab 1). Dari perspektif pengembang aplikasi, arsitektur jaringan diperbaiki dan menyediakan serangkaian layanan khusus untuk aplikasi. Arsitektur **aplikasi**, di sisi lain, dirancang oleh pengembang aplikasi dan menentukan bagaimana aplikasi disusun pada berbagai sistem akhir. Dalam memilih arsitektur aplikasi, pengembang aplikasi kemungkinan akan menggunakan salah satu dari dua paradigma arsitektur utama yang digunakan dalam aplikasi jaringan modern: arsitektur client-server atau arsitektur peer-to-peer (P2P) Dalam arsitektur client-server , ada adalah host yang selalu aktif, yang disebut *server*, yang

meminta layanan dari banyak host lain, yang disebut *klien*. Contoh klasik adalah aplikasi Web yang permintaan layanan server Web selalu aktif dari browser yang berjalan di host klien. Ketika server Web menerima permintaan untuk suatu objek dari host klien, ia merespons dengan mengirimkan objek yang diminta ke host klien. Perhatikan bahwa dengan arsitektur client-server, klien tidak langsung berkomunikasi satu sama lain; misalnya, dalam aplikasi Web, dua browser tidak berkomunikasi secara langsung. Karakteristik lain dari arsitektur client-server adalah bahwa server memiliki alamat tetap yang dikenal, yang disebut alamat IP (yang akan segera kita bahas). Karena server memiliki alamat tetap dan terkenal, dan karena server selalu aktif, klien selalu dapat menghubungi server dengan mengirimkan paket ke alamat IP server.

Beberapa aplikasi yang lebih terkenal dengan arsitektur client-server termasuk Web, FTP, Telnet, dan email. Arsitektur client-server ditunjukkan pada Gambar 2.2(a).

Seringkali dalam aplikasi client-server, host server tunggal tidak mampu memenuhi semua permintaan dari klien. Misalnya, situs jejaring sosial yang populer dapat dengan cepat kewalahan jika hanya memiliki satu server yang menangani semua permintaannya. Untuk alasan ini, **pusat data**, menampung sejumlah besar host, sering digunakan untuk membuat server virtual yang kuat. Layanan Internet paling populer—seperti mesin pencari (mis. Google dan Bing), perdagangan Internet (mis. Amazon dan e-Bay), email berbasis web (mis. Gmail dan Yahoo Mail), jejaring sosial (mis. Facebook dan Twitter)—mempekerjakan satu atau lebih pusat data. Seperti dibahas di Bagian 1.3.3, Google memiliki 30 hingga 50 pusat data yang tersebar di seluruh dunia, yang secara kolektif menangani penelusuran, YouTube, Gmail, dan layanan lainnya. Pusat data dapat memiliki ratusan ribu server, yang harus diberdayakan dan dipelihara. Selain itu, penyedia layanan harus membayar interkoneksi berulang dan biaya bandwidth untuk mengirim data dari pusat data mereka.

Dalam **arsitektur P2P**, ketergantungan minimal (atau tidak ada) pada server khusus di pusat data. Alih-alih, aplikasi mengeksplorasi komunikasi langsung antara pasangan host yang terhubung sebentar-sebentar, yang disebut *peer*. Rekan tidak dimiliki oleh penyedia layanan, melainkan desktop dan laptop yang dikendalikan oleh pengguna, dengan sebagian besar rekan berada di rumah, universitas, dan kantor. Karena peer berkomunikasi tanpa melalui dedicated server, arsitekturnya disebut peer-to-peer. Banyak aplikasi paling populer dan padat lalu lintas saat ini didasarkan pada arsitektur P2P. Aplikasi ini termasuk berbagi file (misalnya, BitTorrent), dibantu rekan



Gambar 2.2 (a) Arsitektur klien-server; (b) arsitektur P2P

akselerasi unduhan (mis. Xunlei), Internet Telephony (mis. Skype), dan IPTV (mis. Kankan dan PPstream). Arsitektur P2P diilustrasikan pada Gambar 2.2(b). Kami menyebutkan bahwa beberapa aplikasi memiliki arsitektur hybrid, menggabungkan elemen client-server dan P2P. Misalnya, untuk banyak aplikasi perpesanan instan, server digunakan untuk melacak alamat IP pengguna, tetapi pesan pengguna-ke-pengguna dikirim langsung antara host pengguna (tanpa melewati server perantara).

Salah satu fitur yang paling menarik dari arsitektur P2P adalah skalabilitasnya **sendiri**. Misalnya, dalam aplikasi berbagi file P2P, meskipun setiap peer menghasilkan beban kerja dengan meminta file, setiap peer juga menambahkan kapasitas layanan ke sistem dengan mendistribusikan file ke peer lainnya. Arsitektur P2P juga hemat biaya, karena biasanya tidak memerlukan infrastruktur server dan bandwidth server yang signifikan (berbeda dengan desain klien-server dengan pusat data). Namun, aplikasi P2P di masa mendatang menghadapi tiga tantangan utama:

1. *Ramah ISP*. Sebagian besar ISP perumahan (termasuk DSL dan ISP kabel) telah diukur untuk penggunaan bandwidth "asimetris", yaitu untuk lebih banyak lagi

88 BAB 2 • LAPISAN APLIKASI

hilir daripada lalu lintas hulu. Tetapi aplikasi streaming video dan distribusi file P2P mengalihkan lalu lintas upstream dari server ke ISP perumahan, sehingga memberikan tekanan yang signifikan pada ISP. Aplikasi P2P masa depan perlu dirancang agar ramah terhadap ISP [Xie 2008].

2. *Keamanan.* Karena sifatnya yang sangat terdistribusi dan terbuka, aplikasi P2P bisa menjadi tantangan untuk mengamankan [Doucer 2002; Yu 2006; Liang 2006; Naoumov 2006; Dhungel 2008; LeBlond 2011].
3. *Insetif.* Kesuksesan aplikasi P2P di masa depan juga bergantung pada meyakinkan pengguna untuk merelakan bandwidth, penyimpanan, dan sumber daya komputasi ke aplikasi, yang merupakan tantangan desain insetif [Feldman 2005; Piatek 2008; Aperjis 2008; Liu 2010].

2.1.2 Proses Berkomunikasi

Sebelum membangun aplikasi jaringan Anda, Anda juga memerlukan pemahaman dasar tentang bagaimana program, yang berjalan di banyak sistem ujung, berkomunikasi satu sama lain. Dalam jargon sistem operasi, sebenarnya bukan program melainkan **proses** yang berkomunikasi. Suatu proses dapat dianggap sebagai program yang berjalan di dalam sistem akhir. Ketika proses berjalan pada sistem akhir yang sama, mereka dapat berkomunikasi satu sama lain dengan komunikasi antarproses, menggunakan aturan yang diatur oleh sistem operasi sistem akhir. Namun dalam buku ini kami tidak secara khusus tertarik pada bagaimana proses dalam host yang sama berkomunikasi, melainkan pada bagaimana proses yang berjalan pada host *yang berbeda* (dengan sistem operasi yang berpotensi berbeda) berkomunikasi.

Proses pada dua sistem akhir yang berbeda berkomunikasi satu sama lain dengan bertukar **pesan** di jaringan komputer. Proses pengiriman membuat dan mengirimkan pesan ke jaringan; proses penerima menerima pesan-pesan ini dan mungkin merespons dengan mengirim pesan kembali. Gambar 2.1 mengilustrasikan bahwa proses yang berkomunikasi satu sama lain berada di lapisan aplikasi tumpukan protokol lima lapis.

Proses Klien dan Server

Aplikasi jaringan terdiri dari pasangan proses yang mengirim pesan satu sama lain melalui jaringan. Misalnya, dalam aplikasi Web proses browser klien bertukar pesan dengan proses server Web. Dalam sistem berbagi file P2P, file ditransfer dari proses di satu peer ke proses di peer lain.

Untuk setiap pasangan proses komunikasi, kami biasanya memberi label salah satu dari dua proses sebagai **klien** dan proses lainnya sebagai **server**. Dengan Web, browser adalah proses klien dan server Web adalah proses server. Dengan berbagi file P2P, rekan yang mengunduh file diberi label sebagai klien, dan rekan yang mengunggah file diberi label sebagai server.

Anda mungkin telah mengamati bahwa dalam beberapa aplikasi, seperti berbagi file P2P, suatu proses dapat berupa klien dan server. Memang, proses dalam sistem berbagi file P2P dapat mengunggah dan mengunduh file. Namun demikian, dalam konteks apa pun yang diberikan

2.1 • PRINSIP APLIKASI JARINGAN 89

sesi komunikasi antara sepasang proses, kita masih bisa melabeli satu proses sebagai klien dan proses lainnya sebagai server. Kami mendefinisikan proses klien dan server sebagai berikut:

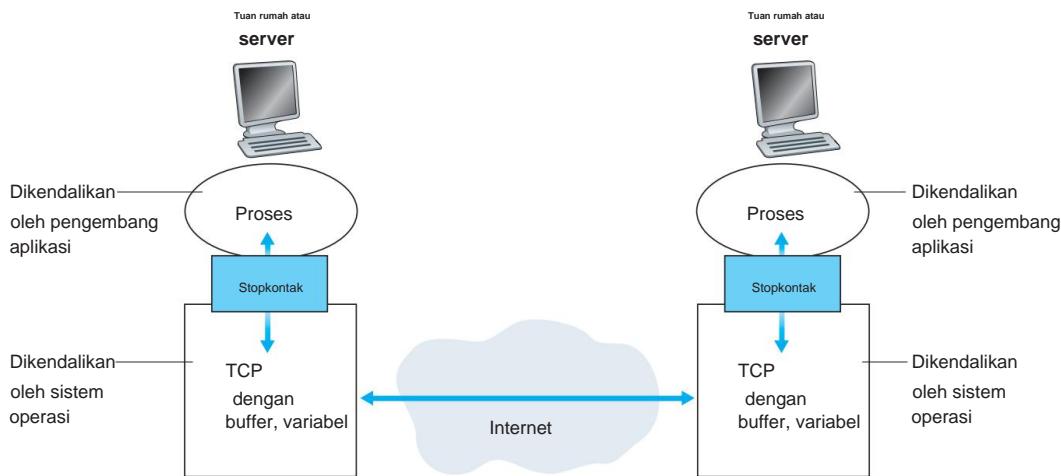
Dalam konteks sesi komunikasi antara sepasang proses, proses yang memulai komunikasi (yaitu, awalnya menghubungi proses lain di awal sesi) diberi label sebagai klien . Proses yang menunggu untuk dihubungi untuk memulai sesi adalah server.

Di Web, proses browser menginisialisasi kontak dengan proses server Web; karenanya proses browser adalah klien dan proses server Web adalah server. Dalam berbagi file P2P, saat Peer A meminta Peer B untuk mengirim file tertentu, Peer A adalah klien dan Peer B adalah server dalam konteks sesi komunikasi khusus ini. Jika tidak ada kebingungan, terkadang kami juga akan menggunakan terminologi "sisi klien dan sisi server aplikasi". Di akhir bab ini, kita akan menelusuri kode sederhana untuk sisi klien dan server aplikasi jaringan.

Antarmuka Antara Proses dan Jaringan Komputer

Seperti disebutkan di atas, sebagian besar aplikasi terdiri dari pasangan proses komunikasi, dengan dua proses di setiap pasangan saling mengirim pesan. Setiap pesan yang dikirim dari satu proses ke proses lainnya harus melalui jaringan yang mendasarinya. Suatu proses mengirimkan pesan ke dalam, dan menerima pesan dari, jaringan melalui antarmuka perangkat lunak yang disebut **soket**. Mari pertimbangkan analogi untuk membantu kita memahami proses dan soket. Suatu proses dianalogikan dengan sebuah rumah dan soketnya dianalogikan dengan pintunya. Ketika suatu proses ingin mengirim pesan ke proses lain di host lain, ia mendorong pesan keluar dari pintunya (soket). Proses pengiriman ini mengasumsikan bahwa ada infrastruktur transportasi di sisi lain pintunya yang akan mengangkut pesan ke pintu proses tujuan. Setelah pesan tiba di host tujuan, pesan melewati pintu (soket) proses penerima, dan proses penerima kemudian bertindak atas pesan tersebut. Gambar 2.3 mengilustrasikan komunikasi soket antara dua proses yang berkomunikasi melalui Internet. (Gambar 2.3 mengasumsikan bahwa protokol transport dasar yang digunakan oleh proses adalah protokol TCP Internet.) Seperti yang ditunjukkan pada gambar ini, soket adalah antarmuka antara lapisan aplikasi dan lapisan transport di dalam sebuah host. Ini juga disebut sebagai **Antarmuka Pemrograman Aplikasi (API)** antara aplikasi dan jaringan, karena soket adalah antarmuka pemrograman tempat aplikasi jaringan dibangun. Pengembang aplikasi memiliki kendali atas semua yang ada di sisi lapisan aplikasi soket tetapi memiliki sedikit kendali atas sisi lapisan transportasi soket. Satu-satunya kontrol yang dimiliki oleh pengembang aplikasi pada sisi transport-layer adalah (1) pilihan protokol transport dan (2) mungkin kemampuan untuk memperbaiki beberapa parameter transport-layer seperti buffer maksimum dan ukuran segmen maksimum (yang akan dicakup) dalam Bab 3). Setelah pengembang aplikasi memilih protokol transport (jika pilihan tersedia),

90 BAB 2 • LAPISAN APLIKASI



Gambar 2.3 Proses aplikasi, soket, dan protokol transport yang mendasarinya

aplikasi dibangun menggunakan layanan transport-layer yang disediakan oleh protokol itu. Kami akan menjelajahi soket secara mendetail di Bagian 2.7.

Proses Pengalaman

Untuk mengirim surat pos ke tujuan tertentu, tujuan tersebut harus memiliki alamat. Demikian pula, agar proses yang berjalan di satu host mengirim paket ke proses yang berjalan di host lain, proses penerima harus memiliki alamat. Untuk mengidentifikasi proses penerima, dua informasi perlu ditentukan: (1) alamat host dan (2) pengidentifikasi yang menentukan proses penerimaan di host tujuan.

Di Internet, host diidentifikasi oleh **alamat IP-nya**. Kita akan membahas alamat IP dengan sangat rinci di Bab 4. Untuk saat ini, yang perlu kita ketahui hanyalah bahwa alamat IP adalah kuantitas 32-bit yang dapat kita anggap sebagai pengidentifikasi host secara unik. Selain mengetahui alamat host tujuan pesan, proses pengiriman juga harus mengidentifikasi proses penerima (lebih spesifik, soket penerima) yang berjalan di host. Informasi ini diperlukan karena pada umumnya sebuah host dapat menjalankan banyak aplikasi jaringan.

Nomor port tujuan melayani tujuan ini. Aplikasi populer telah diberi nomor port tertentu. Misalnya, server Web diidentifikasi dengan nomor port 80. Proses server email (menggunakan protokol SMTP) diidentifikasi dengan nomor port 25. Daftar nomor port terkenal untuk semua protokol standar Internet dapat ditemukan di <http://www.iana.org>. Kami akan memeriksa nomor port secara rinci di Bab 3.

2.1.3 Layanan Transportasi yang Tersedia untuk Aplikasi

Ingin bahwa soket adalah antarmuka antara proses aplikasi dan protokol transport-layer. Aplikasi di sisi pengirim mendorong pesan melalui soket. Di sisi lain soket, protokol transport-layer memiliki tanggung jawab untuk mengirimkan pesan ke soket proses penerima.

Banyak jaringan, termasuk Internet, menyediakan lebih dari satu protokol transport-layer. Saat Anda mengembangkan aplikasi, Anda harus memilih salah satu protokol lapisan transport yang tersedia. Bagaimana Anda membuat pilihan ini? Kemungkinan besar, Anda akan mempelajari layanan yang disediakan oleh protokol lapisan transport yang tersedia, lalu memilih protokol dengan layanan yang paling sesuai dengan kebutuhan aplikasi Anda. Situasinya mirip dengan memilih transportasi kereta api atau pesawat terbang untuk perjalanan antara dua kota. Anda harus memilih satu atau yang lain, dan setiap moda transportasi menawarkan layanan yang berbeda. (Misalnya, kereta menawarkan penjemputan dan pengantaran ke pusat kota, sedangkan pesawat menawarkan waktu perjalanan yang lebih singkat.)

Apa saja layanan yang dapat ditawarkan oleh protokol lapisan transport ke aplikasi yang memintanya? Kami dapat secara luas mengklasifikasikan layanan yang mungkin dalam empat dimensi: transfer data yang andal, throughput, waktu, dan keamanan.

Transfer Data Andal

Seperti dibahas dalam Bab 1, paket bisa hilang dalam jaringan komputer. Misalnya, sebuah paket dapat meluap dari buffer di router, atau dapat dibuang oleh host atau router setelah beberapa bitnya rusak. Untuk banyak aplikasi—seperti surat elektronik, transfer file, akses host jarak jauh, transfer dokumen Web, dan aplikasi keuangan—kehilangan data dapat memiliki konsekuensi yang menghancurkan (dalam kasus terakhir, baik untuk bank atau pelanggan!). Oleh karena itu, untuk mendukung aplikasi ini, sesuatu harus dilakukan untuk menjamin bahwa data yang dikirim oleh salah satu ujung aplikasi terkirim dengan benar dan lengkap ke ujung aplikasi yang lain. Jika sebuah protokol menyediakan layanan pengiriman data yang terjamin, dikatakan menyediakan **transfer data yang andal**. Salah satu layanan penting yang berpotensi disediakan oleh protokol lapisan transport untuk aplikasi adalah transfer data proses-ke-proses yang andal.

Ketika protokol transport menyediakan layanan ini, proses pengiriman hanya dapat meneruskan datanya ke soket dan mengetahui dengan keyakinan penuh bahwa data akan sampai tanpa kesalahan pada proses penerimaan.

Ketika protokol transport-layer tidak menyediakan transfer data yang andal, beberapa data yang dikirim oleh proses pengiriman mungkin tidak akan pernah sampai ke proses penerima. Ini mungkin dapat diterima untuk **aplikasi yang toleran terhadap kehilangan**, terutama aplikasi multimedia seperti audio/video percakapan yang dapat mentolerir sejumlah kehilangan data.

Dalam aplikasi multimedia ini, data yang hilang dapat menyebabkan kesalahan kecil pada audio/video—bukan gangguan yang krusial.

92 BAB 2 • LAPISAN APLIKASI

Hasil

Dalam Bab 1 kami memperkenalkan konsep throughput yang tersedia, yang, dalam konteks sesi komunikasi antara dua proses di sepanjang jalur jaringan, adalah laju di mana proses pengiriman dapat mengirimkan bit ke proses penerima.

Karena sesi lain akan berbagi bandwidth di sepanjang jalur jaringan, dan karena sesi lain ini akan datang dan pergi, throughput yang tersedia dapat berfluktuasi seiring waktu. Pengamatan ini mengarah ke layanan alami lain yang dapat disediakan oleh protokol lapisan transport, yaitu, throughput yang tersedia terjamin pada tingkat tertentu. Dengan layanan seperti itu, aplikasi dapat meminta throughput yang dijamin sebesar r bit/detik, dan protokol transport kemudian akan memastikan bahwa throughput yang tersedia selalu setidaknya r bit/detik. Layanan put yang dijamin seperti itu akan menarik bagi banyak aplikasi. Misalnya, jika aplikasi telepon Internet menyandikan suara pada 32 kbps, aplikasi tersebut perlu mengirim data ke jaringan dan mengirim data ke aplikasi penerima dengan kecepatan ini. Jika protokol transport tidak dapat menyediakan throughput ini, aplikasi perlu menyandikan pada kecepatan yang lebih rendah (dan menerima throughput yang cukup untuk mempertahankan kecepatan pengkodean yang lebih rendah ini) atau mungkin harus menyerah, karena menerima, katakanlah, setengah dari throughput yang dibutuhkan adalah dari sedikit atau tidak ada gunanya untuk aplikasi telepon Internet ini. Aplikasi yang memiliki persyaratan throughput dikatakan sebagai **aplikasi sensitif bandwidth**. Banyak aplikasi multimedia saat ini sensitif bandwidth, meskipun beberapa aplikasi multimedia mungkin menggunakan teknik pengkodean adaptif untuk menyandikan suara atau video digital pada kecepatan yang sesuai dengan throughput yang tersedia saat ini.

Sementara aplikasi sensitif bandwidth memiliki kebutuhan throughput khusus, **aplikasi elastis** dapat menggunakan throughput sebanyak, atau sesedikit yang tersedia. Surat elektronik, transfer file, dan transfer Web semuanya adalah aplikasi elastis. Tentu saja, semakin banyak throughput, semakin baik. Ada pepatah yang mengatakan bahwa seseorang tidak boleh terlalu kaya, terlalu kurus, atau memiliki hasil yang terlalu banyak!

Pengaturan waktu

Protokol transport-layer juga dapat memberikan jaminan waktu. Seperti jaminan throughput, jaminan pengaturan waktu dapat datang dalam berbagai bentuk dan bentuk. Contoh jaminan mungkin bahwa setiap bit yang dipompa pengirim ke soket tiba di soket penerima tidak lebih dari 100 ms kemudian. Layanan seperti itu akan menarik bagi aplikasi interaktif waktu nyata, seperti telepon Internet, lingkungan virtual, telekonferensi, dan permainan multipemain, yang semuanya memerlukan batasan waktu yang ketat pada pengiriman data agar efektif. (Lihat Bab 7, [Gauthier 1999; Ramjee 1994].) Penundaan yang lama dalam telepon Internet, misalnya, cenderung menghasilkan jeda percakapan yang tidak wajar; dalam game multipemain atau lingkungan interaktif virtual, penundaan yang lama antara mengambil tindakan dan melihat respons dari lingkungan (misalnya, dari pemain lain di akhir koneksi end-to-end) membuat aplikasi terasa kurang realistik. Untuk aplikasi non-real-time,

2.1 • PRINSIP APLIKASI JARINGAN 93

penundaan yang lebih rendah selalu lebih disukai daripada penundaan yang lebih tinggi, tetapi tidak ada kendala ketat yang ditempatkan pada penundaan ujung ke ujung.

Keamanan

Terakhir, protokol transport dapat menyediakan aplikasi dengan satu atau lebih layanan keamanan. Misalnya, di host pengirim, protokol transport dapat mengenkripsi semua data yang dikirimkan oleh proses pengiriman, dan di host penerima, protokol transport-layer dapat mendekripsi data sebelum mengirimkan data ke proses penerima.

Layanan seperti itu akan memberikan kerahasiaan antara dua proses, bahkan jika data diamati antara proses pengiriman dan penerimaan. Protokol transport juga dapat memberikan layanan keamanan lain selain kerahasiaan, termasuk integritas data dan autentikasi titik akhir, topik yang akan kita bahas secara detail di Bab 8.

2.1.4 Layanan Transportasi yang Disediakan oleh Internet

Hingga saat ini, kami telah mempertimbangkan layanan transportasi yang *dapat* disediakan oleh jaringan komputer secara umum. Sekarang mari kita lebih spesifik dan memeriksa jenis layanan transportasi yang disediakan oleh Internet. Internet (dan, lebih umum lagi, jaringan TCP/IP) membuat dua protokol transport tersedia untuk aplikasi, UDP dan TCP. Saat Anda (sebagai pengembang aplikasi) membuat aplikasi jaringan baru untuk Internet, salah satu keputusan pertama yang harus Anda buat adalah menggunakan UDP atau TCP. Masing-masing protokol ini menawarkan rangkaian layanan yang berbeda untuk aplikasi pemanggilan. Gambar 2.4 menunjukkan persyaratan layanan untuk beberapa aplikasi terpilih.

Aplikasi	Data hilang	Hasil	Sensitif terhadap waktu
Transfer/unduh file	Tidak rugi	Elastis	TIDAK
Surel	Tidak rugi	Elastis	TIDAK
dokumen web	Tidak rugi	Elastis (beberapa kbps)	TIDAK
telepon internet/ Konferensi video	Toleransi kerugian	Audio: beberapa kbps–1Mbps Video: 10 kbps–5 Mbps	Ya: 100-an mdetik
Streaming audio/ video yang disimpan	Toleransi kerugian	Sama seperti di atas	Ya: beberapa detik
Game interaktif	Toleransi kerugian	Beberapa kbps–10 kbps	Ya: 100-an mdetik
Pesan singkat	Tidak rugi	Elastis	ya dan tidak

Gambar 2.4 Persyaratan aplikasi jaringan yang dipilih

94 BAB 2 • LAPISAN APLIKASI

Layanan TCP

Model layanan TCP mencakup layanan berorientasi koneksi dan layanan transfer data yang andal. Saat aplikasi memanggil TCP sebagai protokol transportnya, aplikasi menerima kedua layanan ini dari TCP.

- *Layanan berorientasi koneksi.* TCP meminta klien dan server bertukar informasi kontrol lapisan transpor satu sama lain *sebelum* pesan tingkat aplikasi mulai mengalir. Prosedur jabat tangan yang disebut ini memberi tahu klien dan server, memungkinkan mereka untuk bersiap menghadapi serangan paket. Setelah fase jabat tangan, **koneksi TCP** dikatakan ada di antara soket kedua proses. Sambungan adalah sambungan full-duplex di mana dua proses dapat mengirim pesan satu sama lain melalui sambungan pada waktu yang sama. Ketika aplikasi selesai mengirim pesan, itu harus memutuskan koneksi. Dalam Bab 3 kita akan membahas layanan berorientasi koneksi secara rinci dan mempelajari bagaimana penerapannya.

FOKUS PADA KEAMANAN

MENGAMANKAN TCP

Baik TCP maupun UDP tidak menyediakan enkripsi apa pun—data yang diteruskan oleh proses pengiriman ke dalam soketnya adalah data yang sama yang dikirimkan melalui jaringan ke proses tujuan. Jadi, misalnya, jika proses pengiriman mengirimkan kata sandi dalam teks-jelas (yaitu, tidak terenkripsi) ke soketnya, kata sandi teks-jelas akan melewati semua tautan antara pengirim dan penerima, berpotensi diendus dan ditemukan di salah satu tautan yang mengintervensi. Karena masalah privasi dan keamanan lainnya menjadi penting untuk banyak aplikasi, komunitas Internet telah mengembangkan peningkatan untuk TCP, yang disebut **Secure Sockets Layer (SSL)**.

TCP-enhanced-with-SSL tidak hanya melakukan semua yang dilakukan TCP tradisional tetapi juga menyediakan layanan keamanan proses-ke-proses yang penting, termasuk enkripsi, integritas data, dan autentikasi titik akhir. Kami menekankan bahwa SSL bukanlah protokol transport Internet ketiga, pada tingkat yang sama dengan TCP dan UDP, melainkan merupakan peningkatan dari TCP, dengan peningkatan yang diterapkan di lapisan aplikasi. Secara khusus, jika sebuah aplikasi ingin menggunakan layanan SSL, itu perlu menyertakan kode SSL (perpustakaan dan kelas yang sudah ada dan sangat optimal) di sisi klien dan server aplikasi. SSL memiliki API soketnya sendiri yang mirip dengan API soket tradisional al TCP. Saat aplikasi menggunakan SSL, proses pengiriman meneruskan data teks-jelas ke soket SSL; SSL di host pengirim kemudian mengenkripsi data dan meneruskan data terenkripsi ke soket TCP. Data terenkripsi berjalan melalui Internet ke soket TCP dalam proses penerimaan. Soket penerima meneruskan data terenkripsi ke SSL, yang mendekripsi data. Terakhir, SSL meneruskan data teks-jelas melalui soket SSL-nya ke proses penerimaan. Kami akan membahas SSL secara mendetail di Bab 8.

2.1 • PRINSIP APLIKASI JARINGAN 95

- *Layanan transfer data yang andal.* Proses komunikasi dapat mengandalkan TCP untuk mengirimkan semua data yang dikirim tanpa kesalahan dan dalam urutan yang benar. Ketika satu sisi aplikasi melewati aliran byte ke soket, itu dapat mengandalkan TCP untuk mengirimkan aliran byte yang sama ke soket penerima, tanpa byte yang hilang atau duplikat.

TCP juga mencakup mekanisme kontrol kemacetan, layanan untuk kesejahteraan umum Internet daripada untuk keuntungan langsung dari proses komunikasi. Mekanisme kontrol kongesti TCP membatasi proses pengiriman (klien atau server) ketika jaringan padat antara pengirim dan penerima. Seperti yang akan kita lihat di Bab 3, kontrol kongesti TCP juga mencoba membatasi setiap koneksi TCP ke bagian bandwidth jaringan yang adil.

Layanan UDP

UDP adalah protokol transport ringan tanpa embel-embel, menyediakan layanan minimal. UDP tidak terhubung, jadi tidak ada jabat tangan sebelum kedua proses mulai berkomunikasi. UDP menyediakan layanan transfer data yang tidak dapat diandalkan—yaitu, ketika suatu proses mengirim pesan ke soket UDP, UDP *tidak* memberikan jaminan bahwa pesan tersebut akan sampai ke proses penerima. Selain itu, pesan yang tiba di proses penerimaan mungkin tiba tidak sesuai urutan.

UDP tidak menyertakan mekanisme kontrol kongesti, sehingga sisi pengirim UDP dapat memompa data ke lapisan di bawahnya (lapisan jaringan) dengan kecepatan berapa pun yang diinginkan. (Perhatikan, bagaimanapun, bahwa throughput end-to-end yang sebenarnya mungkin kurang dari tingkat ini karena kapasitas transmisi yang terbatas dari link intervening atau karena kemacetan).

Layanan yang Tidak Disediakan oleh Protokol Transportasi Internet

Kami telah mengatur layanan protokol transport dalam empat dimensi: transfer data yang andal, throughput, pengaturan waktu, dan keamanan. Manakah dari layanan ini yang disediakan oleh TCP dan UDP? Kami telah mencatat bahwa TCP menyediakan transfer data end-to-end yang andal. Dan kita juga tahu bahwa TCP dapat dengan mudah ditingkatkan pada lapisan aplikasi dengan SSL untuk memberikan layanan keamanan. Namun dalam deskripsi singkat kami tentang TCP dan UDP, yang jelas hilang adalah penyebutan throughput atau jaminan waktu—layanan yang *tidak* disediakan oleh protokol transportasi Internet saat ini. Apakah ini berarti aplikasi sensitif waktu seperti telepon Internet tidak dapat berjalan di Internet saat ini? Jawabannya jelas tidak—Internet telah menghosting aplikasi yang sensitif terhadap waktu selama bertahun-tahun. Aplikasi ini sering bekerja dengan cukup baik karena telah dirancang untuk mengatasi, semaksimal mungkin, dengan kurangnya jaminan ini. Kami akan menyelidiki beberapa trik desain ini di Bab 7. Namun demikian, desain yang cerdas memiliki keterbatasan ketika penundaan berlebihan, atau throughput end-to-end terbatas.

Singkatnya, Internet saat ini seringkali dapat memberikan layanan yang memuaskan untuk aplikasi yang sensitif terhadap waktu, tetapi tidak dapat memberikan jaminan waktu atau throughput apa pun.

96 BAB 2 • LAPISAN APLIKASI

Aplikasi	Protokol Lapisan Aplikasi	Protokol Transportasi yang Mendasari
Surat elektronik	SMTP [RFC 5321]	TCP
Akses terminal jarak jauh	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transfer file	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (misalnya, YouTube)	TCP
telepon internet	SIP [RFC 3261], RTP [RFC 3550], atau berpemilik (misalnya, Skype)	UDP atau TCP

Gambar 2.5 Aplikasi Internet populer, protokol lapisan aplikasinya, dan protokol transport yang mendasarinya

Gambar 2.5 menunjukkan protokol transport yang digunakan oleh beberapa aplikasi Internet populer. Kami melihat bahwa email, akses terminal jarak jauh, Web, dan transfer file semuanya menggunakan TCP. Aplikasi ini memilih TCP terutama karena TCP menyediakan transfer data yang andal, menjamin bahwa semua data pada akhirnya akan sampai ke tujuannya. Karena aplikasi telepon Internet (seperti Skype) seringkali dapat mentolerir beberapa kerugian tetapi membutuhkan tingkat minimal agar efektif, pengembang aplikasi telepon Internet biasanya lebih suka menjalankan aplikasi mereka melalui UDP, sehingga menghindari mekanisme kontrol kemacetan TCP dan paket di atas kepala. Tetapi karena banyak firewall dikonfigurasi untuk memblokir (sebagian besar jenis) lalu lintas UDP, aplikasi telepon Internet sering dirancang untuk menggunakan TCP sebagai cadangan jika komunikasi UDP gagal.

2.1.5 Protokol Lapisan Aplikasi

Kami baru saja mempelajari bahwa proses jaringan berkomunikasi satu sama lain dengan mengirim pesan ke soket. Tetapi bagaimana pesan-pesan ini disusun? Apa arti dari berbagai bidang dalam pesan? Kapan proses mengirim pesan bijak? Pertanyaan-pertanyaan ini membawa kita ke ranah protokol lapisan aplikasi. Protokol **lapisan aplikasi** menentukan bagaimana proses aplikasi, berjalan pada sistem akhir yang berbeda, meneruskan pesan satu sama lain. Secara khusus, protokol lapisan aplikasi mendefinisikan:

- Jenis pesan yang dipertukarkan, misalnya pesan permintaan dan tanggapan pesan
- Sintaks berbagai jenis pesan, seperti bidang dalam pesan dan bagaimana bidang digambarkan

- Semantik dari field, yaitu arti informasi dalam field
- Aturan untuk menentukan kapan dan bagaimana suatu proses mengirim pesan dan merespon pesan

Beberapa protokol lapisan aplikasi ditentukan dalam RFC dan karenanya berada dalam domain publik. Misalnya, protokol lapisan aplikasi Web, HTTP (HyperText Transfer Protocol [RFC 2616]), tersedia sebagai RFC. Jika pengembang browser mengikuti aturan HTTP RFC, browser akan dapat mengambil halaman Web dari server Web mana pun yang juga mengikuti aturan HTTP RFC. Banyak protokol lapisan aplikasi lainnya adalah hak milik dan sengaja tidak tersedia di domain publik. Misalnya, Skype menggunakan protokol lapisan aplikasi berpemilik.

Penting untuk membedakan antara aplikasi jaringan dan protokol lapisan aplikasi. Protokol lapisan aplikasi hanyalah satu bagian dari aplikasi jaringan (walaupun, bagian aplikasi yang sangat penting dari sudut pandang kami!). Mari kita lihat beberapa contoh. Web adalah aplikasi client-server yang memungkinkan pengguna untuk mendapatkan dokumen dari server Web sesuai permintaan. Aplikasi Web terdiri dari banyak komponen, termasuk standar untuk format dokumen (yaitu, HTML), browser Web (misalnya, Firefox dan Microsoft Internet Explorer), server Web (misalnya, server Apache dan Microsoft), dan aplikasi- protokol lapisan. Protokol lapisan aplikasi Web, HTTP, menentukan format dan urutan pesan yang dipertukarkan antara browser dan server Web. Jadi, HTTP hanyalah satu bagian (walaupun merupakan bagian penting) dari aplikasi Web. Sebagai contoh lain, aplikasi email Internet juga memiliki banyak komponen, termasuk server surat yang menampung kotak surat pengguna; klien email (seperti Microsoft Outlook) yang memungkinkan pengguna membaca dan membuat pesan; standar untuk menentukan struktur pesan email; dan protokol lapisan aplikasi yang menentukan bagaimana pesan dikirimkan antar server, bagaimana pesan diteruskan antara server dan klien email, dan bagaimana isi header pesan akan ditafsirkan. Protokol lapisan aplikasi utama untuk surat elektronik adalah SMTP (Simple Mail Transfer Protocol) [RFC 5321]. Dengan demikian, protokol lapisan aplikasi utama email, SMTP, hanyalah satu bagian (walaupun merupakan bagian penting) c

2.1.6 Aplikasi Jaringan yang Dicakup dalam Buku Ini

Domain publik baru dan aplikasi Internet eksklusif sedang dikembangkan setiap hari. Daripada mencakup sejumlah besar aplikasi Internet dengan cara ensiklopedi, kami telah memilih untuk fokus pada sejumlah kecil aplikasi yang tersebar luas dan penting. Dalam bab ini kita membahas lima aplikasi penting: Web, transfer file, surat elektronik, layanan direktori, dan aplikasi P2P. Pertama-tama kita membahas Web, bukan hanya karena ini adalah aplikasi yang sangat populer, tetapi juga karena protokol lapisan aplikasinya, HTTP, sangat lugas dan mudah dipahami. Setelah meliput Web, kami secara singkat memeriksa FTP, karena menyediakan kontras yang bagus dengan HTTP. Kami kemudian membahas surat elektronik, aplikasi mematikan pertama di Internet. E-mail lebih kompleks daripada Web dalam arti tidak menggunakan satu pun

98 BAB 2 • LAPISAN APLIKASI

tetapi beberapa protokol lapisan aplikasi. Setelah email, kami membahas DNS, yang menyediakan layanan direktori untuk Internet. Sebagian besar pengguna tidak berinteraksi dengan DNS secara langsung; sebaliknya, pengguna meminta DNS secara tidak langsung melalui aplikasi lain (termasuk Web, transfer file, dan surat elektronik). DNS mengilustrasikan dengan baik bagaimana sepotong fungsi jaringan inti (terjemahan nama jaringan ke alamat jaringan) dapat diimplementasikan pada lapisan aplikasi di Internet. Terakhir, kami membahas dalam bab ini beberapa aplikasi P2P, dengan fokus pada aplikasi berbagi file, dan layanan pencarian terdistribusi. Di Bab 7, kita akan membahas aplikasi multimedia, termasuk streaming video dan voice-over-IP.

2.2 Web dan HTTP

Sampai awal 1990-an Internet digunakan terutama oleh peneliti, akademisi, dan mahasiswa untuk masuk ke host jarak jauh, untuk mentransfer file dari host lokal ke host jarak jauh dan sebaliknya, untuk menerima dan mengirim berita, dan untuk menerima dan mengirim surat elektronik. . Meskipun aplikasi ini (dan terus) sangat berguna, Internet pada dasarnya tidak dikenal di luar komunitas akademik dan penelitian.

Kemudian, pada awal 1990-an, sebuah aplikasi baru yang besar tiba di tempat—World Wide Web [Berners-Lee 1994]. Web adalah aplikasi Internet pertama yang menarik perhatian masyarakat umum. Itu berubah secara dramatis, dan terus berubah, bagaimana orang berinteraksi di dalam dan di luar lingkungan kerja mereka. Itu meningkatkan Internet dari hanya satu dari banyak jaringan data menjadi satu-satunya jaringan data.

Mungkin yang paling menarik bagi pengguna adalah bahwa Web beroperasi *sesuai permintaan*. Pengguna menerima apa yang mereka inginkan, ketika mereka menginginkannya. Ini tidak seperti radio dan televisi siaran tradisional, yang memaksa pengguna untuk mendengarkan ketika penyedia konten menyediakan konten. Selain tersedia sesuai permintaan, Web memiliki banyak fitur luar biasa lainnya yang disukai dan dihargai orang. Sangatlah mudah bagi setiap individu untuk menyediakan informasi melalui Web—setiap orang dapat menjadi penerbit dengan biaya yang sangat rendah. Hyperlink dan mesin telusur membantu kami menjelajahi lautan situs Web. Grafik merangsang indra kita. Formulir, JavaScript, applet Java, dan banyak perangkat lain memungkinkan kita berinteraksi dengan halaman dan situs. Dan Web berfungsi sebagai platform untuk banyak aplikasi mematikan yang muncul setelah tahun 2003, termasuk YouTube, Gmail, dan Facebook.

2.2.1 Ikhtisar HTTP

HyperText Transfer Protocol (HTTP), protokol lapisan aplikasi Web, adalah jantung dari Web. Ini didefinisikan dalam [RFC 1945] dan [RFC 2616]. HTTP diimplementasikan dalam dua program: program klien dan program server. Program klien dan program server, yang dijalankan pada sistem akhir yang berbeda, berbicara satu sama lain dengan bertukar pesan HTTP. HTTP mendefinisikan struktur pesan-pesan ini dan bagaimana klien dan server bertukar pesan. Sebelum menjelaskan HTTP secara rinci, kita harus meninjau beberapa terminologi Web.

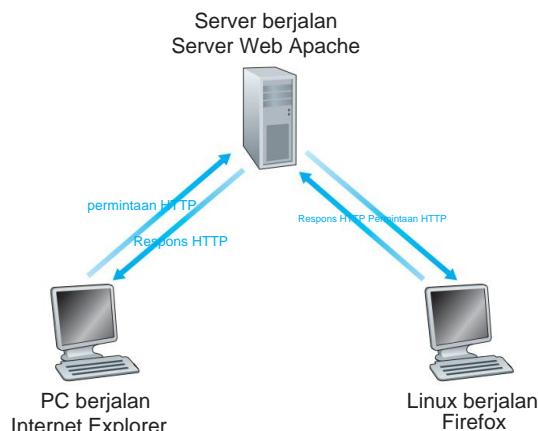
Halaman **Web** (juga disebut dokumen) terdiri dari objek. Objek hanyalah sebuah file—seperti file HTML, gambar JPEG, applet Java, atau klip video—yang dapat dialamatkan oleh satu URL. Sebagian besar halaman Web terdiri dari **file HTML dasar** dan beberapa objek yang direferensikan. Misalnya, jika halaman Web berisi teks HTML dan lima gambar JPEG, maka halaman Web tersebut memiliki enam objek: file dasar HTML ditambah lima gambar. File HTML dasar mereferensikan objek lain di halaman dengan URL objek. Setiap URL memiliki dua komponen: nama host server yang menampung objek dan nama jalur objek. Misalnya, URL

<http://www.someSchool.edu/someDepartment/picture.gif>

memiliki www.someSchool.edu untuk nama host dan /someDepartment/ picture.gif untuk nama path. Karena **browser Web** (seperti Internet Explorer dan Firefox) menerapkan sisi klien HTTP, dalam konteks Web, kami akan menggunakan kata *browser* dan *klien* secara bergantian. **Server web**, yang mengimplementasikan sisi server HTTP, menampung objek Web, masing-masing dapat dialamatkan oleh URL. Server Web populer termasuk Apache dan Microsoft Internet Information Server.

HTTP mendefinisikan bagaimana klien Web meminta halaman Web dari server Web dan bagaimana server mentransfer halaman Web ke klien. Kami membahas interaksi antara klien dan server secara rinci nanti, tetapi gagasan umumnya diilustrasikan pada Gambar 2.6. Saat pengguna meminta halaman Web (misalnya, mengklik hyperlink), browser mengirimkan pesan permintaan HTTP untuk objek di halaman ke server. Server menerima permintaan dan merespons dengan pesan respons HTTP yang berisi objek.

HTTP menggunakan TCP sebagai protokol transport yang mendasarinya (daripada berjalan di atas UDP). Klien HTTP pertama kali memulai koneksi TCP dengan server. Setelah koneksi terjalin, browser dan server memproses akses TCP melalui antarmuka soketnya. Seperti dijelaskan dalam Bagian 2.1, di sisi klien, antarmuka soket adalah pintu antara proses klien dan koneksi TCP; di sisi server itu adalah



Gambar 2.6 Perilaku respons-permintaan HTTP

100 BAB 2 • LAPISAN APLIKASI

pintu antara proses server dan koneksi TCP. Klien mengirim pesan permintaan HTTP ke antarmuka soketnya dan menerima pesan respons HTTP dari antarmuka soketnya. Demikian pula, server HTTP menerima pesan permintaan dari antarmuka soketnya dan mengirimkan pesan respons ke antarmuka soketnya. Setelah klien mengirim pesan ke antarmuka soketnya, pesan tersebut berada di luar kendali klien dan "di tangan" TCP. Ingat dari Bagian 2.1 bahwa TCP menyediakan layanan transfer data yang andal ke HTTP. Ini menyiratkan bahwa setiap pesan permintaan HTTP yang dikirim oleh proses klien akhirnya tiba utuh di server; demikian pula, setiap pesan respons HTTP yang dikirim oleh proses server akhirnya tiba secara utuh di klien. Di sini kita melihat salah satu keuntungan besar dari arsitektur berlapis — HTTP tidak perlu khawatir tentang data yang hilang atau detail tentang bagaimana TCP pulih dari kehilangan atau penataan ulang data dalam jaringan. Itu adalah tugas TCP dan protokol di lapisan bawah tumpukan protokol.

Penting untuk dicatat bahwa server mengirimkan file yang diminta ke klien tanpa menyimpan informasi status apa pun tentang klien. Jika klien tertentu meminta objek yang sama dua kali dalam jangka waktu beberapa detik, server tidak menjawab dengan mengatakan bahwa itu hanya melayani objek ke klien; sebagai gantinya, server mengirim ulang objek, karena sama sekali lupa apa yang dilakukannya sebelumnya. Karena server HTTP tidak memelihara informasi tentang klien, HTTP dikatakan sebagai **protokol tanpa kewarganegaraan**. Kami juga berkomentar bahwa Web menggunakan arsitektur aplikasi client-server, seperti yang dijelaskan di Bagian 2.1. Server Web selalu aktif, dengan alamat IP tetap, dan melayani permintaan dari jutaan browser yang berpotensi berbeda.

2.2.2 Koneksi Non-Persisten dan Persisten

Dalam banyak aplikasi Internet, klien dan server berkomunikasi untuk waktu yang lama, dengan klien membuat serangkaian permintaan dan server merespons setiap permintaan. Bergantung pada aplikasi dan bagaimana aplikasi digunakan, serangkaian permintaan dapat dilakukan secara berurutan, secara berkala dengan interval reguler, atau sesekali. Ketika interaksi klien-server ini terjadi melalui TCP, pengembang aplikasi perlu membuat keputusan penting—haruskah setiap pasangan permintaan/respons dikirim melalui koneksi TCP yang *terpisah*, atau haruskah semua permintaan dan tanggapan terkaitnya dikirim melalui koneksi TCP yang sama? Dalam pendekatan sebelumnya, aplikasi dikatakan menggunakan **koneksi non-persistent**; dan dalam pendekatan terakhir, **koneksi yang gigih**. Untuk mendapatkan pemahaman mendalam tentang masalah desain ini, mari kita periksa kelebihan dan kekurangan koneksi persisten dalam konteks aplikasi tertentu, yaitu HTTP, yang dapat menggunakan koneksi non-persistent dan koneksi persisten. Meskipun HTTP menggunakan koneksi persisten dalam mode defaultnya, klien dan server HTTP dapat dikonfigurasi untuk menggunakan koneksi non-persistent sebagai gantinya.

HTTP dengan Koneksi Tidak Tetap

Mari kita telusuri langkah-langkah mentransfer halaman Web dari server ke klien untuk kasus koneksi non-persisten. Misalkan halaman tersebut terdiri dari HTML dasar

file dan 10 gambar JPEG, dan ke-11 objek ini berada di server yang sama.

Selanjutnya misalkan URL untuk file HTML dasar adalah

<http://www.someSchool.edu/someDepartment/home.index>

Inilah yang terjadi:

1. Proses klien HTTP memulai koneksi TCP ke server
www.someSchool.edu pada nomor port 80, yang merupakan nomor port default untuk HTTP. Terkait dengan koneksi TCP, akan ada soket di klien dan soket di server.
2. Klien HTTP mengirimkan pesan permintaan HTTP ke server melalui soketnya. Pesan permintaan menyertakan nama jalur /someDepartment/home.index.
(Kami akan membahas pesan HTTP secara mendetail di bawah.)
3. Proses server HTTP menerima pesan permintaan melalui soketnya, mengambil objek / someDepartment/home.index dari penyimpanannya (RAM atau disk), merangkum objek dalam pesan respons HTTP, dan mengirimkan pesan respons ke klien melalui soketnya.
4. Proses server HTTP memberitahu TCP untuk menutup koneksi TCP. (Tapi TCP tidak benar-benar mengakhiri koneksi sampai mengetahui dengan pasti bahwa klien telah menerima pesan respons secara utuh.)
5. Klien HTTP menerima pesan respons. Istilah koneksi TCP nates. Pesan tersebut menunjukkan bahwa objek yang dienkapsulasi adalah file HTML. Klien mengekstrak file dari pesan respons, memeriksa file HTML, dan menemukan referensi ke 10 objek JPEG.
6. Empat langkah pertama kemudian diulangi untuk masing-masing objek JPEG yang direferensikan.

Saat browser menerima halaman Web, ia menampilkan halaman tersebut kepada pengguna. Dua browser yang berbeda dapat menginterpretasikan (yaitu, menampilkan kepada pengguna) halaman Web dengan cara yang agak berbeda. HTTP tidak ada hubungannya dengan bagaimana halaman Web ditafsirkan oleh klien. Spesifikasi HTTP ([RFC 1945] dan [RFC 2616]) hanya menentukan protokol komunikasi antara program HTTP klien dan program HTTP server.

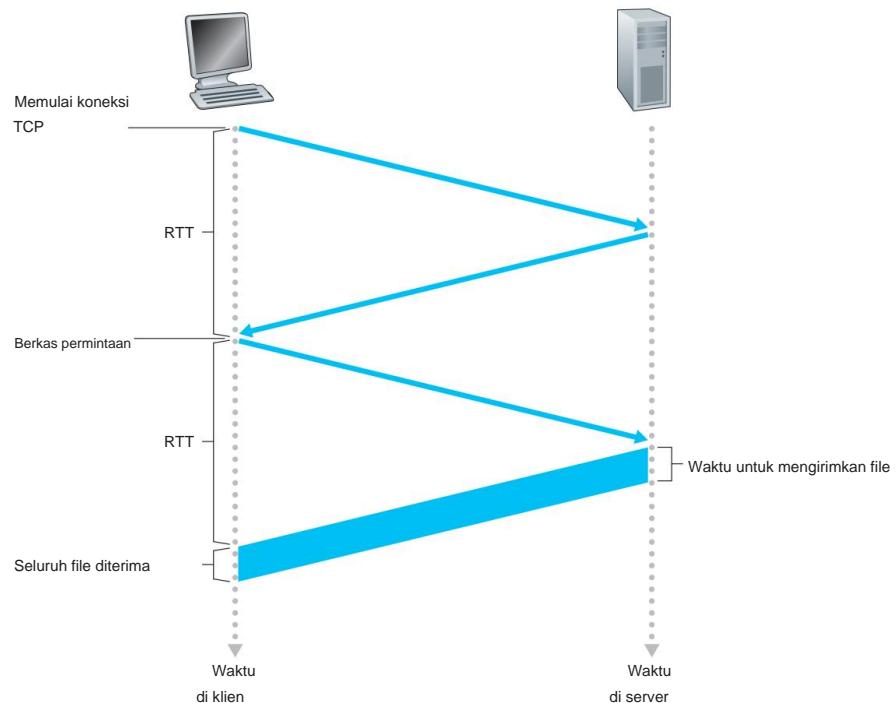
Langkah-langkah di atas mengilustrasikan penggunaan koneksi non-persistent, di mana setiap koneksi TCP ditutup setelah server mengirim objek—koneksi tidak bertahan untuk objek lain. Perhatikan bahwa setiap koneksi TCP mengangkut tepat satu pesan permintaan dan satu pesan tanggapan. Jadi, dalam contoh ini, ketika pengguna meminta halaman Web, 11 koneksi TCP dibuat.

Dalam langkah-langkah yang dijelaskan di atas, kami sengaja mengaburkan apakah klien mendapatkan 10 JPEG melalui 10 koneksi serial TCP, atau apakah beberapa JPEG diperoleh melalui koneksi TCP paralel. Memang, pengguna dapat mengkonfigurasi browser modern untuk mengontrol tingkat paralelisme. Dalam mode defaultnya, sebagian besar browser membuka 5 hingga 10 koneksi TCP paralel, dan masing-masing koneksi ini menangani satu transaksi respons-permintaan. Jika pengguna lebih suka, jumlah maksimum

102 BAB 2 • LAPISAN APLIKASI

koneksi paralel dapat diatur ke satu, dalam hal ini 10 koneksi dibuat secara serial. Seperti yang akan kita lihat di bab selanjutnya, penggunaan koneksi paralel mempersingkat waktu respons.

Sebelum melanjutkan, mari lakukan kalkulasi di belakang amplop untuk memperkirakan jumlah waktu yang berlalu sejak klien meminta file HTML dasar hingga seluruh file diterima oleh klien. Untuk tujuan ini, kita mendefinisikan **round-trip time (RTT)**, yaitu waktu yang dibutuhkan paket kecil untuk melakukan perjalanan dari klien ke server dan kemudian kembali ke klien. RTT mencakup penundaan propagasi paket, penundaan antrian paket di router dan sakelar perantara, dan penundaan pemrosesan paket. (Penundaan ini dibahas di Bagian 1.4.) Sekarang perhatikan apa yang terjadi saat pengguna mengklik hyperlink. Seperti yang ditunjukkan pada Gambar 2.7, ini menyebabkan browser memulai koneksi TCP antara browser dan server Web; ini melibatkan "jabat tangan tiga arah"—klien mengirim segmen TCP kecil ke server, server mengakui dan merespons dengan segmen TCP kecil, dan, akhirnya, klien mengakui kembali ke server. Dua bagian pertama dari jabat tangan tiga arah mengambil satu RTT. Setelah menyelesaikan dua bagian pertama dari jabat tangan, klien mengirimkan pesan permintaan HTTP yang digabungka



Gambar 2.7 Perhitungan di balik amplop untuk waktu yang dibutuhkan untuk meminta dan menerima file HTML

jabat tangan tiga arah (pengakuan) ke dalam koneksi TCP. Setelah pesan permintaan tiba di server, server mengirimkan file HTML ke koneksi TCP. Permintaan/respond HTTP ini memakan RTT lain. Jadi, kira-kira, total waktu respons adalah dua RTT ditambah waktu transmisi di server file HTML.

HTTP dengan Persistent Connections

Koneksi non-persisten memiliki beberapa kekurangan. Pertama, koneksi baru harus dibuat dan dipelihara untuk *setiap objek yang diminta*. Untuk setiap koneksi ini, buffer TCP harus dialokasikan dan variabel TCP harus disimpan di klien dan server. Ini dapat menempatkan beban yang signifikan pada server Web, yang mungkin melayani permintaan dari ratusan klien yang berbeda secara bersamaan.

Kedua, seperti yang baru saja kami jelaskan, setiap objek mengalami penundaan pengiriman dua RTT — satu RTT untuk membuat koneksi TCP dan satu RTT untuk meminta dan menerima objek.

Dengan koneksi persisten, server membiarkan koneksi TCP terbuka setelah mengirimkan respons. Permintaan dan respons selanjutnya antara klien dan server yang sama dapat dikirim melalui koneksi yang sama. Secara khusus, seluruh halaman Web (dalam contoh di atas, file HTML dasar dan 10 gambar) dapat dikirim melalui satu koneksi TCP yang persisten. Selain itu, beberapa halaman Web yang berada di server yang sama dapat dikirim dari server ke klien yang sama melalui satu koneksi TCP yang persisten. Permintaan objek ini dapat dilakukan secara berurutan, tanpa menunggu balasan untuk permintaan yang tertunda (pipelining). Biasanya, server HTTP menutup koneksi saat tidak digunakan untuk waktu tertentu (interval waktu habis yang dapat dikonfigurasi). Ketika server menerima permintaan back-to-back, itu mengirimkan objek back-to-back. Mode default HTTP menggunakan koneksi persisten dengan perpipaan. Kita akan membandingkan secara kuantitatif kinerja koneksi non-persisten dan persisten dalam soal-soal pekerjaan rumah Bab 2 dan 3. Anda juga didorong untuk melihat [Heide mann 1997; Nielsen 1997].

2.2.3 Format Pesan HTTP

Spesifikasi HTTP [RFC 1945; RFC 2616] menyertakan definisi format pesan HTTP. Ada dua jenis pesan HTTP, pesan permintaan dan pesan tanggapan, keduanya dibahas di bawah ini.

Pesan Permintaan HTTP

Di bawah ini kami menyediakan pesan permintaan HTTP biasa:

DAPATKAN /somedir/page.html HTTP/1.1
Host: www.someschool.edu

104 BAB 2 • LAPISAN APLIKASI

Koneksi: tutup

Agen pengguna: Mozilla/5.0 Bahasa
terima: fr

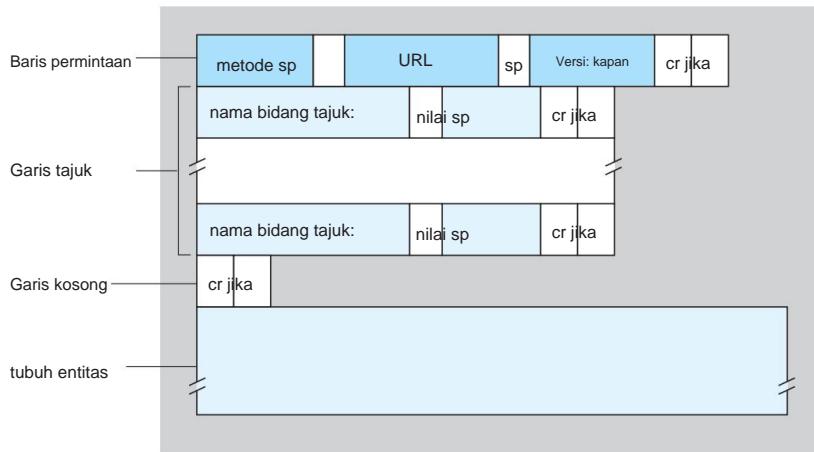
Kita bisa belajar banyak dengan mencermati pesan permintaan sederhana ini. Pertama-tama, kita melihat bahwa pesan tersebut ditulis dalam teks ASCII biasa, sehingga manusia biasa yang melek komputer dapat membacanya. Kedua, kita melihat bahwa pesan terdiri dari lima baris, masing-masing diikuti dengan carriage return dan line feed. Baris terakhir diikuti dengan tambahan carriage return dan line feed. Meskipun pesan permintaan khusus ini memiliki lima baris, pesan permintaan dapat memiliki lebih banyak baris atau sesedikit satu baris. Baris pertama dari pesan permintaan HTTP disebut **baris permintaan**; baris berikutnya disebut **baris header**. Baris permintaan memiliki tiga bidang: bidang metode, bidang URL, dan bidang versi HTTP. Bidang metode dapat mengambil beberapa nilai yang berbeda, termasuk GET, POST, HEAD, PUT, dan DELETE.

Sebagian besar pesan permintaan HTTP menggunakan metode GET. Metode GET digunakan saat browser meminta objek, dengan objek yang diminta diidentifikasi di kolom URL. Dalam contoh ini, browser meminta objek /somedir/page.html. Versinya cukup jelas; dalam contoh ini, browser mengimplementasikan versi HTTP/1.1.

Sekarang mari kita lihat baris header pada contoh. Baris header Host: www.someschool.edu menentukan host tempat objek berada. Anda mungkin berpikir bahwa baris tajuk ini tidak diperlukan, karena sudah ada koneksi TCP ke host. Namun, seperti yang akan kita lihat di Bagian 2.2.5, informasi yang diberikan oleh baris header host diperlukan oleh cache proxy Web. Dengan memasukkan Connection: close header line, browser memberi tahu server bahwa ia tidak ingin repot dengan koneksi yang terus-menerus; ia ingin server menutup koneksi setelah mengirim objek yang diminta. Baris header User-agent: menentukan agen pengguna, yaitu jenis browser yang membuat permintaan ke server. Di sini agen penggunanya adalah Mozilla/5.0, sebuah browser Firefox. Baris tajuk ini berguna karena server sebenarnya dapat mengirim versi berbeda dari objek yang sama ke berbagai jenis agen pengguna.

(Masing-masing versi dialamatkan oleh URL yang sama.) Terakhir, bahasa Terima: tajuk menunjukkan bahwa pengguna lebih suka menerima objek versi Prancis, jika objek seperti itu ada di server; jika tidak, server harus mengirimkan versi standarnya. Bahasa Terima: tajuk hanyalah salah satu dari banyak tajuk negosiasi konten yang tersedia di HTTP.

Setelah melihat contoh, sekarang mari kita lihat format umum dari pesan permintaan, seperti yang ditunjukkan pada Gambar 2.8. Kami melihat bahwa format umum mengikuti contoh kami sebelumnya. Anda mungkin telah memperhatikan, bagaimanapun, bahwa setelah baris header (dan tambahan carriage return dan line feed) ada "badan entitas". Badan entitas kosong dengan metode GET, tetapi digunakan dengan metode POST. Klien HTTP sering menggunakan metode POST saat pengguna mengisi formulir—misalnya, saat pengguna memberikan kata pencarian ke mesin pencari. Dengan pesan POST, pengguna masih meminta halaman Web dari server, tetapi konten tertentu dari halaman Web tersebut



Gambar 2.8 Format umum pesan permintaan HTTP

tergantung pada apa yang dimasukkan pengguna ke dalam bidang formulir. Jika nilai bidang metode adalah POST, maka badan entitas berisi apa yang dimasukkan pengguna ke dalam bidang formulir.

Kami akan lalai jika kami tidak menyebutkan bahwa permintaan yang dibuat dengan formulir tidak harus menggunakan metode POST. Sebagai gantinya, formulir HTML sering menggunakan metode GET dan menyertakan data yang dimasukkan (di bidang formulir) di URL yang diminta. Misalnya, jika formulir menggunakan metode GET, memiliki dua bidang, dan input ke dua bidang tersebut adalah monyet dan pisang, maka URL akan memiliki struktur www.somesite.com/animalsearch?monkeys&bananas. Dalam penjelajahan Web sehari-hari, Anda mungkin telah memperhatikan URL yang diperluas semacam ini.

Metode HEAD mirip dengan metode GET. Ketika server menerima permintaan dengan metode HEAD, ia merespons dengan pesan HTTP tetapi mengabaikan objek yang diminta. Pengembang aplikasi sering menggunakan metode HEAD untuk debugging. Metode PUT sering digunakan bersama dengan alat penerbitan Web. Ini memungkinkan pengguna untuk mengunggah objek ke jalur (direktori) tertentu di server Web tertentu. Metode PUT juga digunakan oleh aplikasi yang perlu mengunggah objek ke server Web. Metode DELETE memungkinkan pengguna, atau aplikasi, untuk menghapus objek di server Web.

Pesan Respons HTTP

Di bawah ini kami menyediakan pesan respons HTTP biasa. Pesan respons ini bisa menjadi respons terhadap pesan permintaan contoh yang baru saja dibahas.

HTTP/1.1 200 oke

Koneksi: tutup

106 BAB 2 • LAPISAN APLIKASI

Tanggal: Sel, 09 Agustus 2011 15:44:04 GMT Server:
Apache/2.2.3 (CentOS)
Terakhir Diubah: Sel, 09 Agustus 2011 15:11:03 GMT Panjang
Konten: 6821 Jenis Konten: teks/html

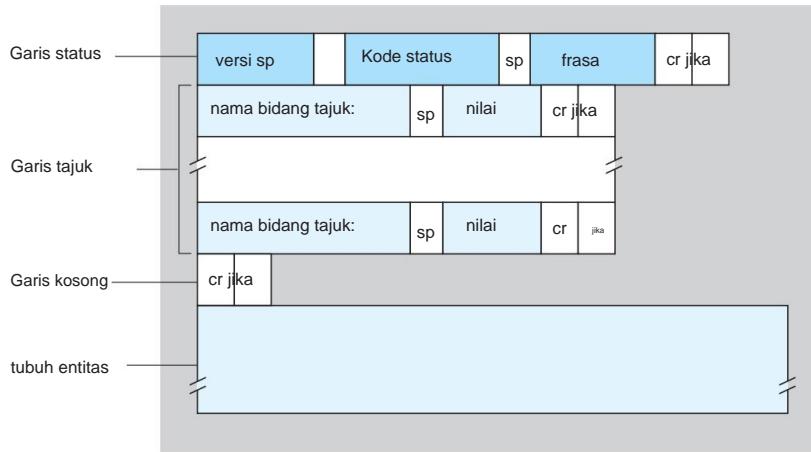
(data data data data data...)

Mari kita perhatikan baik-baik pesan tanggapan ini. Ini memiliki tiga bagian: baris status awal , enam **baris header**, dan kemudian **badan entitas**. Badan entitas adalah inti dari pesan—berisi objek yang diminta itu sendiri (diwakili oleh data data data data data ...). Baris status memiliki tiga bidang: bidang versi protokol, kode status, dan pesan status yang sesuai. Dalam contoh ini, baris status menunjukkan bahwa server menggunakan HTTP/1.1 dan semuanya baik-baik saja (yaitu, server telah menemukan, dan mengirim, objek yang diminta).

Sekarang mari kita lihat baris header. Server menggunakan Connection: close header line untuk memberi tahu klien bahwa ia akan menutup koneksi TCP setelah mengirim pesan. Baris tajuk Tanggal: menunjukkan waktu dan tanggal saat respons HTTP dibuat dan dikirim oleh server. Perhatikan bahwa ini bukan saat objek dibuat atau terakhir diubah; itu adalah waktu ketika server mengambil objek dari sistem filenya, memasukkan objek ke dalam pesan respons, dan mengirimkan pesan respons. Baris header Server: menunjukkan bahwa pesan dibuat oleh server Web Apache; itu analog dengan User-agent: baris header dalam pesan permintaan HTTP. Baris header Last-Modified: menunjukkan waktu dan tanggal saat objek dibuat atau terakhir diubah. Header Last-Modified:, yang akan segera kita bahas lebih detail, sangat penting untuk caching objek, baik di klien lokal maupun di server cache jaringan (juga dikenal sebagai server proxy). Baris header Content-Length: menunjukkan jumlah byte dalam objek yang dikirim. Baris header Content-Type: menunjukkan bahwa objek dalam badan entitas adalah teks HTML. (Tipe objek secara resmi ditunjukkan oleh Content-Type: header dan bukan oleh ekstensi file.)

Setelah melihat sebuah contoh, sekarang mari kita periksa format umum dari sebuah pesan tanggapan, yang ditunjukkan pada Gambar 2.9. Format umum pesan tanggapan ini cocok dengan contoh pesan tanggapan sebelumnya. Katakanlah beberapa kata tambahan tentang kode status dan frase mereka. Kode status dan frase terkait menunjukkan hasil permintaan. Beberapa kode status umum dan frasa terkait meliputi:

- 200 OK: Permintaan berhasil dan informasi dikembalikan dalam tanggapan. •
301 Dipindahkan Secara Permanen: Objek yang diminta telah dipindahkan secara permanen; URL baru ditentukan di Lokasi: tajuk pesan respons. Perangkat lunak klien akan secara otomatis mengambil URL baru.



Gambar 2.9 Format umum pesan respons HTTP

- 400 Permintaan Buruk: Ini adalah kode kesalahan umum yang menunjukkan bahwa permintaan tidak dapat dipahami oleh server.
- 404 Not Found: Dokumen yang diminta tidak ada di server ini. • 505 Versi HTTP Tidak Didukung: Versi protokol HTTP yang diminta tidak didukung oleh server.

Bagaimana Anda ingin melihat pesan respons HTTP yang sebenarnya? Ini sangat direkomendasikan dan sangat mudah dilakukan! Pertama Telnet ke server Web favorit Anda. Kemudian ketik pesan permintaan satu baris untuk beberapa objek yang disimpan di server. Misalnya, jika Anda memiliki akses ke command prompt, ketik:

telnet cis.poly.edu 80

DAPATKAN /~ross/ HTTP/1.1

Host: cis.poly.edu

(Tekan carriage return dua kali setelah mengetik baris terakhir.) Ini membuka koneksi TCP ke port 80 dari host cis.poly.edu dan kemudian mengirimkan pesan permintaan HTTP. Anda akan melihat pesan tanggapan yang menyertakan file HTML dasar dari beranda Profesor Ross. Jika Anda lebih suka melihat baris pesan HTTP dan tidak menerima objek itu sendiri, ganti GET dengan HEAD. Terakhir, ganti /~ross/ dengan /~banana/ dan lihat pesan respons seperti apa yang Anda dapatkan.

Pada bagian ini kita membahas sejumlah baris header yang dapat digunakan dalam pesan HTTP request dan response. Spesifikasi HTTP mendefinisikan banyak, banyak



108 BAB 2 • LAPISAN APLIKASI

lebih banyak baris header yang dapat disisipkan oleh browser, server Web, dan server cache jaringan. Kami hanya membahas sebagian kecil dari totalitas baris tajuk. Kami akan membahas beberapa lagi di bawah ini dan sejumlah kecil lainnya ketika kami membahas caching Web jaringan di Bagian 2.2.5. Diskusi yang sangat mudah dibaca dan komprehensif tentang protokol HTTP, termasuk header dan kode statusnya, diberikan dalam [Krishnamurthy 2001].

Bagaimana cara browser memutuskan baris tajuk mana yang akan disertakan dalam pesan permintaan? Bagaimana server Web memutuskan baris tajuk mana yang akan disertakan dalam pesan tanggapan? Brower akan menghasilkan baris header sebagai fungsi dari jenis dan versi browser (misalnya, browser HTTP/1.0 tidak akan menghasilkan baris header 1.1), konfigurasi pengguna browser (misalnya, bahasa pilihan), dan apakah browser saat ini memiliki versi objek yang di-cache, tetapi mungkin kedaluwarsa. Server web berperilaku serupa: Ada berbagai produk, versi, dan konfigurasi, yang semuanya memengaruhi baris header mana yang disertakan dalam pesan respons.

2.2.4 Interaksi Pengguna-Server: Cookie

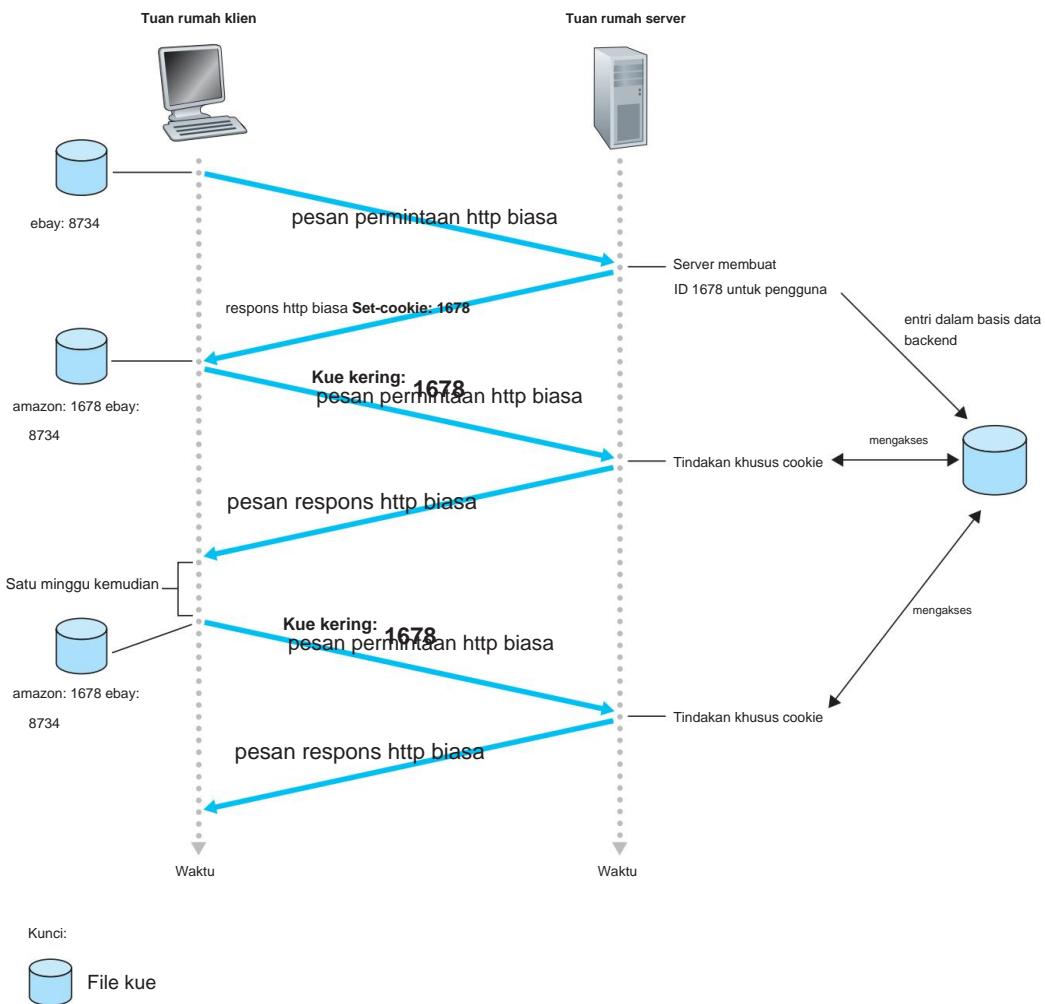
Kami sebutkan di atas bahwa server HTTP tidak memiliki kewarganegaraan. Ini menyederhanakan desain server dan memungkinkan para insinyur untuk mengembangkan server Web berkinerja tinggi yang dapat menangani ribuan koneksi TCP secara bersamaan. Namun, sering diinginkan situs Web untuk mengidentifikasi pengguna, baik karena server ingin membatasi akses pengguna atau karena ingin menyajikan konten sebagai fungsi dari identitas pengguna. Untuk tujuan ini, HTTP menggunakan cookie. Cookie, didefinisikan dalam [RFC 6265], memungkinkan situs melacak pengguna. Sebagian besar situs Web komersial utama menggunakan cookie saat ini.

Seperti yang ditunjukkan pada Gambar 2.10, teknologi cookie memiliki empat komponen: (1) baris header cookie dalam pesan respons HTTP; (2) baris tajuk cookie di pesan permintaan HTTP; (3) file cookie disimpan di sistem akhir pengguna dan dikelola oleh browser pengguna; dan (4) database back-end di situs Web. Dengan menggunakan Gambar 2.10, mari kita telusuri contoh cara kerja cookie. Misalkan Susan, yang selalu mengakses Web menggunakan Internet Explorer dari PC rumahnya, menghubungi Amazon.com untuk pertama kalinya. Misalkan di masa lalu dia telah mengunjungi situs eBay.

Saat permintaan masuk ke server Web Amazon, server membuat nomor identifikasi unik dan membuat entri di basis data back-end yang diindeks oleh nomor identifikasi. Server Web Amazon kemudian merespons ke browser Susan, termasuk dalam respons HTTP, header Set-cookie:, yang berisi nomor identifikasi. Misalnya, baris tajuk mungkin:

Set-cookie: 1678

Saat browser Susan menerima pesan respons HTTP, ia melihat header Set cookie:. Browser kemudian menambahkan baris ke file cookie khusus yang dikelolanya. Baris ini menyertakan nama host server dan nomor identifikasi di header Set-cookie:. Perhatikan bahwa file cookie sudah memiliki entri untuk



Gambar 2.10 Menjaga status pengguna dengan cookie

eBay, karena Susan pernah mengunjungi situs itu di masa lalu. Saat Susan terus menelusuri situs Amazon, setiap kali dia meminta halaman Web, browsernya memeriksa file cookie-nya, mengekstrak nomor identifikasinya untuk situs ini, dan meletakkan baris header cookie yang menyertakan nomor identifikasi dalam permintaan HTTP. Secara khusus, setiap permintaan HTTP-nya ke server Amazon menyertakan baris tajuk:

Kuki: 1678

110 BAB 2 • LAPISAN APLIKASI

Dengan cara ini, server Amazon dapat melacak aktivitas Susan di situs Amazon. Meskipun situs Web Amazon tidak perlu mengetahui nama Susan, ia tahu persis halaman mana yang dikunjungi pengguna 1678, dalam urutan apa, dan pada jam berapa!

Amazon menggunakan cookie untuk menyediakan layanan keranjang belanjanya—Amazon dapat menyimpan daftar semua pembelian yang diinginkan Susan, sehingga dia dapat membayarnya secara kolektif di akhir sesi.

Jika Susan kembali ke situs Amazon, katakanlah, satu minggu kemudian, browsernya akan terus memasukkan baris tajuk Cookie: 1678 dalam pesan permintaan. Amazon juga merekomendasikan produk kepada Susan berdasarkan halaman Web yang dia kunjungi di Amazon sebelumnya. Jika Susan juga mendaftarkan dirinya ke Amazon—memberikan nama lengkap, alamat email, alamat pos, dan informasi kartu kredit—Amazon kemudian dapat memasukkan informasi ini ke dalam basis datanya, dengan demikian menghubungkan nama Susan dengan nomor identifikasinya (dan semua halaman yang dia daftarkan). telah mengunjungi situs tersebut di masa lalu!). Beginilah cara Amazon dan situs e-niaga lainnya menyediakan "belanja sekali klik"—ketika Susan memilih untuk membeli item selama kunjungan berikutnya, dia tidak perlu memasukkan kembali nama, nomor kartu kredit, atau alamatnya.

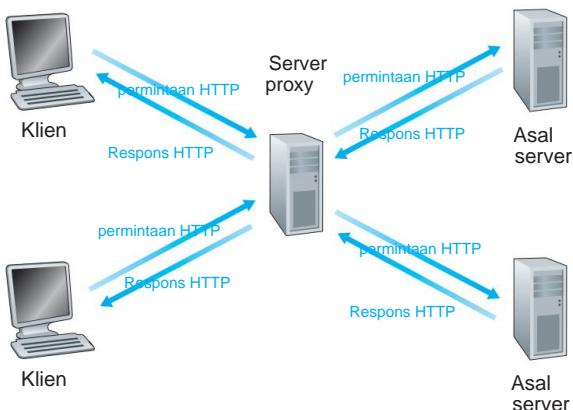
Dari diskusi ini kita melihat bahwa cookie dapat digunakan untuk mengidentifikasi pengguna. Pertama kali pengguna mengunjungi situs, pengguna dapat memberikan identifikasi pengguna (mungkin namanya). Selama sesi berikutnya, browser meneruskan header cookie ke server, sehingga mengidentifikasi pengguna ke server. Dengan demikian, cookie dapat digunakan untuk membuat lapisan sesi pengguna di atas HTTP tanpa kewarganegaraan. Misalnya, ketika pengguna masuk ke aplikasi email berbasis Web (seperti Hotmail), browser mengirimkan informasi cookie ke server, mengizinkan server untuk mengidentifikasi pengguna selama sesi pengguna dengan aplikasi tersebut.

Meskipun cookie sering menyederhanakan pengalaman belanja Internet bagi pengguna, cookie kontroversial karena juga dapat dianggap sebagai pelanggaran privasi. Seperti yang baru saja kita lihat, dengan menggunakan kombinasi cookie dan informasi akun yang disediakan pengguna, situs Web dapat mempelajari banyak hal tentang pengguna dan berpotensi menjual informasi ini ke pihak ketiga. Cookie Central [Cookie Central 2012] mencakup informasi ekstensif tentang kontroversi cookie.

2.2.5 Caching Web

Cache **Web** —juga disebut **server proxy**—adalah entitas jaringan yang memenuhi permintaan HTTP atas nama server Web asal. Cache Web memiliki penyimpanan disknya sendiri dan menyimpan salinan objek yang baru diminta di penyimpanan ini. Seperti yang ditunjukkan pada Gambar 2.11, browser pengguna dapat dikonfigurasi sehingga semua permintaan HTTP pengguna pertama-tama diarahkan ke cache Web. Setelah browser dikonfigurasi, setiap permintaan browser untuk objek pertama-tama diarahkan ke cache Web. Sebagai contoh, misalkan browser meminta objek <http://www.someschool.edu/campus.gif>. Inilah yang terjadi:

1. Browser membuat koneksi TCP ke cache Web dan mengirimkan permintaan HTTP untuk objek ke cache Web.



Gambar 2.11 Klien meminta objek melalui cache Web

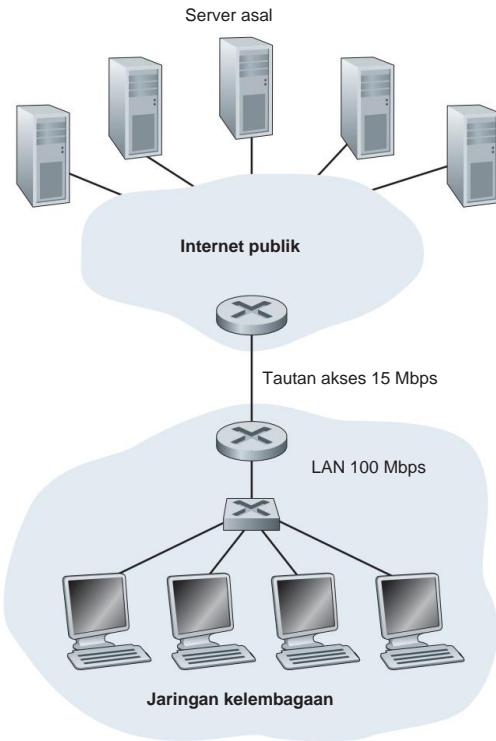
2. Cache Web memeriksa apakah ada salinan objek yang disimpan secara lokal. Jika ya, cache Web mengembalikan objek dalam pesan respons HTTP ke browser klien.
3. Jika cache Web tidak memiliki objek, cache Web membuka koneksi TCP ke server asal, yaitu ke www.someschool.edu. Cache Web kemudian mengirimkan permintaan HTTP untuk objek ke koneksi TCP cache-ke-server. Setelah menerima permintaan ini, server asal mengirimkan objek dalam respons HTTP ke cache Web.
4. Saat cache Web menerima objek, ia menyimpan salinannya di penyimpanan lokalnya dan mengirimkan salinannya, di dalam pesan respons HTTP, ke browser klien (melalui koneksi TCP yang ada antara browser klien dan cache Web).

Perhatikan bahwa cache adalah server dan klien sekaligus. Ketika menerima permintaan dari dan mengirimkan tanggapan ke browser, itu adalah server. Ketika mengirimkan permintaan ke dan menerima tanggapan dari server asal, itu adalah klien.

Biasanya cache Web dibeli dan dipasang oleh ISP. Misalnya, sebuah universitas mungkin memasang cache di jaringan kampusnya dan mengonfigurasi semua browser kampus untuk mengarah ke cache. Atau ISP perumahan utama (seperti AOL) mungkin memasang satu atau lebih cache di jaringannya dan melakukan prakonfigurasi pada browser yang dikirim untuk menunjuk ke cache yang diinstal.

Caching web telah melihat penerapan di Internet karena dua alasan. Pertama, cache Web dapat secara substansial mengurangi waktu respons untuk permintaan klien, terutama jika bandwidth bottleneck antara klien dan server asal jauh lebih sedikit daripada bandwidth bottleneck antara klien dan cache. Jika ada koneksi berkecepatan tinggi antara klien dan cache, seperti yang sering terjadi, dan jika cache memiliki objek yang diminta, maka cache akan dapat mengirimkan objek dengan cepat ke klien. Kedua, seperti yang akan segera kami ilustrasikan dengan sebuah contoh, cache Web dapat mengurangi lalu lintas secara substansial

112 BAB 2 • LAPISAN APLIKASI



Gambar 2.12 Hambatan antara jaringan kelembagaan dan Internet

tautan akses institusi ke Internet. Dengan mengurangi lalu lintas, institusi (misalnya, perusahaan atau universitas) tidak perlu mengupgrade bandwidth dengan cepat, sehingga mengurangi biaya. Selain itu, cache Web secara substansial dapat mengurangi lalu lintas Web di Internet secara keseluruhan, sehingga meningkatkan performa untuk semua aplikasi.

Untuk mendapatkan pemahaman yang lebih mendalam tentang manfaat cache, mari kita lihat contoh dalam konteks Gambar 2.12. Gambar ini menunjukkan dua jaringan—jaringan institusional dan sisanya dari Internet publik. Jaringan institusional adalah LAN berkecepatan tinggi. Router di jaringan institusional dan router di Internet dihubungkan dengan tautan 15 Mbps. Server asal terpasang ke Internet tetapi berlokasi di seluruh dunia. Misalkan ukuran objek rata-rata adalah 1 Mbits dan tingkat permintaan rata-rata dari browser institusi ke server asal adalah 15 permintaan per detik. Misalkan pesan permintaan HTTP sangat kecil dan dengan demikian tidak menciptakan lalu lintas di jaringan atau di tautan akses (dari router institusional ke router Internet). Anggap juga bahwa jumlah waktu yang diperlukan dari saat router di sisi Internet dari link akses pada Gambar 2.12 meneruskan permintaan HTTP (dalam sebuah datagram IP) hingga menerima respons (biasanya dalam banyak datagram IP) adalah dua detik rata-rata. Secara informal, kami menyebut penundaan terakhir ini sebagai "penundaan Internet".

Total waktu respons—yaitu, waktu dari permintaan browser terhadap suatu objek hingga penerimaan objek—adalah jumlah penundaan LAN, penundaan akses (yaitu, penundaan antara dua router), dan Internet menunda. Sekarang mari kita lakukan perhitungan yang sangat kasar untuk memperkirakan keterlambatan ini. Intensitas lalu lintas pada LAN (lihat Bagian 1.4.2) adalah

$$(15 \text{ permintaan/dtk}) (1 \text{ Mbit/permintaan}) / (100 \text{ Mbps}) = 0,15$$

sedangkan intensitas lalu lintas pada link akses (dari router Internet ke router institusi) adalah

$$(15 \text{ permintaan/dtk}) (1 \text{ Mbit/permintaan}) / (15 \text{ Mbps}) = 1$$

Intensitas lalu lintas 0,15 pada LAN biasanya menghasilkan, paling banyak, penundaan puluhan milidetik; karenanya, kita dapat mengabaikan penundaan LAN. Namun, seperti yang dibahas dalam Bagian 1.4.2, ketika intensitas lalu lintas mendekati 1 (seperti kasus link akses pada Gambar 2.12), delay pada link menjadi sangat besar dan tumbuh tanpa batas.

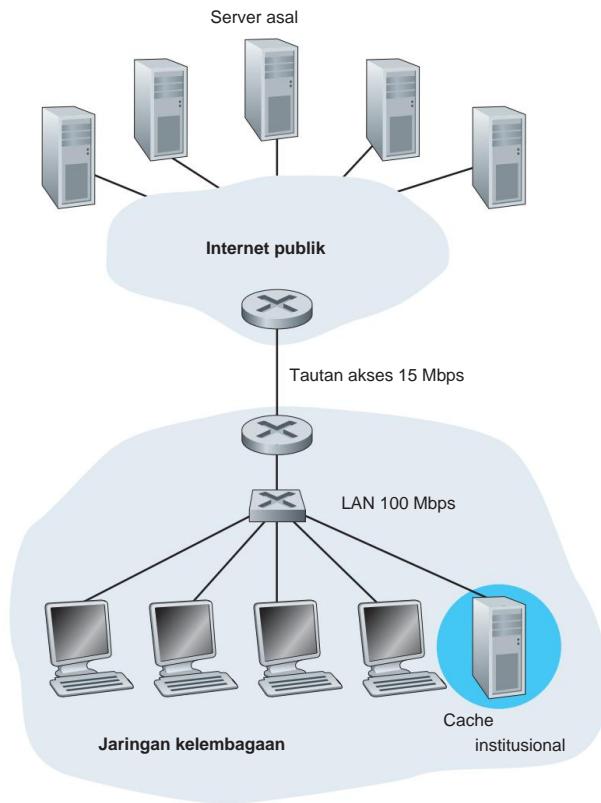
Dengan demikian, waktu respons rata-rata untuk memenuhi permintaan akan berada dalam hitungan menit, jika tidak lebih, yang tidak dapat diterima oleh pengguna institusi. Jelas ada sesuatu yang harus dilakukan.

Salah satu solusi yang mungkin adalah meningkatkan kecepatan akses dari 15 Mbps menjadi, katakanlah, 100 Mbps. Ini akan menurunkan intensitas lalu lintas pada link akses menjadi 0,15, yang diterjemahkan menjadi penundaan yang dapat diabaikan antara dua router. Dalam hal ini, total waktu respons kira-kira akan menjadi dua detik, yaitu penundaan Internet. Tetapi solusi ini juga berarti bahwa institusi tersebut harus memutakhirkkan tautan aksesnya dari 15 Mbps menjadi 100 Mbps, suatu proposisi yang mahal.

Sekarang pertimbangkan solusi alternatif untuk tidak memutakhirkkan tautan akses tetapi memasang cache Web di jaringan institusional. Solusi ini dilustrasikan pada Gambar 2.13. Hit rate—fraksi permintaan yang dipenuhi oleh cache—biasanya berkisar antara 0,2 hingga 0,7 dalam praktiknya. Untuk tujuan ilustrasi, misalkan cache memberikan hit rate 0,4 untuk institusi ini. Karena klien dan cache terhubung ke LAN berkecepatan tinggi yang sama, 40 persen permintaan akan segera dipenuhi, katakanlah, dalam 10 milidetik, oleh cache. Namun demikian, 60 persen permintaan sisanya masih harus dipenuhi oleh server asal. Tetapi dengan hanya 60 persen objek yang diminta melewati link akses, intensitas lalu lintas pada link akses berkurang dari 1,0 menjadi 0,6. Biasanya, intensitas lalu lintas kurang dari 0,8 sesuai dengan penundaan kecil, katakanlah, puluhan milidetik, pada tautan 15 Mbps. Penundaan ini dapat diabaikan dibandingkan dengan penundaan Internet dua detik. Mengingat pertimbangan ini, keterlambatan rata-rata adalah

$$0,4 (0,01 \text{ detik}) + 0,6 (2,01 \text{ detik})$$

yang hanya sedikit lebih besar dari 1,2 detik. Dengan demikian, solusi kedua ini memberikan waktu respons yang bahkan lebih rendah daripada solusi pertama, dan tidak mengharuskan institusi untuk meningkatkan tautannya ke Internet. Institusi tentu saja harus membeli



Gambar 2.13 Menambahkan cache ke jaringan institusional

dan instal cache Web. Namun biaya ini rendah—banyak cache menggunakan perangkat lunak domain publik yang dijalankan pada PC murah.

Melalui penggunaan **Jaringan Distribusi Konten (CDN)**, cache Web semakin memainkan peran penting di Internet. Perusahaan CDN menginstal banyak cache yang didistribusikan secara geografis di seluruh Internet, sehingga melokalkan sebagian besar lalu lintas. Ada CDN bersama (seperti Akamai dan Limelight) dan CDN khusus (seperti Google dan Microsoft). Kami akan membahas CDN lebih detail di Bab 7.

2.2.6 GET Bersyarat

Meskipun caching dapat mengurangi waktu respons yang dirasakan pengguna, ini menimbulkan masalah baru—salinan objek yang berada di cache mungkin basi. Dengan kata lain, objek yang ditempatkan di server Web mungkin telah dimodifikasi sejak salinan di-cache di klien. Untungnya, HTTP memiliki mekanisme yang memungkinkan cache memverifikasi bahwa objeknya mutakhir. Mekanisme ini disebut **GET bersyarat**. HTTP

pesan permintaan adalah apa yang disebut pesan GET bersyarat jika (1) pesan permintaan menggunakan metode GET dan (2) pesan permintaan menyertakan baris header If-Modified-Since::

Untuk mengilustrasikan bagaimana GET bersyarat beroperasi, mari kita telusuri sebuah contoh. Pertama, atas nama browser yang meminta, cache proxy mengirimkan pesan permintaan ke server Web:

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

Kedua, server Web mengirimkan pesan respons dengan objek yang diminta ke cache:

HTTP/1.1 200 OK
Tanggal: Sab, 8 Okt 2011 15:39:29 Server:
Apache/1.3.0 (Unix)
Terakhir Dimodifikasi: Rab, 7 Sep 2011 09:23:24 Content-Type:
image/gif
(data data data data data...)

Cache meneruskan objek ke browser yang meminta tetapi juga menyimpan objek secara lokal. Yang penting, cache juga menyimpan tanggal modifikasi terakhir beserta objeknya. Ketiga, satu minggu kemudian, browser lain meminta objek yang sama melalui cache, dan objek tersebut masih ada di dalam cache. Karena objek ini mungkin telah dimodifikasi di server Web dalam seminggu terakhir, cache melakukan pemeriksaan terkini dengan mengeluarkan GET bersyarat. Secara khusus, cache mengirimkan:

DAPATKAN /fruit/kiwi.gif HTTP/1.1 Host:
www.exotiquecuisine.com Jika-dimodifikasi-sejak: Rab, 7 Sep 2011 09:23:24

Perhatikan bahwa nilai baris header If-modified-since: persis sama dengan nilai baris header Last-Modified: yang dikirim oleh server satu minggu yang lalu. GET bersyarat ini memberi tahu server untuk mengirim objek hanya jika objek tersebut telah dimodifikasi sejak tanggal yang ditentukan. Misalkan objek belum dimodifikasi sejak 7 Sep 2011 09:23:24. Kemudian, keempat, server Web mengirimkan pesan tanggapan ke cache:

HTTP/1.1 304 Tidak Dimodifikasi
Tanggal: Sab, 15 Okt 2011 15:39:29 Server:
Apache/1.3.0 (Unix)
(badan entitas kosong)

116 BAB 2 • LAPISAN APLIKASI

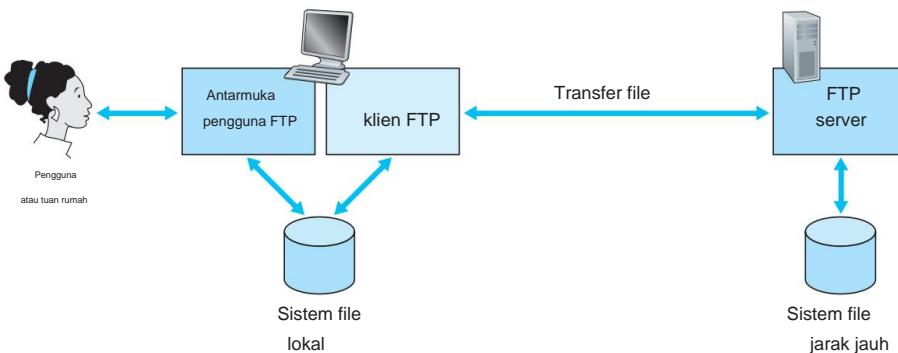
Kami melihat bahwa sebagai tanggapan terhadap GET bersyarat, server Web masih mengirimkan pesan tanggapan tetapi tidak menyertakan objek yang diminta dalam pesan tanggapan. Menyertakan objek yang diminta hanya akan membuang bandwidth dan meningkatkan waktu respons yang dirasakan pengguna, terutama jika objeknya besar. Perhatikan bahwa pesan respons terakhir ini memiliki 304 Tidak Dimodifikasi di baris status, yang memberi tahu cache bahwa ia dapat melanjutkan dan untuk meneruskan salinan objek yang di-cache (cache proxy) ke browser yang meminta.

Ini mengakhiri diskusi kita tentang HTTP, protokol Internet pertama (protokol lapisan aplikasi) yang telah kita pelajari secara mendetail. Kami telah melihat format pesan HTTP dan tindakan yang diambil oleh klien Web dan server saat pesan ini dikirim dan diterima.

Kami juga telah mempelajari sedikit infrastruktur aplikasi Web, termasuk cache, cookie, dan database back-end, yang semuanya terkait dengan protokol HTTP.

2.3 Transfer File: FTP

Dalam sesi FTP biasa, pengguna duduk di depan satu host (host lokal) dan ingin mentransfer file ke atau dari host jarak jauh. Agar pengguna dapat mengakses akun jarak jauh, pengguna harus memberikan identifikasi pengguna dan kata sandi. Setelah memberikan informasi otorisasi ini, pengguna dapat mentransfer file dari sistem file lokal ke sistem file jarak jauh dan sebaliknya. Seperti yang ditunjukkan pada Gambar 2.14, pengguna berinteraksi dengan FTP melalui agen pengguna FTP. Pengguna pertama memberikan nama host dari host jarak jauh, menyebabkan proses klien FTP di host lokal untuk membuat koneksi TCP dengan proses server FTP di host jarak jauh. Pengguna kemudian memberikan identifikasi pengguna dan kata sandi, yang dikirim melalui koneksi TCP sebagai bagian dari perintah FTP. Setelah server mengotorisasi pengguna, pengguna menyalin satu atau lebih file yang disimpan di sistem file lokal ke sistem file jarak jauh (atau sebaliknya).



Gambar 2.14 FTP memindahkan file antara sistem file lokal dan jarak jauh



Gambar 2.15 Kontrol dan koneksi data

HTTP dan FTP keduanya merupakan protokol transfer file dan memiliki banyak karakteristik umum; misalnya, keduanya berjalan di atas TCP. Namun, kedua protokol lapisan aplikasi tersebut memiliki beberapa perbedaan penting. Perbedaan yang paling mencolok adalah FTP menggunakan dua koneksi TCP paralel untuk mentransfer file, **koneksi kontrol**, dan **koneksi data**. Koneksi kontrol digunakan untuk mengirimkan informasi kontrol antara dua host — informasi seperti identifikasi pengguna, kata sandi, perintah untuk mengubah direktori jarak jauh, dan perintah untuk "meletakkan" dan "mendapatkan" file. Koneksi data digunakan untuk benar-benar mengirim file. Karena FTP menggunakan koneksi kontrol yang terpisah, FTP dikatakan mengirimkan informasi kontrolnya **out-of-band**. HTTP, seperti yang Anda ingat, mengirimkan baris header permintaan dan respons ke koneksi TCP yang sama yang membawa file yang ditransfer itu sendiri. Untuk alasan ini, HTTP dikatakan mengirim informasi kontrolnya **di-band**. Pada bagian selanjutnya, kita akan melihat bahwa SMTP, protokol utama untuk surat elektronik, juga mengirimkan informasi kontrol dalam jalur. Kontrol FTP dan koneksi data diilustrasikan pada Gambar 2.15.

Ketika seorang pengguna memulai sesi FTP dengan host jarak jauh, sisi klien FTP (pengguna) pertama-tama memulai koneksi TCP kontrol dengan sisi server (host jarak jauh) pada nomor port server 21. Sisi klien FTP mengirimkan identifikasi pengguna dan kata sandi melalui koneksi kontrol ini. Sisi klien FTP juga mengirimkan, melalui koneksi kontrol, perintah untuk mengubah direktori jarak jauh. Ketika sisi server menerima perintah untuk transfer file melalui koneksi kontrol (baik ke atau dari host jarak jauh), sisi server memulai koneksi data TCP ke sisi klien. FTP mengirimkan tepat satu file melalui koneksi data dan kemudian menutup koneksi data. Jika, selama sesi yang sama, pengguna ingin mentransfer file lain, FTP membuka koneksi data lain. Dengan demikian, dengan FTP, koneksi kontrol tetap terbuka sepanjang sesi pengguna, tetapi koneksi data baru dibuat untuk setiap file yang ditransfer dalam satu sesi (yaitu, koneksi data tidak tetap).

Sepanjang sesi, server FTP harus mempertahankan **keadaan** tentang pengguna. Khususnya, server harus mengaitkan koneksi kontrol dengan akun pengguna tertentu, dan server harus melacak direktori pengguna saat ini saat pengguna menjelajahi pohon direktori jarak jauh. Melacak informasi status ini untuk setiap sesi pengguna yang sedang berlangsung secara signifikan membatasi jumlah total sesi yang dapat dipertahankan FTP secara bersamaan. Ingatlah bahwa HTTP, di sisi lain, tidak memiliki kewarganegaraan—tidak harus melacak status pengguna apa pun.

2.3.1 Perintah dan Balasan FTP

Kami mengakhiri bagian ini dengan diskusi singkat tentang beberapa perintah dan balasan FTP yang lebih umum. Perintah, dari klien ke server, dan balasan, dari server ke klien, dikirim melalui koneksi kontrol dalam format ASCII 7-bit. Jadi, seperti perintah HTTP, perintah FTP dapat dibaca oleh orang-orang. Untuk menggambarkan perintah yang berurutan, carriage return dan line feed mengakhiri setiap perintah. Setiap perintah terdiri dari empat karakter ASCII huruf besar, beberapa dengan argumen opsional. Beberapa perintah yang lebih umum diberikan di bawah ini:

- USER username: Digunakan untuk mengirimkan identifikasi pengguna ke server.
- PASS password: Digunakan untuk mengirim password user ke server.
- LIST: Digunakan untuk meminta server mengirimkan kembali daftar semua file di direktori jarak jauh saat ini. Daftar file dikirim melalui koneksi data (baru dan non-persisten) daripada koneksi TCP kontrol.
- Nama file RETR: Digunakan untuk mengambil (yaitu, mendapatkan) file dari direktori host jarak jauh saat ini. Perintah ini menyebabkan remote host untuk memulai koneksi data dan mengirim file yang diminta melalui koneksi data.
- Nama file STOR: Digunakan untuk menyimpan (yaitu, meletakkan) file ke dalam direktori saat ini dari host jarak jauh.

Biasanya ada korespondensi satu-ke-satu antara perintah yang dikeluarkan pengguna dan perintah FTP yang dikirim melalui koneksi kontrol. Setiap perintah diikuti dengan balasan, dikirim dari server ke klien. Balasannya adalah angka tiga digit, dengan pesan opsional mengikuti nomor tersebut. Strukturnya serupa dengan kode status dan frasa di baris status pesan respons HTTP.

Beberapa balasan umum, beserta kemungkinan pesannya, adalah sebagai berikut:

- 331 Nama pengguna OK, kata sandi diperlukan • 125 Koneksi data sudah terbuka; transfer dimulai • 425 Tidak dapat membuka koneksi data • 452 Kesalahan penulisan file

Pembaca yang tertarik mempelajari tentang perintah dan balasan FTP lainnya disarankan untuk membaca RFC 959.

2.4 Surat Elektronik di Internet

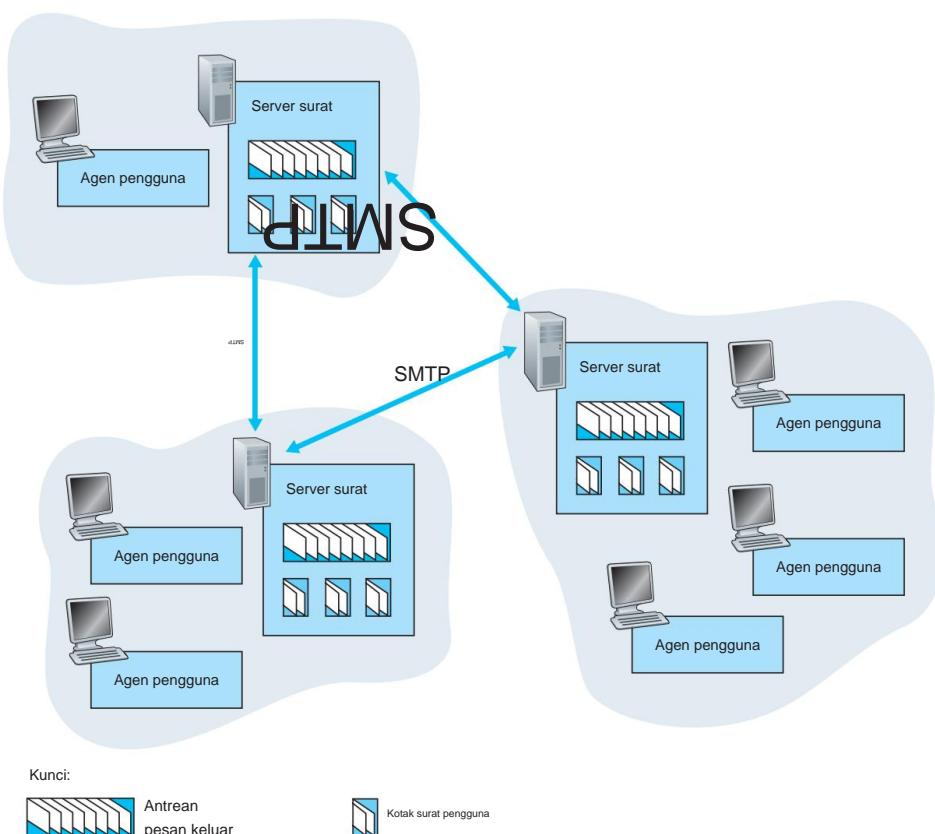
Surat elektronik telah ada sejak awal Internet. Itu adalah aplikasi paling populer ketika Internet masih dalam masa pertumbuhan [Segaller 1998], dan telah

menjadi lebih dan lebih rumit dan kuat selama bertahun-tahun. Itu tetap menjadi salah satu aplikasi Internet yang paling penting dan digunakan.

Seperti halnya surat pos biasa, email adalah media komunikasi asinkron — orang mengirim dan membaca pesan saat nyaman bagi mereka, tanpa harus berkoordinasi dengan jadwal orang lain. Berbeda dengan surat pos, surat elektronik cepat, mudah didistribusikan, dan tidak mahal. E-mail modern memiliki banyak fitur canggih, termasuk pesan dengan lampiran, hyperlink, teks berformat HTML, dan foto tersemat.

Pada bagian ini, kami memeriksa protokol lapisan aplikasi yang merupakan inti dari email Internet. Namun sebelum kita masuk ke diskusi mendalam tentang protokol ini, mari kita lihat sistem email Internet tingkat tinggi dan komponennya.

Gambar 2.16 menyajikan tampilan tingkat tinggi dari sistem email Internet. Kami melihat dari diagram ini bahwa ia memiliki tiga komponen utama: **agen pengguna**, **server email**, dan



Gambar 2.16 Tampilan tingkat tinggi dari sistem email Internet

120 BAB 2 • LAPISAN APLIKASI

Protokol Transfer Surat Sederhana (SMTP). Kami sekarang menjelaskan masing-masing komponen ini dalam konteks pengirim, Alice, mengirim pesan email ke penerima, Bob. Agen pengguna memungkinkan pengguna untuk membaca, membalas, meneruskan, menyimpan, dan menulis pesan. Microsoft Outlook dan Apple Mail adalah contoh agen pengguna untuk email. Ketika Alice selesai menulis pesannya, agen penggunanya mengirim pesan ke server emailnya, di mana pesan tersebut ditempatkan di antrian pesan keluar server email. Saat Bob ingin membaca pesan, agen penggunanya mengambil pesan dari kotak suratnya di server suratnya.

Server email membentuk inti dari infrastruktur email. Setiap penerima, seperti Bob, memiliki **kotak surat** yang terletak di salah satu server surat. Kotak surat Bob mengelola dan memelihara pesan yang telah dikirimkan kepadanya. Pesan tipikal memulai perjalannya di agen pengguna pengirim, melakukan perjalanan ke server surat pengirim, dan melakukan perjalanan ke server surat penerima, di mana pesan disimpan di kotak surat penerima.

SEJARAH KASUS

EMAIL WEB

Pada bulan Desember 1995, hanya beberapa tahun setelah Web " diciptakan ", Sabeer Bhatia dan Jack Smith mengunjungi pemodal ventura Internet Draper Fisher Jurvetson dan mengusulkan pengembangan sistem email berbasis Web gratis. Ideya adalah untuk memberikan akun email gratis kepada siapa saja yang menginginkannya, dan membuat akun tersebut dapat diakses dari Web. Sebagai imbalan atas 15 persen saham perusahaan, Draper Fisher Jurvetson membiayai Bhatia dan Smith, yang membentuk perusahaan bernama Hotmail.

Dengan tiga orang penuh waktu dan 14 orang paruh waktu yang bekerja untuk opsi saham, mereka dapat mengembangkan dan meluncurkan layanan tersebut pada Juli 1996. Dalam sebulan setelah diluncurkan, mereka memiliki 100.000 pelanggan. Pada bulan Desember 1997, kurang dari 18 bulan setelah meluncurkan layanan tersebut, Hotmail memiliki lebih dari 12 juta pelanggan dan diakuisisi oleh Microsoft, dilaporkan seharga \$400 juta. Keberhasilan Hotmail sering dikaitkan dengan "keunggulan penggerak pertama" dan "pemasaran viral" intrinsik email. (Mungkin beberapa siswa yang membaca buku ini akan menjadi salah satu pengusaha baru yang menyusun dan mengembangkan layanan Internet penggerak pertama dengan pemasaran viral yang melekat.)

Email web terus berkembang, menjadi lebih canggih dan kuat setiap hari tahun. Salah satu layanan paling populer saat ini adalah Google gmail, yang menawarkan giga byte penyimpanan gratis, pemfilteran spam tingkat lanjut dan deteksi virus, enkripsi email (menggunakan SSL), pengambilan email dari layanan email pihak ketiga, dan pencarian-berorientasi antar muka. Perpesanan asinkron dalam jejaring sosial, seperti Facebook, juga menjadi populer dalam beberapa tahun terakhir.

Saat Bob ingin mengakses pesan di kotak suratnya, server surat yang berisi kotak suratnya mengautentikasi Bob (dengan nama pengguna dan kata sandi). Server surat Alice juga harus menangani kegagalan di server surat Bob. Jika server Alice tidak dapat mengirimkan email ke server Bob, server Alice menyimpan pesan dalam **antrean pesan** dan mencoba mentransfer pesan nanti. Upaya ulang sering dilakukan setiap 30 menit atau lebih; jika tidak berhasil setelah beberapa hari, server menghapus pesan tersebut dan memberi tahu pengirim (Alice) dengan pesan email.

SMTP adalah protokol lapisan aplikasi utama untuk surat elektronik Internet. Ia menggunakan layanan transfer data TCP yang andal untuk mentransfer surat dari server surat pengirim ke server surat penerima. Seperti kebanyakan protokol lapisan aplikasi, SMTP memiliki dua sisi: sisi klien, yang mengeksekusi di server email pengirim, dan sisi server, yang mengeksekusi di server email penerima. Sisi klien dan server SMTP berjalan di setiap server email. Saat server surat mengirimkan surat ke server surat lain, ia bertindak sebagai klien SMTP. Saat server surat menerima surat dari server surat lain, ia bertindak sebagai server SMTP.

2.4.1 SMTP

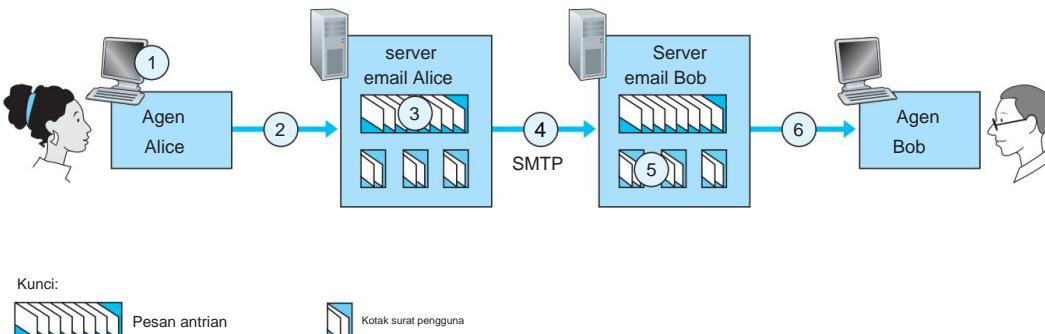
SMTP, didefinisikan dalam RFC 5321, adalah inti dari surat elektronik Internet. Seperti disebutkan di atas, SMTP mentransfer pesan dari server surat pengirim ke server surat penerima. SMTP jauh lebih tua dari HTTP. (SMTP RFC asli berasal dari tahun 1982, dan SMTP sudah ada jauh sebelum itu.) Meskipun SMTP memiliki banyak kualitas luar biasa, sebagaimana dibuktikan dengan keberadaannya di mana-mana di Internet, ini tetap merupakan teknologi warisan yang memiliki karakteristik kuno tertentu.

Misalnya, ini membatasi isi (bukan hanya header) semua pesan email menjadi ASCII 7-bit sederhana. Pembatasan ini masuk akal pada awal 1980-an ketika kapasitas transmisi langka dan tidak ada yang mengirim email lampiran besar atau file gambar, audio, atau video besar. Namun saat ini, di era multimedia, pembatasan ASCII 7-bit agak menyusahkan—membutuhkan data multimedia biner untuk dikodekan ke ASCII sebelum dikirim melalui SMTP; dan itu membutuhkan pesan ASCII yang sesuai untuk didekodekan kembali ke biner setelah transpor SMTP. Ingat dari Bagian 2.2 bahwa HTTP tidak memerlukan data multimedia untuk dikodekan ASCII sebelum transfer.

Untuk mengilustrasikan operasi dasar SMTP, mari kita telusuri pemandangan umum nario. Misalkan Alice ingin mengirimkan pesan ASCII sederhana kepada Bob.

1. Alice memanggil agen penggunanya untuk email, memberikan alamat email Bob (untuk contoh, bob@someschool.edu), menulis pesan, dan menginstruksikan agen pengguna untuk mengirim pesan.
2. Agen pengguna Alice mengirim pesan ke server suratnya, di mana pesan itu ditempatkan di a pesan buntut.

122 BAB 2 • LAPISAN APLIKASI

**Gambar 2.17** Alice mengirim pesan ke Bob

3. Sisi klien SMTP, yang berjalan di server surat Alice, melihat pesan dalam antrean pesan. Ini membuka koneksi TCP ke server SMTP, berjalan di server surat Bob.
4. Setelah beberapa jabat tangan SMTP awal, klien SMTP mengirimkan pesan Alice ke dalam koneksi TCP.
5. Di server surat Bob, sisi server SMTP menerima pesan. Server surat Bob kemudian menempatkan pesan di kotak surat Bob.
6. Bob meminta agen penggunanya untuk membaca pesan tersebut sesuai keinginannya.

Skenario dirangkum dalam Gambar 2.17.

Penting untuk diperhatikan bahwa SMTP biasanya tidak menggunakan server surat perantara untuk mengirim surat, bahkan ketika dua server surat terletak di ujung dunia yang berlawanan. Jika server Alice berada di Hong Kong dan server Bob berada di St. Louis, koneksi TCP adalah koneksi langsung antara server Hong Kong dan St. Louis. Secara khusus, jika server surat Bob sedang tidak aktif, pesan tetap berada di server surat Alice dan menunggu upaya baru—pesan tidak ditempatkan di beberapa server surat perantara.

Sekarang mari kita lihat lebih dekat bagaimana SMTP mentransfer pesan dari server surat pengirim ke server surat penerima. Kita akan melihat bahwa protokol SMTP memiliki banyak kesamaan dengan protokol yang digunakan untuk interaksi manusia secara tatap muka. Pertama, SMTP klien (berjalan di host server email pengirim) memiliki TCP yang membuat koneksi ke port 25 di server SMTP (berjalan di host server email penerima). Jika server sedang down, klien mencoba lagi nanti. Setelah koneksi ini dibuat, server dan klien melakukan beberapa jabat tangan lapisan aplikasi — seperti halnya manusia sering memperkenalkan diri sebelum mentransfer informasi dari satu ke yang lain, klien SMTP dan server memperkenalkan diri sebelum mentransfer informasi. Selama fase jabat tangan SMTP ini,

Klien SMTP menunjukkan alamat email pengirim (orang yang membuat pesan) dan alamat email penerima. Setelah klien SMTP dan server memperkenalkan diri satu sama lain, klien mengirimkan pesan.

SMTP dapat mengandalkan layanan transfer data TCP yang andal untuk mengirimkan pesan ke server tanpa kesalahan. Klien kemudian mengulangi proses ini melalui koneksi TCP yang sama jika memiliki pesan lain untuk dikirim ke server; jika tidak, ia menginstruksikan TCP untuk menutup koneksi.

Selanjutnya mari kita lihat contoh transkrip pesan yang dipertukarkan antara klien SMTP (C) dan server SMTP (S). Nama host klien adalah crepes.fr dan nama host server adalah hamburger.edu. Baris teks ASCII yang diawali dengan C: persis seperti baris yang dikirim klien ke soket TCP-nya, dan baris teks ASCII yang diawali dengan S: persis seperti baris yang dikirim server ke soket TCP-nya. Transkrip berikut dimulai setelah koneksi TCP dibuat.

```
S: 220 hamburger.edu C: HELO  
crepes.fr S: 250 Halo crepes.fr,  
senang bertemu denganmu C: MAIL FROM: <alice@crepes.fr> S: 250  
alice@crepes.fr ... Pengirim ok C : RCPT KE: <bob@hamburger.edu> S:  
250 bob@hamburger.edu... Penerima ok C: DATA
```

```
S: 354 Masukkan email, diakhiri dengan “.” pada baris dengan sendirinya C: Apakah  
Anda suka saus tomat?
```

```
C: Bagaimana dengan acar?  
C: .
```

```
S: 250 Pesan diterima untuk pengiriman C: QUIT S: 221  
hamburger.edu menutup koneksi
```

Pada contoh di atas, klien mengirimkan pesan (“Apakah Anda suka saus tomat? Bagaimana dengan acar?”) dari mail server crepes.fr ke mail server ham burger.edu. Sebagai bagian dari dialog, klien mengeluarkan lima perintah: HELO (singkatan dari HELLO), MAIL FROM, RCPT TO, DATA, dan QUIT. Perintah-perintah ini cukup jelas. Klien juga mengirimkan baris yang terdiri dari satu titik, yang menunjukkan akhir pesan ke server. (Dalam jargon ASCII, setiap pesan diakhiri dengan CRLF.CRLF, di mana CR dan LF masing-masing berarti carriage return dan line feed.) Server mengeluarkan balasan untuk setiap perintah, dengan setiap balasan memiliki kode balasan dan beberapa (opsional) penjelasan bahasa Inggris. Kami sebutkan di sini bahwa SMTP menggunakan koneksi persisten: Jika server email pengirim memiliki beberapa pesan untuk dikirim ke server email penerima yang sama, ia dapat mengirim semua pesan melalui koneksi TCP yang sama. Untuk setiap pesan, klien memulai proses dengan

124 BAB 2 • LAPISAN APLIKASI

MAIL FROM baru: crepes.fr, menandai akhir pesan dengan periode terisolasi, dan mengeluarkan QUIT hanya setelah semua pesan terkirim.

Sangat disarankan agar Anda menggunakan Telnet untuk melakukan dialog langsung dengan server SMTP. Untuk melakukan ini, masalah

```
nama server telnet 25
```

di mana namaserver adalah nama server email lokal. Saat Anda melakukan ini, Anda hanya membuat koneksi TCP antara host lokal Anda dan server email.

Setelah mengetik baris ini, Anda akan segera menerima balasan 220 dari server. Kemudian keluarkan perintah SMTP HELO, MAIL FROM, RCPT TO, DATA, CRLF.CRLF, dan QUIT pada waktu yang tepat. Juga sangat disarankan agar Anda mengerjakan Tugas Pemrograman 3 di akhir bab ini. Dalam penugasan itu, Anda akan membuat agen pengguna sederhana yang mengimplementasikan sisi klien dari SMTP.

Ini akan memungkinkan Anda untuk mengirim pesan email ke penerima sewenang-wenang melalui server email lokal.

2.4.2 Perbandingan dengan HTTP

Sekarang mari kita bandingkan secara singkat SMTP dengan HTTP. Kedua protokol digunakan untuk mentransfer file dari satu host ke host lainnya: HTTP mentransfer file (juga disebut objek) dari server Web ke klien Web (biasanya browser); SMTP mentransfer file (yaitu pesan email) dari satu server email ke server email lain. Saat mentransfer file, HTTP dan SMTP persisten menggunakan koneksi persisten. Dengan demikian, kedua protokol tersebut memiliki karakteristik yang sama. Namun, ada perbedaan penting. Pertama, HTTP terutama merupakan protokol **tarik** — seseorang memuat informasi di server Web dan pengguna menggunakan HTTP untuk menarik informasi dari server sesuai keinginan mereka. Khususnya, koneksi TCP diprakarsai oleh mesin yang ingin menerima file.

Di sisi lain, SMTP pada dasarnya adalah **protokol push**— server email pengirim mendorong file ke server email penerima. Khususnya, koneksi TCP dimulai oleh mesin yang ingin mengirim file.

Perbedaan kedua, yang telah kita singgung sebelumnya, adalah bahwa SMTP mensyaratkan setiap pesan, termasuk isi setiap pesan, dalam format ASCII 7-bit. Jika pesan berisi karakter yang bukan ASCII 7-bit (misalnya, karakter Prancis dengan aksen) atau berisi data biner (seperti file gambar), maka pesan harus dikodekan menjadi ASCII 7-bit. Data HTTP tidak memberlakukan batasan ini.

Perbedaan penting ketiga menyangkut bagaimana dokumen yang terdiri dari teks dan gambar (bersama dengan kemungkinan jenis media lainnya) ditangani. Seperti yang kita pelajari di Bagian 2.2, HTTP mengenkapsulasi setiap objek dalam pesan respons HTTP-nya sendiri. Surat Internet menempatkan semua objek pesan ke dalam satu pesan.

2.4.3 Format Pesan Surat

Ketika Alice menulis surat biasa untuk Bob, dia mungkin menyertakan semua jenis informasi tajuk periferal di bagian atas surat, seperti alamat Bob, alamat pengirimnya sendiri, dan tanggal. Demikian pula, ketika pesan email dikirim dari satu orang ke orang lain, header yang berisi informasi periferal mendahului badan pesan itu sendiri. Informasi periferal ini terkandung dalam serangkaian baris header, yang didefinisikan dalam RFC 5322. Baris header dan isi pesan dipisahkan oleh baris kosong (yaitu, dengan CRLF). RFC 5322 menentukan format yang tepat untuk baris header email serta interpretasi semantiknya. Seperti halnya HTTP, setiap baris header berisi teks yang dapat dibaca, terdiri dari kata kunci diikuti titik dua diikuti dengan nilai. Beberapa kata kunci diperlukan dan lainnya bersifat opsional. Setiap tajuk harus memiliki Dari: baris tajuk dan To: baris tajuk; tajuk dapat menyertakan Subjek: baris tajuk serta baris tajuk opsional lainnya. Penting untuk dicatat bahwa baris header ini *berbeda* dari perintah SMTP yang kita pelajari di Bagian 2.4.1 (walaupun mengandung beberapa kata umum seperti "dari" dan "ke"). Perintah di bagian itu adalah bagian dari protokol jabat tangan SMTP; baris header yang diperiksa di bagian ini adalah bagian dari pesan email itu sendiri.

Header pesan tipikal terlihat seperti ini:

```
From: alice@crepes.fr To:  
bob@hamburger.edu Subject:  
Mencari arti hidup.
```

Setelah header pesan, baris kosong mengikuti; kemudian isi pesan (dalam ASCII) mengikuti. Anda harus menggunakan Telnet untuk mengirim pesan ke server email yang berisi beberapa baris header, termasuk baris header Subject:. Untuk melakukannya, terbitkan telnet serverName 25, seperti yang dibahas di Bagian 2.4.1.

2.4.4 Protokol Akses Surat

Setelah SMTP mengirimkan pesan dari server surat Alice ke server surat Bob, pesan tersebut ditempatkan di kotak surat Bob. Sepanjang diskusi ini kami diam-diam berasumsi bahwa Bob membaca emailnya dengan masuk ke host server dan kemudian menjalankan pembaca email yang berjalan di host tersebut. Hingga awal 1990-an, ini adalah cara standar dalam melakukan sesuatu. Namun saat ini, akses email menggunakan arsitektur klien-server—pengguna biasa membaca email dengan klien yang dijalankan di sistem akhir pengguna, misalnya, di PC kantor, laptop, atau smartphone. Dengan menjalankan klien email di PC lokal, pengguna menikmati serangkaian fitur yang kaya, termasuk kemampuan untuk melihat pesan multimedia dan lampiran.

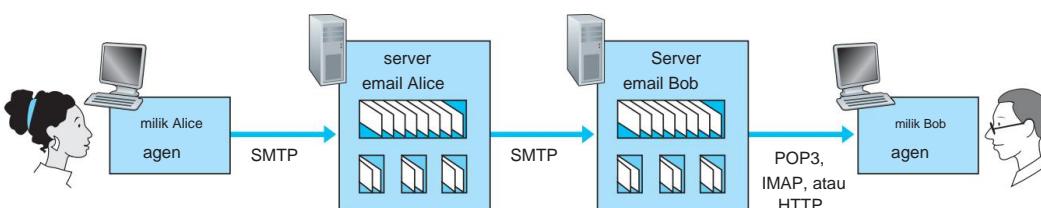
Mengingat bahwa Bob (penerima) mengeksekusi agen penggunanya di PC lokalnya, wajar untuk mempertimbangkan untuk menempatkan server email di PC lokalnya juga. Dengan pendekatan ini,

126 BAB 2 • LAPISAN APLIKASI

Server surat Alice akan berdialog langsung dengan PC Bob. Namun, ada masalah dengan pendekatan ini. Ingatlah bahwa server surat mengelola kotak surat dan menjalankan sisi klien dan server SMTP. Jika server email Bob berada di PC lokalnya, maka PC Bob harus selalu hidup, dan terhubung ke Internet, untuk menerima email baru, yang dapat tiba kapan saja. Ini tidak praktis bagi banyak pengguna Internet. Sebagai gantinya, pengguna biasa menjalankan agen pengguna di PC lokal tetapi mengakses kotak suratnya yang disimpan di server surat bersama yang selalu aktif. Server email ini digunakan bersama dengan pengguna lain dan biasanya dikelola oleh ISP pengguna (misalnya, universitas atau perusahaan).

Sekarang mari kita pertimbangkan jalur yang diambil pesan email saat dikirim dari Alice ke Bob. Kami baru mengetahui bahwa di beberapa titik di sepanjang jalur, pesan email perlu disimpan di server surat Bob. Ini dapat dilakukan hanya dengan meminta agen pengguna Alice mengirim pesan langsung ke server surat Bob. Dan ini bisa dilakukan dengan SMTP—memang, SMTP telah dirancang untuk mendorong email dari satu host ke host lainnya. Namun, biasanya agen pengguna pengirim tidak berdialog langsung dengan server email penerima. Sebaliknya, seperti yang ditunjukkan pada Gambar 2.18, agen pengguna Alice menggunakan SMTP untuk mendorong pesan email ke server emailnya, kemudian server email Alice menggunakan SMTP (sebagai klien SMTP) untuk menyampaikan pesan email ke server email Bob. Mengapa prosedur dua langkah? Terutama karena tanpa menyampaikan melalui server email Alice, agen pengguna Alice tidak memiliki jalan lain ke server email tujuan yang tidak dapat dijangkau. Dengan meminta Alice terlebih dahulu menyimpan email di server suratnya sendiri, server surat Alice dapat berulang kali mencoba mengirim pesan ke server surat Bob, katakanlah setiap 30 menit, hingga server surat Bob beroperasi. (Dan jika server email Alice sedang down, maka dia memiliki jalan lain untuk mengeluh kepada administrator sistemnya!) RFC SMTP menentukan bagaimana perintah SMTP dapat digunakan untuk menyampaikan pesan di beberapa server SMTP.

Tapi masih ada satu bagian yang hilang dari teka-teki itu! Bagaimana penerima seperti Bob, yang menjalankan agen pengguna di PC lokalnya, mendapatkan pesannya, yang berada di server surat dalam ISP Bob? Perhatikan bahwa agen pengguna Bob tidak dapat menggunakan SMTP untuk mendapatkan pesan karena mendapatkan pesan adalah operasi penarikan, sedangkan SMTP adalah



Gambar 2.18 Protokol email dan entitas komunikasinya

protokol dorong. Teka-teki ini diselesaikan dengan memperkenalkan protokol akses surat khusus yang mentransfer pesan dari server surat Bob ke PC lokalnya. Saat ini ada sejumlah protokol akses email yang populer, termasuk **Post Office Protocol—Versi 3 (POP3)**, **Internet Mail Access Protocol (IMAP)**, dan **HTTP**.

Gambar 2.18 memberikan ringkasan protokol yang digunakan untuk surat Internet: SMTP digunakan untuk mentransfer surat dari server surat pengirim ke server surat penerima; SMTP juga digunakan untuk mentransfer surat dari agen pengguna pengirim ke server surat pengirim. Protokol akses email, seperti POP3, digunakan untuk mentransfer email dari server email penerima ke agen pengguna penerima.

POP3

POP3 adalah protokol akses email yang sangat sederhana. Itu didefinisikan dalam [RFC 1939], yang pendek dan cukup mudah dibaca. Karena protokolnya sangat sederhana, fungsinya agak terbatas. POP3 dimulai ketika agen pengguna (klien) membuka koneksi TCP ke server email (server) pada port 110. Dengan koneksi TCP dibuat, POP3 berkembang melalui tiga fase: otorisasi, transaksi, dan pembaruan.

Selama fase pertama, otorisasi, agen pengguna mengirimkan nama pengguna dan kata sandi (dengan jelas) untuk mengautentikasi pengguna. Selama fase kedua, transaksi, agen pengguna mengambil pesan; juga selama fase ini, agen pengguna dapat menandai pesan untuk dihapus, menghapus tanda penghapusan, dan mendapatkan statistik email. Fase ketiga, pembaruan, terjadi setelah klien mengeluarkan perintah keluar, mengakhiri sesi POP3; saat ini, server email menghapus pesan yang ditandai untuk dihapus.

Dalam transaksi POP3, agen pengguna mengeluarkan perintah, dan server merespons setiap perintah dengan balasan. Ada dua kemungkinan respons: +OK (terkadang diikuti oleh data server-ke-klien), digunakan oleh server untuk menunjukkan bahwa perintah sebelumnya baik-baik saja; dan -ERR, digunakan oleh server untuk menunjukkan bahwa ada yang salah dengan perintah sebelumnya.

Fase otorisasi memiliki dua perintah utama: user <username> dan pass <password>. Untuk mengilustrasikan kedua perintah ini, kami sarankan Anda melakukan Telnet langsung ke server POP3, menggunakan port 110, dan mengeluarkan perintah ini. Misalkan mailServer adalah nama server email Anda. Anda akan melihat sesuatu seperti:

```
server surat telnet 110
+ OK server POP3 siap pengguna
bob
+ Oke
lulus lapar
+OK pengguna berhasil masuk
```

Jika Anda salah mengeja perintah, server POP3 akan membalas dengan pesan -ERR.

128 BAB 2 • LAPISAN APLIKASI

Sekarang mari kita lihat fase transaksi. Agen pengguna yang menggunakan POP3 seringkali dapat dikonfigurasi (oleh pengguna) untuk "mengunduh dan menghapus" atau "mengunduh dan menyimpan". Urutan perintah yang dikeluarkan oleh agen pengguna POP3 tergantung pada mana dari dua mode ini agen pengguna beroperasi. Dalam mode unduh-dan-hapus, agen pengguna akan mengeluarkan perintah daftar, retr, dan hapus. Sebagai contoh, misalkan pengguna memiliki dua pesan di kotak suratnya. Dalam dialog di bawah ini, C: (singkatan dari klien) adalah agen pengguna dan S: (singkatan dari server) adalah server email. Transaksi akan terlihat seperti:

```
C: daftar  
S: 1 498  
S: 2 912  
S: .  
C: retr 1 S:  
(bla bla...  
S: .....  
S: .....bla)  
S: .  
C: dele 1 C:  
retr 2 S: (bla  
bla ...  
S: .....  
S: .....bla)  
S: .  
C: bagian 2  
C: berhenti  
S: +OK server POP3 ditutup
```

Agen pengguna pertama-tama meminta server email untuk mencantumkan ukuran setiap pesan yang disimpan. Agen pengguna kemudian mengambil dan menghapus setiap pesan dari server. Perhatikan bahwa setelah fase otorisasi, agen pengguna hanya menggunakan empat perintah: daftar, retr, hapus, dan keluar. Sintaks untuk perintah ini ditentukan dalam RFC 1939. Setelah memproses perintah keluar, server POP3 memasuki fase pembaruan dan menghapus pesan 1 dan 2 dari kotak surat.

Masalah dengan mode unduh-dan-hapus ini adalah penerima, Bob, mungkin nomaden dan mungkin ingin mengakses pesan emailnya dari beberapa mesin, misalnya, PC kantornya, PC rumahnya, dan komputer portabelnya. Mode unduh dan hapus mempartisi pesan surat Bob melalui tiga mesin ini; khususnya, jika Bob pertama kali membaca pesan di PC kantornya, dia tidak akan dapat membaca ulang pesan dari portabelnya di rumah nanti malam. Dalam mode unduh dan simpan, agen pengguna meninggalkan pesan di server surat setelah mengunduhnya. Dalam hal ini, Bob dapat membaca ulang pesan dari mesin yang berbeda; dia dapat mengakses pesan dari kantor dan mengaksesnya lagi di akhir minggu dari rumah.

Selama sesi POP3 antara agen pengguna dan server email, server POP3 menyimpan beberapa informasi status; khususnya, ini melacak pesan pengguna mana yang telah ditandai dihapus. Namun, server POP3 tidak membawa informasi status di seluruh sesi POP3. Kurangnya informasi status di seluruh sesi sangat menyederhanakan implementasi server POP3.

IMAP

Dengan akses POP3, setelah Bob mengunduh pesannya ke mesin lokal, dia dapat membuat folder surat dan memindahkan pesan yang diunduh ke dalam folder.

Bob kemudian dapat menghapus pesan, memindahkan pesan melintasi folder, dan mencari pesan (berdasarkan nama atau subjek pengirim). Namun paradigma ini—yaitu, folder dan pesan di mesin lokal—menimbulkan masalah bagi pengguna nomaden, yang lebih memilih mempertahankan hierarki folder di server jarak jauh yang dapat diakses dari komputer mana pun. Ini tidak mungkin dilakukan dengan POP3—protokol POP3 tidak menyediakan cara apa pun bagi pengguna untuk membuat folder jarak jauh dan menetapkan pesan ke folder.

Untuk mengatasi ini dan masalah lainnya, protokol IMAP, yang didefinisikan dalam [RFC 3501], diciptakan. Seperti POP3, IMAP adalah protokol akses email. Ini memiliki lebih banyak fitur daripada POP3, tetapi juga jauh lebih kompleks. (Dan dengan demikian implementasi sisi klien dan server secara signifikan lebih kompleks.)

Server IMAP akan mengaitkan setiap pesan dengan folder; ketika sebuah pesan pertama kali tiba di server, itu terkait dengan folder INBOX penerima. Penerima kemudian dapat memindahkan pesan ke folder baru buatan pengguna, membaca pesan, menghapus pesan, dan sebagainya. Protokol IMAP memberikan perintah untuk memungkinkan pengguna membuat folder dan memindahkan pesan dari satu folder ke folder lainnya. IMAP juga menyediakan perintah yang memungkinkan pengguna mencari folder jarak jauh untuk pesan yang cocok dengan kriteria tertentu. Perhatikan bahwa, tidak seperti POP3, server IMAP mempertahankan informasi status pengguna di seluruh sesi IMAP—misalnya, nama folder dan pesan mana yang terkait dengan folder mana.

Fitur penting lainnya dari IMAP adalah memiliki perintah yang mengizinkan agen pengguna untuk mendapatkan komponen pesan. Misalnya, agen pengguna dapat memperoleh hanya header pesan dari sebuah pesan atau hanya satu bagian dari pesan MIME multibagian.

Fitur ini berguna saat ada koneksi bandwidth rendah (misalnya, tautan modem berkecepatan lambat) antara agen pengguna dan server suratnya. Dengan koneksi bandwidth rendah, pengguna mungkin tidak ingin mengunduh semua pesan di kotak suratnya, khususnya menghindari pesan panjang yang mungkin berisi, misalnya, klip audio atau video.

Email Berbasis Web

Semakin banyak pengguna saat ini mengirim dan mengakses email mereka melalui browser Web mereka. Hotmail memperkenalkan akses berbasis Web pada pertengahan 1990-an. Sekarang berbasis web

130 BAB 2 • LAPISAN APLIKASI

e-mail juga disediakan oleh Google, Yahoo!, serta hampir semua universitas dan perusahaan besar. Dengan layanan ini, agen pengguna adalah browser Web biasa, dan pengguna berkomunikasi dengan kotak surat jarak jauhnya melalui HTTP. Ketika penerima, seperti Bob, ingin mengakses pesan di kotak suratnya, pesan email dikirim dari server surat Bob ke browser Bob menggunakan protokol HTTP daripada protokol POP3 atau IMAP. Saat pengirim, seperti Alice, ingin mengirim pesan email, pesan email dikirim dari browsernya ke server emailnya melalui HTTP, bukan melalui SMTP. Server email Alice, bagaimanapun, masih mengirim pesan ke, dan menerima pesan dari, server email lain menggunakan SMTP.

2.5 DNS—Layanan Direktori Internet

Kita manusia dapat diidentifikasi dengan banyak cara. Misalnya, kita bisa dikenali dari nama-nama yang tertera di akte kelahiran kita. Kami dapat diidentifikasi dengan nomor jaminan sosial kami. Kami dapat diidentifikasi dengan nomor SIM kami. Meskipun masing-masing pengidentifikasi ini dapat digunakan untuk mengidentifikasi orang, dalam konteks tertentu satu pengidentifikasi mungkin lebih tepat daripada yang lain. Misalnya, komputer di IRS (badan pemungut pajak terkenal di Amerika Serikat) lebih suka menggunakan nomor jaminan sosial dengan panjang tetap daripada nama akta kelahiran. Di sisi lain, masyarakat awam lebih memilih nama akte kelahiran yang lebih mnemonik daripada nomor jaminan sosial. (Memang, dapatkah Anda membayangkan mengatakan, "Hai. Nama saya 132-67-9875. Silakan temui suami saya, 178-87-1146.")

Sama seperti manusia dapat diidentifikasi dalam banyak cara, begitu juga dengan host Internet. Satu pengidentifikasi untuk sebuah host adalah **nama host-nya**. Nama host—seperti cnn.com, www.yahoo. com, gaia.cs.umass.edu, dan cis.poly.edu—adalah mnemonik dan karenanya dihargai oleh manusia. Namun, nama host memberikan sedikit, jika ada, informasi tentang lokasi dalam Internet dari host tersebut. (Nama host seperti www.eurecom.fr, yang diakhiri dengan kode negara .fr, memberi tahu kita bahwa host mungkin berada di Prancis, tetapi tidak menjelaskan lebih lanjut.) Selain itu, karena nama host dapat terdiri dari karakter alfanumerik dengan panjang variabel , mereka akan sulit diproses oleh router. Untuk alasan ini, host juga diidentifikasi dengan apa yang disebut **alamat IP**.

Kami membahas alamat IP secara rinci di Bab 4, tetapi berguna untuk mengatakan beberapa kata singkat tentangnya sekarang. Alamat IP terdiri dari empat byte dan memiliki struktur hierarki yang kaku. Alamat IP terlihat seperti 121.7.106.83, di mana setiap titik memisahkan salah satu byte yang dinyatakan dalam notasi desimal dari 0 hingga 255. Alamat IP bersifat hierarkis karena saat kami memindai alamat dari kiri ke kanan, kami memperoleh informasi yang lebih spesifik tentang di mana host berada di Internet (yaitu, di dalam jaringan mana, di jaringan jaringan). Demikian pula, saat kami memindai alamat pos dari bawah ke atas, kami memperoleh informasi yang lebih spesifik tentang lokasi penerima.

2.5.1 Layanan yang Disediakan oleh DNS

Kita baru saja melihat bahwa ada dua cara untuk mengidentifikasi host—dengan nama host dan alamat IP. Orang-orang lebih suka pengidentifikasi nama host yang lebih mnemonik, sementara router lebih memilih alamat IP yang terstruktur secara hierarkis dan panjang tetap. Untuk merekonsiliasi preferensi ini, kami memerlukan layanan direktori yang menerjemahkan nama host ke alamat IP. Ini adalah tugas utama **sistem nama domain** Internet (DNS). DNS adalah (1) basis data terdistribusi yang diimplementasikan dalam hierarki **server DNS**, dan (2) protokol lapisan aplikasi yang memungkinkan host untuk menanyakan basis data terdistribusi. Server DNS seringkali merupakan mesin UNIX yang menjalankan perangkat lunak Berkeley Internet Name Domain (BIND) [BIND 2012]. Protokol DNS berjalan di atas UDP dan menggunakan port 53.

DNS umumnya digunakan oleh protokol lapisan aplikasi lainnya—termasuk HTTP, SMTP, dan FTP—untuk menerjemahkan nama host yang disediakan pengguna ke alamat IP. Sebagai contoh, pertimbangkan apa yang terjadi ketika browser (yaitu, klien HTTP), berjalan di beberapa host pengguna, meminta URL www.someschool.edu/index.html. Agar host pengguna dapat mengirimkan pesan permintaan HTTP ke server Web www.someschool.edu, host pengguna harus mendapatkan alamat IP www.someschool.edu terlebih dahulu. Ini dilakukan sebagai berikut.

1. Mesin pengguna yang sama menjalankan sisi klien dari aplikasi DNS.
2. Browser mengekstrak nama host, www.someschool.edu, dari URL dan meneruskan nama host ke sisi klien aplikasi DNS.
3. Klien DNS mengirimkan kueri yang berisi nama host ke server DNS.
4. Klien DNS akhirnya menerima balasan, yang menyertakan alamat IP untuk nama host.
5. Setelah browser menerima alamat IP dari DNS, browser dapat memulai koneksi TCP ke proses server HTTP yang terletak di port 80 di alamat IP tersebut.

Kami melihat dari contoh ini bahwa DNS menambahkan penundaan tambahan—terkadang substansial—ke aplikasi Internet yang menggunakananya. Untungnya, seperti yang kita diskusikan di bawah, alamat IP yang diinginkan sering di-cache di server DNS "terdekat", yang membantu mengurangi lalu lintas jaringan DNS serta penundaan DNS rata-rata.

DNS menyediakan beberapa layanan penting lainnya selain menerjemahkan nama host ke alamat IP:

- **Host aliasing.** Host dengan nama host yang rumit dapat memiliki satu atau lebih nama alias. Misalnya, nama host seperti `relay1.west-coast.enterprise.com` dapat memiliki, katakanlah, dua alias seperti `enterprise.com` dan `www.enterprise.com`. Dalam hal ini, nama host `relay1.west-coast.enterprise.com` dikatakan sebagai **nama host kanonis**. Nama host alias, jika ada, biasanya lebih mnemonik daripada nama host kanonis.

132 BAB 2 • LAPISAN APLIKASI



PRINSIP DALAM PRAKTEK

DNS: FUNGSI JARINGAN KRITIS MELALUI CLIENT-SERVER

PARADIGMA

Seperti HTTP, FTP, dan SMTP, protokol DNS adalah protokol lapisan aplikasi karena (1) berjalan di antara sistem akhir yang berkomunikasi menggunakan paradigma client-server dan (2) bergantung pada protokol transport end-to-end yang mendasarinya untuk mentransfer Pesan DNS antara sistem akhir yang berkomunikasi. Namun, dalam pengertian lain, peran DNS sangat berbeda dari Web, transfer file, dan aplikasi email. Tidak seperti aplikasi ini, DNS bukanlah aplikasi yang berinteraksi langsung dengan pengguna. Sebaliknya, DNS menyediakan fungsi Internet inti—yaitu, menerjemahkan nama host ke alamat IP yang mendasarinya, untuk aplikasi pengguna dan perangkat lunak lain di Internet. Kami mencatat di Bagian 1.2 bahwa sebagian besar kerumitan dalam arsitektur Internet terletak di "tepi" jaringan. DNS, yang mengimplementasikan proses penerjemahan nama-ke-alamat kritis menggunakan klien dan server yang terletak di tepi jaringan, adalah contoh lain dari filosofi desain tersebut.

DNS dapat dipanggil oleh aplikasi untuk mendapatkan nama host kanonis untuk nama host alias yang disediakan serta alamat IP host. • **aliasing server surat.**

Untuk alasan yang jelas, sangat diharapkan alamat email menjadi mnemonik.

Misalnya, jika Bob memiliki akun di Hotmail, alamat email Bob mungkin sesederhana bob@hotmail.com. Namun, nama host dari server surat Hotmail lebih rumit dan jauh lebih sedikit mnemonik daripada sekadar hotmail.com (misalnya, nama host kanonis mungkin seperti relay1.west-coast.hotmail.com). DNS dapat dipanggil oleh aplikasi email untuk mendapatkan nama host kanonis untuk nama host alias yang disediakan serta alamat IP host. Faktanya, data MX (lihat di bawah) mengizinkan server email dan server Web perusahaan untuk memiliki nama host (alias) yang identik; misalnya, server Web dan server email perusahaan keduanya dapat disebut enterprise.com. • **Distribusi beban.** DNS juga digunakan untuk melakukan distribusi beban di antara server yang

direplikasi, seperti server Web yang direplikasi. Situs yang sibuk, seperti cnn.com, direplikasi melalui beberapa server, dengan setiap server berjalan pada sistem akhir yang berbeda dan masing-masing memiliki alamat IP yang berbeda. Untuk server Web yang direplikasi, satu set alamat IP dikaitkan dengan satu nama host kanonis. Basis data DNS berisi kumpulan alamat IP ini. Saat klien membuat kueri DNS untuk nama yang dipetakan ke sekumpulan alamat, server merespons dengan seluruh rangkaian alamat IP, tetapi merotasi urutan alamat dalam setiap balasan. Karena klien biasanya mengirimkan pesan permintaan HTTP-nya ke alamat IP yang tercantum pertama kali dalam set, rotasi DNS mendistribusikan lalu lintas di antara server yang direplikasi.

Rotasi DNS juga digunakan untuk email sehingga beberapa server email dapat memiliki nama alias yang sama. Juga, perusahaan distribusi konten seperti Akamai telah menggunakan DNS dengan cara yang lebih canggih [Dilley 2002] untuk menyediakan distribusi konten Web (lihat Bab 7).

DNS ditentukan dalam RFC 1034 dan RFC 1035, dan diperbarui dalam beberapa RFC tambahan. Ini adalah sistem yang kompleks, dan kami hanya menyentuh aspek kunci dari operasinya di sini. Pembaca yang tertarik dirujuk ke RFC ini dan buku oleh Albitz dan Liu [Albitz 1993]; lihat juga makalah retrospektif [Mockapetris 1988], yang memberikan deskripsi bagus tentang apa dan mengapa DNS, dan [Mockapetris 2005].

2.5.2 Tinjauan tentang Cara Kerja DNS

Kami sekarang menyajikan ikhtisar tingkat tinggi tentang cara kerja DNS. Diskusi kita akan fokus pada layanan terjemahan hostname-to-IP-address.

Misalkan beberapa aplikasi (seperti browser Web atau pembaca email) yang berjalan di host pengguna perlu menerjemahkan nama host ke alamat IP. Aplikasi akan memanggil sisi klien DNS, menentukan nama host yang perlu diterjemahkan. (Pada banyak mesin berbasis UNIX, `gethostbyname()` adalah panggilan fungsi yang dipanggil aplikasi untuk melakukan terjemahan.) DNS di host pengguna kemudian mengambil alih, mengirimkan pesan kueri ke jaringan. Semua kueri DNS dan pesan balasan dikirim dalam datagram UDP ke port 53. Setelah penundaan, mulai dari milidetik hingga detik, DNS di host pengguna menerima pesan balasan DNS yang menyediakan pemetaan yang diinginkan. Pemetaan ini kemudian diteruskan ke aplikasi pemanggil. Jadi, dari sudut pandang aplikasi pemanggil di host pengguna, DNS adalah kotak hitam yang menyediakan layanan terjemahan yang sederhana dan lugas. Namun kenyataannya, kotak hitam yang mengimplementasikan layanan ini kompleks, terdiri dari sejumlah besar server DNS yang didistribusikan di seluruh dunia, serta protokol lapisan aplikasi yang menentukan cara server DNS dan host kueri berkomunikasi.

Desain sederhana untuk DNS akan memiliki satu server DNS yang berisi semua ping peta. Dalam desain terpusat ini, klien cukup mengarahkan semua kueri ke server DNS tunggal, dan server DNS merespons langsung ke klien yang meminta. Meskipun kesederhanaan desain ini menarik, namun tidak sesuai untuk Internet saat ini, dengan jumlah host yang sangat banyak (dan terus bertambah). Masalah dengan desain terpusat meliputi:

- **Satu titik kegagalan.** Jika server DNS macet, begitu pula seluruh Internet! • **Volume lalu lintas.** Satu server DNS harus menangani semua kueri DNS (untuk semua permintaan HTTP dan pesan email yang dihasilkan dari ratusan juta host).

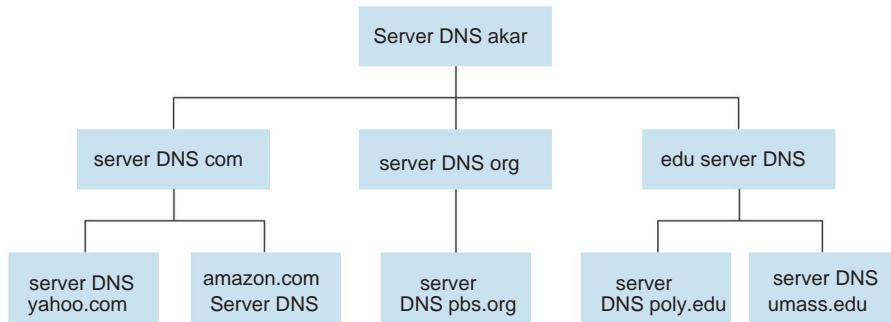
134 BAB 2 • LAPISAN APLIKASI

- **Jauh database terpusat.** Satu server DNS tidak dapat "mendekati" semua klien yang meminta. Jika kami menempatkan satu server DNS di New York City, maka semua kueri dari Australia harus melakukan perjalanan ke belahan dunia lain, mungkin melalui tautan yang lambat dan padat. Ini dapat menyebabkan penundaan yang signifikan.
- **Pemeliharaan.** Server DNS tunggal harus menyimpan catatan untuk semua host Internet. Database terpusat ini tidak hanya akan menjadi besar, tetapi juga harus sering diperbarui untuk memperhitungkan setiap host baru.

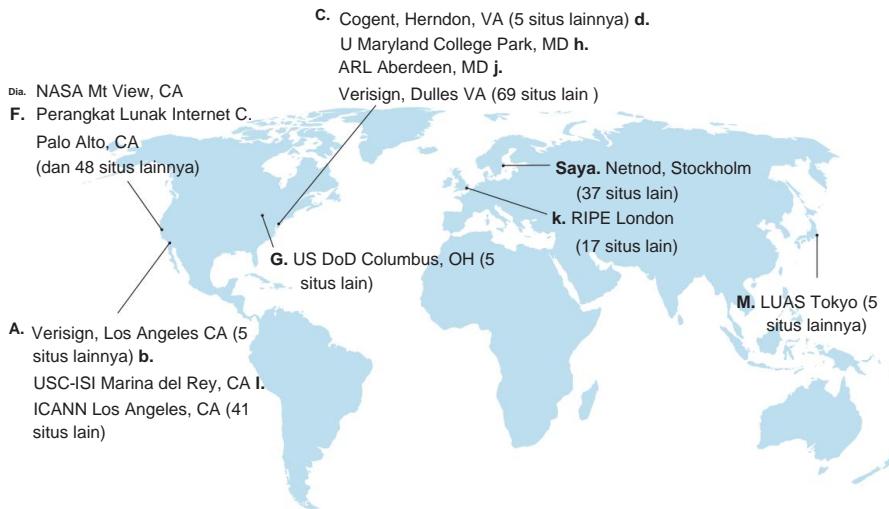
Singkatnya, database terpusat dalam satu server DNS *tidak dapat diskalakan*. Akibatnya, DNS didistribusikan dengan desain. Faktanya, DNS adalah contoh yang bagus tentang bagaimana database terdistribusi dapat diimplementasikan di Internet.

Database Hirarkis Terdistribusi

Untuk mengatasi masalah skala, DNS menggunakan sejumlah besar server, diatur secara hierarkis dan didistribusikan ke seluruh dunia. Tidak ada server DNS tunggal yang memiliki semua pemetaan untuk semua host di Internet. Sebaliknya, ping peta didistribusikan ke seluruh server DNS. Untuk perkiraan pertama, ada tiga kelas server DNS — server DNS akar, server DNS domain tingkat atas (TLD), dan server DNS otoritatif — diatur dalam hierarki seperti yang ditunjukkan pada Gambar 2.19. Untuk memahami bagaimana ketiga kelas server ini berinteraksi, misalkan klien DNS ingin menentukan alamat IP untuk nama host www.amazon.com. Untuk perkiraan pertama, acara berikut akan berlangsung. Klien pertama-tama menghubungi salah satu server root, yang mengembalikan alamat IP untuk server TLD untuk domain tingkat atas com. Klien kemudian menghubungi salah satu server TLD ini, yang mengembalikan alamat IP server resmi untuk amazon.com. Terakhir, klien menghubungi salah satu server resmi untuk amazon.com, yang mengembalikan alamat IP



Gambar 2.19 Bagian dari hirarki server DNS



Gambar 2.20 DNS root server tahun 2012 (nama, organisasi, lokasi)

untuk nama host `www.amazon.com`. Kami akan segera memeriksa proses pencarian DNS ini secara lebih mendetail. Tapi pertama-tama mari kita lihat lebih dekat ketiga kelas server DNS ini:

- **Akar server DNS.** Di Internet ada 13 server DNS root (berlabel A hingga M), sebagian besar berlokasi di Amerika Utara. Peta Oktober 2006 dari server DNS root ditunjukkan pada Gambar 2.20; daftar server DNS root saat ini tersedia melalui [Root-servers 2012]. Meskipun kami telah merujuk ke masing-masing dari 13 server DNS root seolah-olah itu adalah satu server, setiap "server" sebenarnya adalah jaringan server yang direplikasi, untuk tujuan keamanan dan keandalan. Secara keseluruhan, ada 247 root server pada musim gugur 2011.
- **Server domain tingkat atas (TLD).** Server ini bertanggung jawab atas domain tingkat atas seperti `.com`, `.org`, `.net`, `.edu`, dan `.gov`, dan semua domain tingkat atas negara seperti `.uk`, `.fr`, `.ca`, dan `.jp`. Perusahaan Verisign Global Registry Services mengelola server TLD untuk domain tingkat atas `.com`, dan perusahaan Educause mengelola server TLD untuk domain tingkat atas `.edu`. Lihat [IANA TLD 2012] untuk daftar semua domain tingkat atas.
- **Server DNS otoritatif.** Setiap organisasi dengan host yang dapat diakses publik (seperti server Web dan server email) di Internet harus menyediakan catatan DNS yang dapat diakses publik yang memetakan nama host tersebut ke alamat IP. Server DNS otoritatif organisasi menampung catatan DNS ini. Sebuah organisasi dapat

136 BAB 2 • LAPISAN APLIKASI

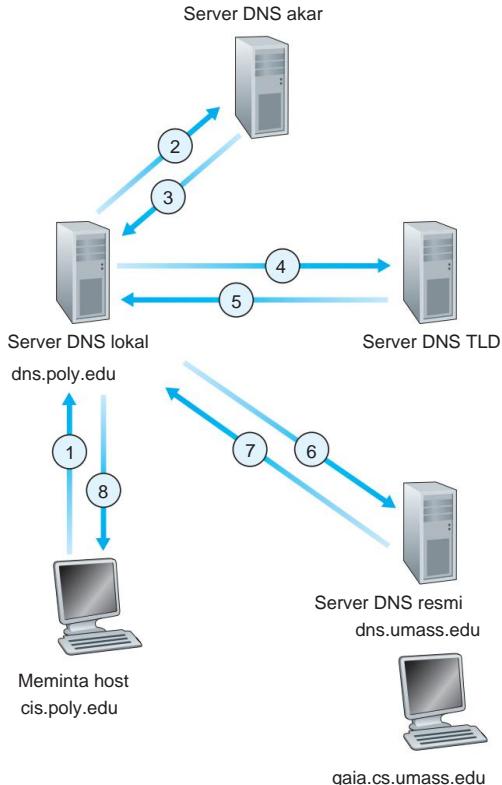
memilih untuk mengimplementasikan server DNS otoritatifnya sendiri untuk menyimpan catatan ini; mengubah aslinya, organisasi dapat membayar agar catatan ini disimpan di server DNS resmi dari beberapa penyedia layanan. Sebagian besar universitas dan perusahaan besar mengimplementasikan dan memelihara server DNS otoritatif primer dan sekunder (cadangan) mereka sendiri.

Root, TLD, dan server DNS otoritatif semuanya termasuk dalam hierarki server DNS, seperti yang ditunjukkan pada Gambar 2.19. Ada jenis server DNS penting lainnya yang disebut **server DNS lokal**. Server DNS lokal tidak sepenuhnya termasuk dalam hierarki server tetapi tetap merupakan pusat dari arsitektur DNS. Setiap ISP—seperti universitas, departemen akademik, perusahaan karyawan, atau ISP perumahan—memiliki server DNS lokal (juga disebut server nama default). Ketika host terhubung ke ISP, ISP menyediakan host dengan alamat IP dari satu atau lebih server DNS lokalnya (biasanya melalui DHCP, yang dibahas di Bab 4). Anda dapat dengan mudah menentukan alamat IP server DNS lokal Anda dengan mengakses jendela status jaringan di Windows atau UNIX. Server DNS lokal host biasanya "dekat dengan" host. Untuk ISP institusional, server DNS lokal mungkin berada di LAN yang sama dengan host; untuk ISP perumahan, biasanya dipisahkan dari host oleh tidak lebih dari beberapa router. Saat host membuat kueri DNS, kueri dikirim ke server DNS lokal, yang bertindak sebagai proxy, meneruskan kueri ke hierarki server DNS, seperti yang akan kita bahas lebih detail di bawah.

Mari kita lihat contoh sederhana. Misalkan host cis.poly.edu menginginkan alamat IP gaia.cs.umass.edu. Misalkan juga server DNS lokal Politeknik disebut dns.poly.edu dan server DNS otoritatif untuk gaia.cs.umass.edu disebut dns.umass.edu. Seperti yang ditunjukkan pada Gambar 2.21, host cis.poly.edu pertama-tama mengirimkan pesan permintaan DNS ke server DNS lokalnya, dns.poly.edu. Pesan kueri berisi hostname yang akan diterjemahkan, yaitu gaia.cs.umass.edu. Server DNS lokal meneruskan pesan kueri ke server DNS root. Server DNS root mencatat akhiran edu dan mengembalikan ke server DNS lokal daftar alamat IP untuk server TLD yang bertanggung jawab atas edu. Server DNS lokal kemudian mengirim ulang pesan kueri ke salah satu server TLD ini. Server TLD mencatat akhiran umass.edu dan merespons dengan alamat IP server DNS otoritatif untuk University of Massachusetts, yaitu, dns.umass.edu. Terakhir, server DNS lokal mengirim ulang pesan kueri langsung ke dns.umass.edu, yang merespons dengan alamat IP gaia.cs.umass.edu. Perhatikan bahwa dalam contoh ini, untuk memperoleh pemetaan untuk satu nama host, delapan pesan DNS dikirim: empat pesan kueri dan empat pesan balasan! Kami akan segera melihat bagaimana caching DNS mengurangi lalu lintas kueri ini.

Contoh kami sebelumnya mengasumsikan bahwa server TLD mengetahui server DNS otoritatif untuk nama host. Secara umum ini tidak selalu benar. Sebaliknya, server TLD mungkin hanya mengetahui server DNS perantara, yang pada gilirannya mengetahui server DNS otoritatif untuk nama host. Sebagai contoh, misalkan lagi bahwa Universitas

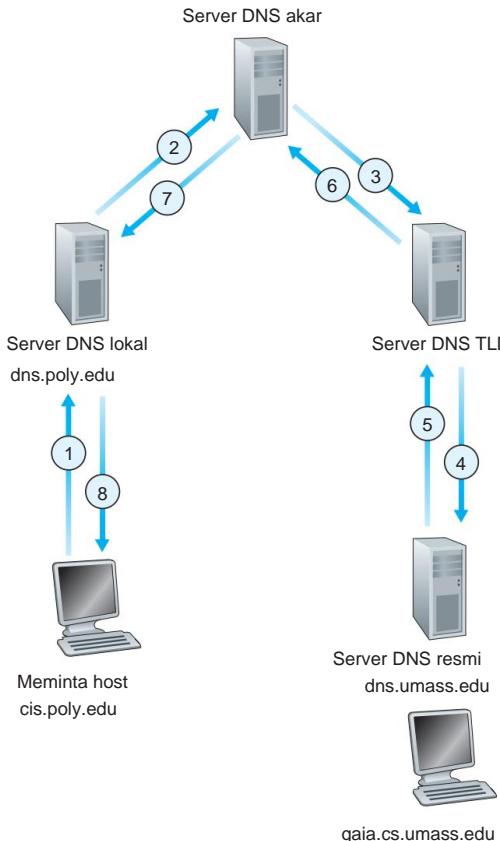
2.5 • DNS—LAYANAN DIREKTORI INTERNET 137

**Gambar 2.21** Interaksi berbagai server DNS

Massachusetts memiliki server DNS untuk universitas, disebut dns.umass.edu. Anggap juga bahwa setiap departemen di University of Massachusetts memiliki server DNS sendiri, dan bahwa setiap server DNS departemen berwenang untuk semua host di departemen. Dalam hal ini, ketika server DNS perantara, dns.umass.edu, menerima kueri untuk host dengan nama host yang diakhiri dengan cs.umass.edu, ia mengembalikan ke dns.poly.edu alamat IP dari dns.cs.umass.edu, yang berwenang untuk semua nama host yang diakhiri dengan cs.umass.edu. Server DNS lokal dns.poly.edu kemudian mengirimkan kueri ke server DNS otoritatif, yang mengembalikan pemetaan yang diinginkan ke server DNS lokal, yang kemudian mengembalikan ping pong peta ke host yang meminta. Dalam hal ini, total 10 pesan DNS dikirim!

Contoh yang ditunjukkan pada Gambar 2.21 memanfaatkan **kueri rekursif** dan **kueri iteratif**. Kueri yang dikirim dari cis.poly.edu ke dns.poly.edu adalah kueri rekursif, karena kueri tersebut meminta dns.poly.edu untuk mendapatkan pemetaan pada

138 BAB 2 • LAPISAN APLIKASI

**Gambar 2.22** Permintaan rekursif dalam DNS

kepentingan. Namun tiga kueri berikutnya bersifat iteratif karena semua balasan langsung dikembalikan ke dns.poly.edu. Secara teori, kueri DNS apa pun dapat bersifat iteratif atau rekursif. Sebagai contoh, Gambar 2.22 menunjukkan rangkaian kueri DNS yang semua kuerinya bersifat rekursif. Dalam praktiknya, kueri biasanya mengikuti pola pada Gambar 2.21: Permintaan dari host yang meminta ke server DNS lokal bersifat rekursif, dan kueri yang tersisa bersifat iteratif.

Caching DNS

Diskusi kami sejauh ini telah mengabaikan **caching DNS**, fitur yang sangat penting dari sistem DNS. Sebenarnya, DNS secara ekstensif mengeksplorasi caching DNS untuk meningkatkan kinerja penundaan dan untuk mengurangi jumlah pesan DNS yang memantul.

internet. Gagasan di balik caching DNS sangat sederhana. Dalam rantai kueri, saat server DNS menerima balasan DNS (berisi, misalnya, pemetaan dari nama host ke alamat IP), ia dapat meng-cache pemetaan di memori lokalnya. Misalnya, pada Gambar 2.21, setiap kali server DNS lokal dns.poly.edu menerima balasan dari beberapa server DNS, ia dapat meng-cache informasi apa pun yang terkandung dalam balasan tersebut. Jika pasangan nama host/alamat IP di-cache di server DNS dan kueri lain datang ke server DNS untuk nama host yang sama, server DNS dapat memberikan alamat IP yang diinginkan, meskipun tidak resmi untuk nama host. Karena host dan pemetaan antara nama host dan alamat IP sama sekali tidak permanen, server DNS membuang informasi yang di-cache setelah periode waktu tertentu (sering kali disetel ke dua hari).

Sebagai contoh, misalkan host apricot.poly.edu meminta alamat IP untuk hostname cnn.com ke dns.poly.edu. Selain itu, misalkan beberapa jam kemudian, host Universitas Politeknik lain, katakanlah, kiwi.poly.fr, juga menanyakan dns.poly.edu dengan nama host yang sama. Karena caching, server DNS lokal akan dapat segera mengembalikan alamat IP cnn.com ke host permintaan kedua ini tanpa harus meminta server DNS lainnya. Server DNS lokal juga dapat meng-cache alamat IP server TLD, sehingga memungkinkan server DNS lokal melewati server DNS root dalam rantai kueri (ini sering terjadi).

2.5.3 Catatan dan Pesan DNS

Server DNS yang bersama-sama mengimplementasikan **catatan sumber daya penyimpanan database terdistribusi DNS (RR)**, termasuk RR yang menyediakan ping peta alamat hostname-to-IP. Setiap pesan balasan DNS membawa satu atau beberapa catatan sumber daya. Dalam subbagian ini dan berikutnya, kami memberikan ikhtisar singkat tentang catatan dan pesan sumber daya DNS; detail lebih lanjut dapat ditemukan di [Abitz 1993] atau di DNS RFC [RFC 1034; RFC 1035].

Catatan sumber daya adalah empat tupel yang berisi bidang-bidang berikut:

(Nama, Nilai, Jenis, TTL)

TTL adalah waktu hidup dari catatan sumber daya; itu menentukan kapan sumber daya harus dihapus dari cache. Dalam catatan contoh yang diberikan di bawah ini, kami mengabaikan bidang TTL. Arti Nama dan Nilai tergantung pada Jenis:

- Jika Type=A, maka Name adalah nama host dan Value adalah alamat IP untuk nama host. Jadi, catatan Tipe A menyediakan ping peta nama host-ke-alamat IP standar. Sebagai contoh, (relay1.bar.foo.com, 145.37.93.126, A) adalah catatan Tipe A.
- Jika Type=NS, maka Name adalah domain (seperti foo.com) dan Value adalah nama host dari server DNS otoritatif yang mengetahui cara mendapatkan alamat IP untuk host di domain tersebut. Catatan ini digunakan untuk merutekan kueri DNS lebih jauh

140 BAB 2 • LAPISAN APLIKASI

rantai kueri. Sebagai contoh, (foo.com, dns.foo.com, NS) adalah catatan Tipe NS.

- Jika Type=CNAME, maka Value adalah nama host kanonis untuk Nama hostname alias. Data ini dapat memberikan host kueri nama kanonis untuk nama host. Sebagai contoh, (foo.com, relay1.bar.foo.com, CNAME) adalah data CNAME.
- Jika Type=MX, maka Value adalah nama kanonis dari server email yang memiliki alias Nama hostname. Sebagai contoh, (foo.com, mail.bar.foo.com, MX) adalah data MX. Catatan MX memungkinkan nama host server surat memiliki alias sederhana. Perhatikan bahwa dengan menggunakan data MX, sebuah perusahaan dapat memiliki nama samaran yang sama untuk server emailnya dan untuk salah satu server lainnya (seperti server Webnya). Untuk mendapatkan nama kanonis untuk server email, klien DNS akan meminta data MX; untuk mendapatkan nama kanonis untuk server lain, klien DNS akan meminta catatan CNAME.

Jika server DNS berwenang untuk nama host tertentu, maka server DNS akan berisi catatan Tipe A untuk nama host. (Bahkan jika server DNS tidak otoritatif, mungkin berisi catatan Tipe A di cache-nya.) Jika server tidak otoritatif untuk nama host, maka server akan berisi catatan Tipe NS untuk domain yang menyertakan nama host; itu juga akan berisi catatan Tipe A yang memberikan alamat IP dari server DNS di bidang Nilai dari catatan NS. Sebagai contoh, misalkan server TLD edu tidak otoritatif untuk host gaia.cs.umass.edu. Kemudian server ini akan berisi record untuk domain yang menyertakan host gaia.cs.umass.edu, misalnya (umass.edu, dns.umass.edu, NS). Server TLD edu juga akan berisi data Tipe A, yang memetakan server DNS dns.umass.edu ke alamat IP, misalnya, (dns.umass.edu, 128.119.40.111, A).

Pesan DNS

Sebelumnya di bagian ini, kami mengacu pada kueri DNS dan membahas pesan. Ini adalah hanya dua jenis pesan DNS. Selain itu, pesan query dan reply memiliki format yang sama, seperti yang ditunjukkan pada Gambar 2.23. Semantik dari berbagai field dalam pesan DNS adalah sebagai berikut:

- 12 byte pertama adalah *bagian header*, yang memiliki sejumlah field. Bidang pertama adalah angka 16-bit yang mengidentifikasi kueri. Pengidentifikasi ini disalin ke pesan balasan ke kueri, memungkinkan klien mencocokkan balasan yang diterima dengan kueri yang dikirim. Ada sejumlah bendera di bidang bendera. Bendera kueri/balasan 1-bit menunjukkan apakah pesan tersebut adalah kueri (0) atau balasan (1). Bendera otoritatif 1-bit diatur dalam pesan balasan saat server DNS adalah server otoritatif untuk nama yang diminta. Bendera yang diinginkan rekursi 1-bit disetel saat klien (host atau server DNS) menginginkan server DNS melakukan rekursi saat tidak memiliki catatan. Bidang tersedia rekursi 1-bit diatur dalam balasan jika server DNS mendukung rekursi. Di tajuk,

Identifikasi	Bendera	
Jumlah pertanyaan	Jumlah RR jawaban	12 byte
Jumlah RR otoritas	Jumlah RR tambahan	
Pertanyaan (jumlah variabel pertanyaan)		Nama, ketik bidang untuk kueri
Jawaban (jumlah variabel catatan sumber daya)		RR dalam menanggapi permintaan
Otoritas (jumlah variabel catatan sumber daya)		Catatan untuk server otoritatif
Informasi tambahan (jumlah variabel catatan sumber daya)		Info "bermanfaat" tambahan yang dapat digunakan

Gambar 2.23 Format pesan DNS

ada juga empat jumlah bidang. Bidang ini menunjukkan jumlah kejadian dari empat jenis bagian data yang mengikuti header.

- Bagian *soal* berisi informasi tentang query yang sedang dibuat.

Bagian ini mencakup (1) bidang nama yang berisi nama yang sedang ditanyakan, dan (2) bidang jenis yang menunjukkan jenis pertanyaan yang ditanyakan tentang nama—misalnya, alamat host yang terkait dengan nama (Tipe A) atau server email untuk sebuah nama (Tipe MX). • Dalam balasan dari server DNS, *bagian jawaban* berisi catatan sumber daya untuk nama yang awalnya ditanyakan. Ingatlah bahwa di setiap catatan sumber daya terdapat Jenis (misalnya, A, NS, CNAME, dan MX), Nilai, dan TTL.

Balasan dapat mengembalikan banyak RR dalam jawaban, karena nama host dapat memiliki banyak alamat IP (misalnya, untuk server Web yang direplikasi, seperti yang dibahas sebelumnya di bagian ini).

- *Bagian otoritas* berisi catatan dari server otoritatif lainnya.
- *Bagian tambahan* berisi catatan bermanfaat lainnya. Misalnya, kolom jawaban dalam balasan ke kueri MX berisi catatan sumber daya yang menyediakan nama host kanonis dari server email. Bagian tambahan berisi catatan Tipe A yang menyediakan alamat IP untuk nama host kanonis dari server email.

Bagaimana Anda ingin mengirim pesan kueri DNS langsung dari host yang sedang Anda kerjakan ke beberapa server DNS? Ini dapat dengan mudah dilakukan dengan **nslookup**

142 BAB 2 • LAPISAN APLIKASI

program, yang tersedia dari sebagian besar platform Windows dan UNIX. Misalnya, dari host Windows, buka Command Prompt dan aktifkan program nslookup hanya dengan mengetikkan "nslookup." Setelah menjalankan nslookup, Anda dapat mengirim kueri DNS ke server DNS apa pun (root, TLD, atau otoritatif). Setelah menerima pesan balasan dari server DNS, nslookup akan menampilkan catatan yang disertakan dalam balasan (dalam format yang dapat dibaca manusia). Sebagai alternatif untuk menjalankan nslookup dari host Anda sendiri, Anda dapat mengunjungi salah satu dari banyak situs Web yang memungkinkan Anda menggunakan nslookup dari jarak jauh. (Cukup ketik "nslookup" ke dalam mesin pencari dan Anda akan dibawa ke salah satu situs ini.) Lab DNS Wireshark di akhir bab ini akan memungkinkan Anda menjelajahi DNS dengan lebih detail.

Memasukkan Catatan ke dalam Database DNS

Pembahasan di atas berfokus pada bagaimana record diambil dari database DNS.

Anda mungkin bertanya-tanya bagaimana catatan masuk ke database sejak awal. Mari kita lihat bagaimana ini dilakukan dalam konteks contoh spesifik. Misalkan Anda baru saja membuat perusahaan rintisan baru yang menarik bernama Network Utopia. Hal pertama yang pasti ingin Anda lakukan adalah mendaftarkan nama domain networkutopia.com di registrar. Registrar adalah entitas komersial yang memverifikasi keunikan nama domain, memasukkan nama domain ke dalam database DNS (seperti dibahas di bawah), dan memungut sedikit biaya dari Anda untuk layanannya. Sebelum tahun 1999, satu registrar, Network Solutions, memonopoli pendaftaran nama domain untuk domain com, net, dan org. Tapi sekarang ada banyak registrar yang bersaing untuk mendapatkan pelanggan, dan Internet Corporation for Assigned Names and Numbers (ICANN) mengakreditasi berbagai registrar. Daftar lengkap registrar terakreditasi tersedia di <http://www.internic.net>.

Saat Anda mendaftarkan nama domain networkutopia.com dengan beberapa reg istrar, Anda juga perlu memberikan registrar nama dan alamat IP server DNS otoritatif primer dan sekunder Anda. Misalkan nama dan alamat IP adalah dns1.networkutopia.com, dns2.networkutopia.com, 212.212.212.1, dan 212.212.212.2. Untuk masing-masing dari dua server DNS otoritatif ini, pencatat kemudian akan memastikan bahwa catatan Tipe NS dan Tipe A dimasukkan ke server TLD com. Secara khusus, untuk server otoritatif utama untuk networkutopia.com, pencatat akan memasukkan dua catatan sumber daya berikut ke dalam sistem DNS:

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

(dns1.networkutopia.com, 212.212.212.1, A)

Anda juga harus memastikan bahwa catatan sumber daya Tipe A untuk server Web Anda www.networkutopia.com dan catatan sumber daya Tipe MX untuk server surat Anda mail.networkutopia.com dimasukkan ke dalam server DNS otoritatif Anda. (Sampai saat ini, isi setiap server DNS dikonfigurasikan secara statis,



FOKUS PADA KEAMANAN

KERENTANAN DNS

Kita telah melihat bahwa DNS adalah komponen penting dari infrastruktur Internet, dengan banyak layanan penting - termasuk Web dan email - tidak dapat berfungsi tanpanya. Karena itu kami secara alami bertanya, bagaimana DNS dapat diserang? Apakah DNS adalah bebek duduk, menunggu untuk dimatikan, sementara mengambil sebagian besar aplikasi Internet bersamanya?

Jenis serangan pertama yang terlintas dalam pikiran adalah serangan banjir bandwidth DDoS (lihat Bagian 1.6) terhadap server DNS. Sebagai contoh, seorang penyerang dapat mencoba untuk mengirim ke setiap server DNS root banjir paket, sehingga sebagian besar pertanyaan DNS yang sah tidak pernah dijawab. Serangan DDoS skala besar terhadap server akar DNS benar-benar terjadi pada 21 Oktober 2002. Dalam serangan ini, penyerang memanfaatkan jaringan bot untuk mengirimkan muatan truk pesan ping ICMP ke masing-masing dari 13 server akar DNS.

(Pesan ICMP dibahas di Bab 4. Untuk saat ini, cukup diketahui bahwa paket ICMP adalah tipe khusus dari datagram IP.) Untungnya, serangan berskala besar ini menyebabkan kerusakan minimal, berdampak kecil atau tidak sama sekali pada pengalaman Internet pengguna. Penyerang berhasil mengarahkan banjir paket di server root. Tetapi banyak dari server root DNS dilindungi oleh filter paket, dikonfigurasi untuk selalu memblokir semua pesan ping ICMP yang diarahkan ke server root. Server yang dilindungi ini terhindar dan berfungsi seperti biasa. Selain itu, sebagian besar server DNS lokal meng-cache alamat IP server domain tingkat atas, yang memungkinkan proses kueri untuk sering melewati server akar DNS.

Serangan DDoS yang berpotensi lebih efektif terhadap DNS akan mengirimkan banjir kueri DNS ke server domain tingkat atas, misalnya, ke semua server domain tingkat atas yang menangani domain .com. Akan lebih sulit untuk memfilter kueri DNS yang diarahkan ke server DNS; dan server domain tingkat atas tidak semudah dilewati seperti server root. Tetapi tingkat keparahan serangan semacam itu sebagian akan dikurangi dengan melakukan caching di server DNS lokal.

DNS berpotensi diserang dengan cara lain. Dalam serangan man-in-the-middle, penyerang mencegat kueri dari host dan mengembalikan balasan palsu. Dalam serangan DNS poisoning, penyerang mengirimkan balasan palsu ke server DNS, mengelabui server agar menerima catatan palsu ke dalam cache-nya. Salah satu dari serangan ini dapat digunakan, misalnya, untuk mengarahkan ulang pengguna Web yang tidak menaruh curiga ke situs Web penyerang. Serangan-serangan ini, bagaimanapun, sulit untuk diimplementasikan, karena membutuhkan paket-paket yang mencegat atau mencekik server [Skoudis 2006].

Serangan DNS penting lainnya bukanlah serangan pada layanan DNS itu sendiri, melainkan mengeksplorasi infrastruktur DNS untuk meluncurkan serangan DDoS terhadap host yang ditargetkan (misalnya, server email universitas Anda). Dalam serangan ini, penyerang mengirimkan kueri DNS ke banyak server DNS otoritatif, dengan setiap kueri memiliki alamat sumber palsu dari host yang ditargetkan. Server DNS kemudian mengirim balasan mereka langsung ke host target. Jika kueri dapat dibuat sedemikian rupa sehingga responsnya jauh lebih besar

144 BAB 2 • LAPISAN APLIKASI**FOKUS PADA KEAMANAN**

(dalam byte) daripada kueri (disebut amplifikasi), maka penyerang berpotensi melampaui target tanpa harus menghasilkan banyak lalu lintasnya sendiri. Serangan refleksi semacam itu yang mengeksplorasi DNS memiliki keberhasilan yang terbatas hingga saat ini [Mirkovic 2005].

Singkatnya, DNS telah menunjukkan dirinya sangat kuat terhadap serangan.

Hingga saat ini, belum ada serangan yang berhasil menggagalkan layanan DNS.

Ada serangan reflektor yang berhasil; namun, serangan ini dapat (dan sedang) diatasi dengan konfigurasi server DNS yang sesuai.

misalnya, dari file konfigurasi yang dibuat oleh manajer sistem. Baru-baru ini, opsi UPDATE telah ditambahkan ke protokol DNS untuk memungkinkan data ditambahkan atau dihapus secara dinamis dari database melalui pesan DNS. [RFC 2136] dan [RFC 3007] menentukan pembaruan dinamis DNS.)

Setelah semua langkah ini selesai, orang akan dapat mengunjungi situs Web Anda dan mengirim email ke karyawan di perusahaan Anda. Mari kita akhiri pembahasan kita tentang DNS dengan memverifikasi bahwa pernyataan ini benar. Verifikasi ini juga membantu memperkuat apa yang telah kita pelajari tentang DNS. Misalkan Alice di Australia ingin melihat halaman Web www.networkutopia.com. Seperti yang telah dibahas sebelumnya, host-nya pertama-tama akan mengirimkan kueri DNS ke server DNS lokalnya. Server DNS lokal kemudian akan menghubungi server TLD com. (Server DNS lokal juga harus menghubungi server DNS root jika alamat server com TLD tidak di-cache.) Server TLD ini berisi catatan sumber daya Tipe NS dan Tipe A yang tercantum di atas, karena pencatat telah memasukkan catatan sumber daya ini ke semua server com TLD. TLD com server mengirimkan balasan ke server DNS lokal Alice, dengan balasan berisi dua catatan sumber daya. Server DNS lokal kemudian mengirimkan kueri DNS ke 212.212.212.1, meminta catatan Tipe A yang terkait dengan www.networkutopia.com. Catatan ini memberikan alamat IP dari server Web yang diinginkan, katakanlah, 212.212.71.4, yang diteruskan oleh server DNS lokal ke host Alice. Browser Alice sekarang dapat memulai koneksi TCP ke host 212.212.71.4 dan mengirim permintaan HTTP melalui koneksi tersebut. Wah! Ada lebih banyak hal yang terjadi daripada yang terlihat ketika seseorang menjelajahi Web!

2.6 Aplikasi Peer-to-Peer

Aplikasi yang dijelaskan dalam bab ini sejauh ini—termasuk Web, email, dan DNS—semua menggunakan arsitektur server-klien dengan ketergantungan signifikan pada server infrastruktur yang selalu aktif. Ingat dari Bagian 2.1.1 bahwa dengan arsitektur P2P, ketergantungan minimal (atau tidak ada) pada server infrastruktur yang selalu aktif. Sebaliknya, pasangan host yang terhubung sebentar-sebentar, yang disebut peer, berkomunikasi langsung satu sama lain

Rekan tidak dimiliki oleh penyedia layanan, melainkan desktop dan laptop yang dikendalikan oleh pengguna.

Pada bagian ini kita akan memeriksa dua aplikasi berbeda yang sangat cocok untuk desain P2P. Yang pertama adalah distribusi file, di mana aplikasi mendistribusikan file dari satu sumber ke sejumlah besar peer. Distribusi file adalah tempat yang bagus untuk memulai penyelidikan kami tentang P2P, karena jelas memperlihatkan skalabilitas mandiri arsitektur P2P. Sebagai contoh spesifik untuk distribusi file, kami akan menjelaskan sistem BitTorrent yang populer. Aplikasi P2P kedua yang akan kita periksa adalah database yang didistribusikan melalui komunitas peer yang besar. Untuk aplikasi ini, kita akan mendalamai konsep Distributed Hash Table (DHT).

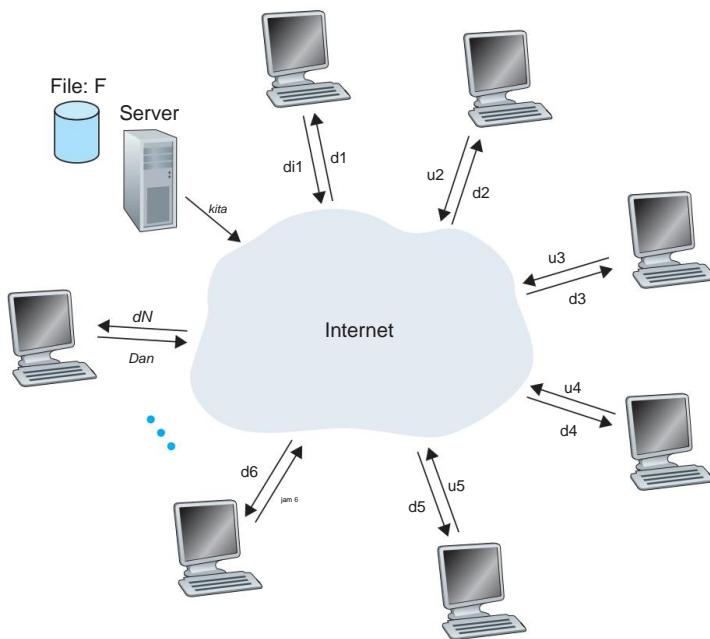
2.6.1 Distribusi File P2P

Kami memulai terjun ke P2P dengan mempertimbangkan aplikasi yang sangat alami, yaitu mendistribusikan file besar dari satu server ke sejumlah besar host (disebut peer). File tersebut mungkin versi baru sistem operasi Linux, tambalan perangkat lunak untuk sistem operasi atau aplikasi yang ada, file musik MP3, atau file video MPEG. Dalam distribusi file client-server, server harus mengirimkan salinan file ke masing-masing rekan — menempatkan beban yang sangat besar pada server dan menghabiskan banyak bandwidth server. Dalam distribusi file P2P, setiap peer dapat mendistribusikan ulang bagian mana pun dari file yang telah diterimanya ke peer lainnya, sehingga membantu server dalam proses distribusi. Pada 2012, protokol distribusi file P2P paling populer adalah BitTorrent. Awalnya dikembangkan oleh Bram Cohen, sekarang ada banyak klien BitTorrent independen berbeda yang sesuai dengan protokol BitTorrent, sama seperti ada sejumlah klien browser Web yang sesuai dengan protokol HTTP. Dalam sub-bagian ini, pertama-tama kita memeriksa skalabilitas mandiri arsitektur P2P dalam konteks distribusi file. Kami kemudian menjelaskan BitTorrent secara mendetail, menyoroti karakteristik dan fitur terpentingnya.

Skalabilitas Arsitektur P2P

Untuk membandingkan arsitektur klien-server dengan arsitektur peer-to-peer, dan mengilustrasikan skalabilitas P2P yang melekat, kami sekarang mempertimbangkan model kuantitatif sederhana untuk mendistribusikan file ke sekumpulan peer tetap untuk kedua jenis arsitektur. Seperti yang ditunjukkan pada Gambar 2.24, server dan peer terhubung ke Internet dengan link akses. Tunjukkan tingkat ungahan tautan akses server oleh $kami$, tingkat ungahan tautan akses rekan ke- i oleh ui , dan tingkat unduhan tautan akses rekan ke- i oleh di . Juga nyatakan ukuran file yang akan didistribusikan (dalam bit) oleh F dan jumlah peer yang ingin mendapatkan salinan file sebesar N . **Waktu distribusi** adalah waktu yang diperlukan untuk mendapatkan salinan file ke semua N rekan. Dalam analisis kami tentang waktu distribusi di bawah ini, untuk arsitektur klien-server dan P2P, kami membuat asumsi penyederhanaan (dan secara umum akurat [Akella 2003]) bahwa inti Internet memiliki banyak

146 BAB 2 • LAPISAN APLIKASI

**Gambar 2.24** Ilustrasi masalah distribusi file

bandwidth, menyiratkan bahwa semua hambatan ada di jaringan akses. Kami juga menganggap bahwa server dan klien tidak berpartisipasi dalam aplikasi jaringan lain, sehingga semua bandwidth akses upload dan download mereka dapat sepenuhnya dikhususkan untuk mendistribusikan file ini.

Pertama-tama mari kita tentukan waktu distribusi untuk arsitektur client-server, yang kita tandai dengan D_{cs} . Dalam arsitektur client-server, tidak ada peer yang membantu mendistribusikan file. Kami melakukan pengamatan berikut:

- Server harus mengirimkan satu salinan file ke masing-masing N peer. Jadi server harus mengirimkan bit NF . Karena kecepatan unggahan server waktu untuk dis adalah us , upeti file setidaknya harus NF/us . • Biarkan $dmin$ menunjukkan tingkat unduhan rekan dengan tingkat unduhan terendah, yaitu, $dmin = \min\{d_1, d_2, \dots, d_N\}$. Rekan dengan tingkat unduhan terendah tidak dapat memperoleh semua F bit file dalam waktu kurang dari $F/dmin$ detik. Dengan demikian waktu distribusi minimum setidaknya $F/dmin$.

Menyatukan kedua pengamatan ini, kami memperoleh

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

Ini memberikan batas bawah pada waktu distribusi minimum untuk arsitektur client-server. Dalam soal pekerjaan rumah, Anda akan diminta untuk menunjukkan bahwa server dapat menjadwalkan pengirimannya sehingga batas bawah benar-benar tercapai. Jadi mari kita ambil batas bawah yang diberikan di atas sebagai waktu distribusi sebenarnya, yaitu,

$$D_{CS} = \max \frac{F}{NF}, \frac{d_{min}}{r} \quad (2.1)$$

Kita lihat dari Persamaan 2.1 bahwa untuk N yang cukup besar, waktu distribusi client-server diberikan oleh NF/us . Dengan demikian, waktu distribusi meningkat secara linier dengan jumlah peer N . Jadi, misalkan, jika jumlah peer dari satu minggu ke minggu berikutnya meningkat seribu kali lipat dari seribu menjadi satu juta, waktu yang diperlukan untuk mendistribusikan file ke semua rekan bertambah 1.000.

Sekarang mari kita lihat analisis serupa untuk arsitektur P2P, di mana setiap peer dapat membantu server dalam mendistribusikan file. Secara khusus, ketika peer menerima beberapa data file, itu dapat menggunakan kapasitas uploadnya sendiri untuk mendistribusikan ulang data ke peer lainnya. Menghitung waktu distribusi untuk arsitektur P2P agak lebih rumit daripada arsitektur client-server, karena waktu distribusi bergantung pada bagaimana setiap peer mendistribusikan bagian file ke peer lainnya. Namun demikian, ekspresi sederhana untuk waktu distribusi minimal dapat diperoleh [Kumar 2006]. Untuk itu, pertama-tama kami melakukan pengamatan sebagai berikut:

- Pada awal pendistribusian, hanya server yang memiliki file tersebut. Untuk memasukkan file ini ke dalam komunitas peer, server harus mengirimkan setiap bit file setidaknya satu kali ke link aksesnya. Dengan demikian, waktu distribusi minimum setidaknya F/us . (Berbeda dengan skema client-server, bit yang dikirim sekali oleh server mungkin tidak perlu dikirim lagi oleh server, karena peer dapat mendistribusikan ulang bit di antara mereka sendiri.) • Seperti arsitektur client-server, peer dengan tingkat unduhan terendah tidak dapat memperoleh semua F bit file dalam waktu kurang dari F/d_{min} detik. Dengan demikian waktu distribusi minimum setidaknya F/d_{min} . • Terakhir, amati bahwa total kapasitas upload sistem secara keseluruhan sama dengan kecepatan upload server ditambah kecepatan upload masing-masing peer, yaitu, $utotal = us + u1 + + uN$. Sistem harus mengirimkan (mengunggah) bit bit yang mungkin dapat diunggah mengirimkan kecepatan lebih cepat dari $utotal$. Dengan demikian, waktu distribusi minimum juga paling sedikit $NF/(us + u1 + + uN)$.
- ⋮

Menyatukan ketiga pengamatan ini, kami memperoleh waktu distribusi minimum untuk P2P, dilambangkan dengan DP2P.

$$\text{DP2P} = \frac{F}{kita}, \frac{NF}{hubungi kami + a} \quad (2.2)$$

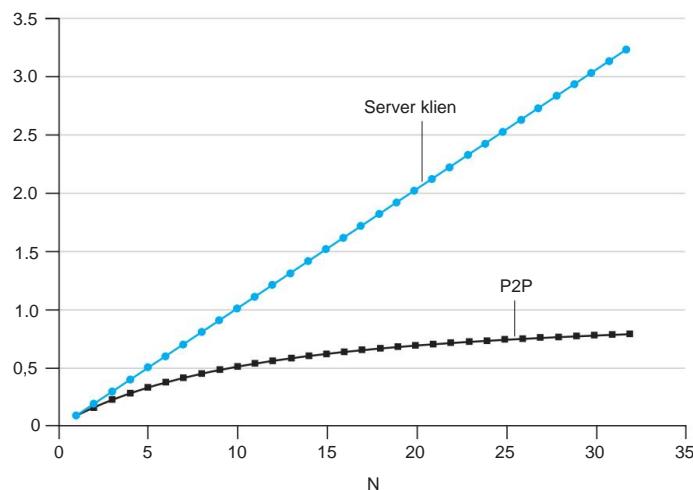
$\leq \max c_F$ saya=1 uis

148 BAB 2 • LAPISAN APLIKASI

Persamaan 2.2 memberikan batas bawah untuk waktu distribusi minimum untuk arsitektur P2P. Ternyata jika kita membayangkan bahwa setiap rekan dapat mendistribusikan kembali sedikit segera setelah menerima bit, maka ada skema redistribusi yang benar-benar mencapai batas bawah ini [Kumar 2006]. (Kami akan membuktikan kasus khusus dari hasil ini dalam pekerjaan rumah.) Pada kenyataannya, di mana potongan file didistribusikan ulang daripada bit individu, Persamaan 2.2 berfungsi sebagai perkiraan yang baik dari waktu distribusi minimum yang sebenarnya. Jadi, mari kita ambil batas bawah yang disediakan oleh Persamaan 2.2 sebagai waktu distribusi minimum yang sebenarnya, yaitu,

$$\text{DP2P} = \max c \frac{F}{\begin{array}{c} \text{kita} \\ \text{dmin} \end{array}}, \frac{NF}{\begin{array}{c} N \\ \text{kami} + a \\ \text{saya=1} \end{array}} \quad (2.3)$$

Gambar 2.25 membandingkan waktu distribusi minimum untuk arsitektur client-server dan P2P dengan asumsi bahwa semua peer memiliki tingkat upload u yang sama. Pada Gambar 2.25, kita telah menetapkan $F/u = 1$ jam, $us = 10u$, dan $dmin = us$. Dengan demikian, peer dapat mengirimkan seluruh file dalam satu jam, kecepatan transmisi server adalah 10 kali kecepatan upload peer, dan (untuk kesederhanaan) kecepatan download peer diatur cukup besar agar tidak berpengaruh. Kita lihat dari Gambar 2.25 bahwa untuk arsitektur client-server, waktu distribusi meningkat secara linear dan tanpa batas seiring bertambahnya jumlah peer. Namun, untuk arsitektur P2P, waktu distribusi minimal tidak selalu lebih kecil dari waktu distribusi arsitektur client-server; itu juga kurang dari satu jam untuk *sejumlah* peer N . Dengan demikian, aplikasi dengan arsitektur P2P dapat diskalakan sendiri. Skalabilitas ini merupakan konsekuensi langsung dari peer menjadi redistributor serta konsumen bit.



Gambar 2.25 Waktu distribusi untuk arsitektur P2P dan client-server

BitTorrent

BitTorrent adalah protokol P2P populer untuk distribusi file [Chao 2011]. Dalam bahasa sewa BitTor, kumpulan semua rekan yang berpartisipasi dalam distribusi file tertentu disebut torrent . Peer dalam torrent mengunduh *potongan* file berukuran sama dari satu sama lain, dengan ukuran potongan tipikal 256 KBytes. Saat peer pertama kali bergabung dengan torrent, ia tidak memiliki potongan. Seiring waktu, ia mengumpulkan lebih banyak bongkahan. Saat mengunduh potongan, ia juga mengunggah potongan ke rekan lain. Setelah rekan mendapatkan seluruh file, mungkin (secara egois) meninggalkan torrent, atau (secara altruistik) tetap berada di torrent dan terus mengunggah potongan ke rekan lain. Juga, rekan mana pun dapat meninggalkan torrent kapan saja hanya dengan sebagian dari potongan, dan kemudian bergabung kembali dengan torrent.

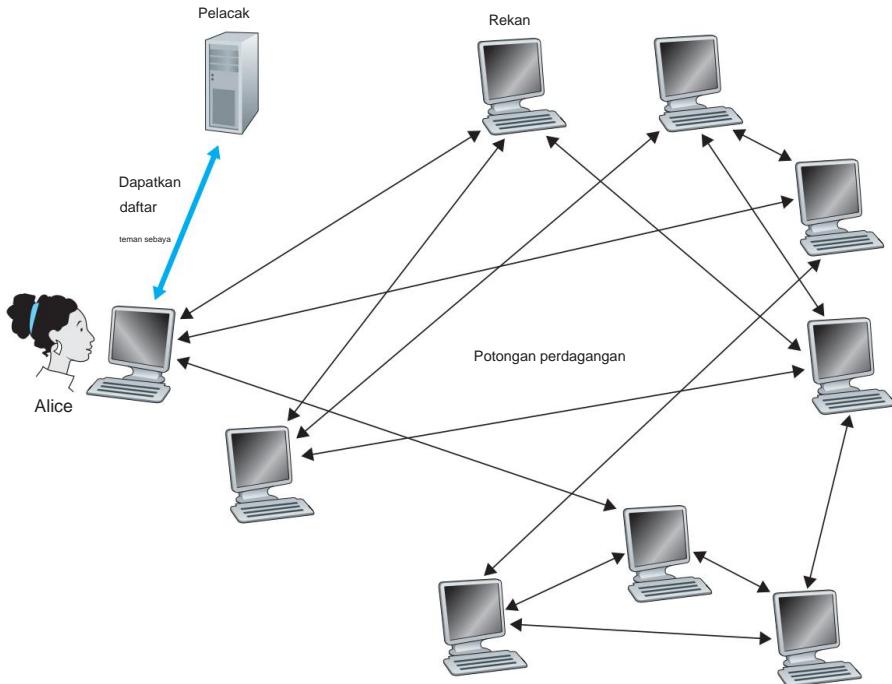
Sekarang mari kita lihat lebih dekat bagaimana BitTorrent beroperasi. Karena BitTorrent adalah protokol dan sistem yang agak rumit, kami hanya akan menjelaskan mekanismenya yang paling penting, menyapu beberapa detail di bawah permadani; ini akan memungkinkan kita untuk melihat hutan melalui pepohonan. Setiap torrent memiliki simpul infrastruktur yang disebut *pelacak*. Ketika rekan bergabung dengan torrent, ia mendaftarkan diri ke pelacak dan secara berkala memberi tahu pelacak bahwa itu masih ada di dalam torrent. Dengan cara ini, pelacak melacak rekan-rekan yang berpartisipasi dalam torrent. Torrent tertentu mungkin memiliki kurang dari sepuluh atau lebih dari seribu rekan yang berpartisipasi kapan saja.

Seperti yang ditunjukkan pada Gambar 2.26, ketika rekan baru, Alice, bergabung dengan torrent, pelacak secara acak memilih subkumpulan rekan (agar konkret, katakanlah 50) dari kumpulan rekan yang berpartisipasi, dan mengirimkan alamat IP dari 50 rekan ini ke Alice. Dengan memiliki daftar peer ini, Alice mencoba membuat koneksi TCP bersamaan dengan semua peer di daftar ini. Sebut saja semua peer yang dengannya Alice berhasil membangun koneksi TCP sebagai "peer tetangga". (Pada Gambar 2.26, Alice diperlihatkan hanya memiliki tiga peer yang bertetangga. Biasanya, dia akan memiliki lebih banyak lagi.) Seiring berjalannya waktu, beberapa dari peer ini dapat pergi dan peer lainnya (di luar 50 awal) mungkin mencoba membuat koneksi TCP dengan Alice. Jadi, peer tetangga peer akan berfluktuasi dari waktu ke waktu.

Pada waktu tertentu, setiap peer akan memiliki subset potongan dari file, dengan peer yang berbeda memiliki subset yang berbeda. Secara berkala, Alice akan meminta setiap peer tetangganya (melalui koneksi TCP) untuk daftar potongan yang mereka miliki. Jika Alice memiliki L tetangga yang berbeda, dia akan mendapatkan L daftar potongan. Dengan pengetahuan ini, Alice akan mengeluarkan permintaan (sekali lagi melalui koneksi TCP) untuk potongan yang saat ini tidak dia miliki.

Jadi pada saat tertentu, Alice akan memiliki subset potongan dan akan mengetahui potongan mana yang dimiliki tetangganya. Dengan informasi ini, Alice harus mengambil dua keputusan penting. Pertama, potongan mana yang harus dia minta terlebih dahulu dari tetangganya? Dan kedua, ke tetangga mana dia harus mengirimkan potongan yang diminta? Dalam memutuskan potongan mana yang akan diminta, Alice menggunakan teknik yang disebut **paling langka terlebih dahulu**. Idenya adalah untuk menentukan, dari antara potongan yang tidak dia miliki, potongan yang paling langka di antara tetangganya (yaitu, potongan yang memiliki salinan berulang paling sedikit di antara tetangganya) dan kemudian meminta potongan paling langka tersebut terlebih dahulu. Dengan cara ini, bongkahan yang paling langka didistribusikan kembali dengan lebih cepat, bertujuan untuk (secara kasar) menyamakan jumlah salinan dari setiap bongkahan di torrent.

150 BAB 2 • LAPISAN APLIKASI

**Gambar 2.26** Distribusi file dengan BitTorrent

Untuk menentukan permintaan mana yang dia tanggapi, BitTorrent menggunakan algoritme perdagangan yang cerdas. Ide dasarnya adalah bahwa Alice memberikan prioritas kepada tetangga yang saat ini memasok datanya *dengan kecepatan tertinggi*. Khususnya, untuk setiap tetangganya, Alice terus-menerus mengukur tingkat di mana dia menerima bit dan menentukan empat rekan yang memberi makan bitnya pada tingkat tertinggi. Dia kemudian membalas dengan mengirimkan potongan ke empat rekan yang sama ini. Setiap 10 detik, dia menghitung ulang tarif dan kemungkinan memodifikasi set empat rekan. Dalam istilah BitTorrent, keempat rekan ini dikatakan **tidak tercekik**. Yang penting, setiap 30 detik, dia juga memilih satu tetangga tambahan secara acak dan mengirimkannya potongan. Sebut saja peer Bob yang dipilih secara acak. Dalam istilah persewaan BitTor, Bob dikatakan secara **optimis tidak tercekik**. Karena Alice mengirimkan data ke Bob, dia mungkin menjadi salah satu dari empat pengunggah Bob, dalam hal ini Bob akan mulai mengirim data ke Alice. Jika kecepatan Bob mengirimkan data ke Alice cukup tinggi, Bob kemudian dapat menjadi salah satu dari empat pengunggah teratas Alice. Dengan kata lain, setiap 30 detik, Alice akan secara acak memilih mitra dagang baru dan mulai perdagangan dengan mitra tersebut. Jika kedua rekan puas dengan perdagangan, mereka akan menempatkan satu sama lain di daftar empat teratas mereka dan melanjutkan perdagangan satu sama lain sampai salah satu rekan menemukan pasangan yang lebih baik. Efeknya adalah peer yang mampu mengunggah dengan kecepatan yang sesuai cenderung saling menemukan. Pemilihan tetangga secara acak juga memungkinkan peer baru untuk mendapat-

potongan, sehingga mereka dapat memiliki sesuatu untuk diperdagangkan. Semua peer tetangga selain lima peer ini (empat peer "atas" dan satu peer probing) "tersedak", artinya, mereka tidak menerima bongkahan apa pun dari Alice. BitTorrent memiliki sejumlah mekanisme menarik yang tidak dibahas di sini, termasuk potongan (mini-chunks), pipelining, random first selection, endgame mode, dan anti-snubbing [Cohen 2003].

Mekanisme insentif untuk perdagangan yang baru saja dijelaskan sering disebut sebagai tit-for-tat [Cohen 2003]. Telah ditunjukkan bahwa skema insentif ini dapat dilakukan [Liogkas 2006; Locher 2006; Piatek 2007]. Namun demikian, ekosistem BitTorrent sangat sukses, dengan jutaan rekan serempak secara aktif berbagi file dalam ratusan ribu torrent. Jika BitTorrent telah dirancang tanpa tit-for-tat (atau varian), tetapi sebaliknya persis sama, BitTorrent kemungkinan besar tidak akan ada sekarang, karena mayoritas penggunanya adalah freerider [Saroui 2002].

Varian menarik dari protokol BitTorrent diusulkan [Guo 2005; Piatek 2007]. Selain itu, banyak aplikasi streaming langsung P2P, seperti PPLive dan ppstream, terinspirasi oleh BitTorrent [Hei 2007].

2.6.2 Tabel Hash Terdistribusi (DHT)

Pada bagian ini, kami akan mempertimbangkan bagaimana mengimplementasikan database sederhana di jaringan P2P. Mari kita mulai dengan menjelaskan versi terpusat dari basis data sederhana ini, yang hanya berisi pasangan (kunci, nilai). Misalnya, kuncinya bisa berupa nomor jaminan sosial dan nilainya bisa berupa nama manusia yang sesuai; dalam hal ini, contoh key-value pair adalah (156-45-7081, Johnny Wu). Atau kuncinya bisa berupa nama konten (misalnya, nama film, album, dan perangkat lunak), dan nilainya bisa berupa alamat IP tempat konten disimpan; dalam hal ini, contoh pasangan kunci-nilai adalah (Led Zeppelin IV, 128.17.123.38). Kami meminta database dengan kunci. Jika ada satu atau beberapa pasangan kunci-nilai dalam database yang cocok dengan kunci kueri, database mengembalikan nilai yang sesuai. Jadi, misalnya, jika database menyimpan nomor jaminan sosial dan nama manusia yang sesuai, kita dapat melakukan kueri dengan nomor jaminan sosial tertentu, dan database mengembalikan nama manusia yang memiliki nomor jaminan sosial tersebut. Atau, jika database menyimpan nama konten dan alamat IP yang sesuai, kita dapat melakukan kueri dengan nama konten tertentu, dan database mengembalikan alamat IP yang menyimpan konten tertentu.

Membangun database seperti itu sangat mudah dengan arsitektur client-server yang menyimpan semua pasangan (kunci, nilai) dalam satu server pusat. Jadi di bagian ini, kami akan mempertimbangkan cara membuat versi P2P terdistribusi dari database ini yang akan menyimpan pasangan (kunci, nilai) di jutaan peer. Dalam sistem P2P, setiap peer hanya akan menyimpan sebagian kecil dari totalitas pasangan (kunci, nilai). Kami akan mengizinkan rekan mana pun untuk menanyakan basis data terdistribusi dengan kunci tertentu. Basis data terdistribusi kemudian akan menemukan peer yang memiliki pasangan (kunci, nilai) yang sesuai dan mengembalikan pasangan nilai kunci ke peer kueri. Rekan mana pun juga akan diizinkan untuk memasukkan pasangan kunci-nilai baru ke dalam database. Database terdistribusi seperti itu disebut sebagai **tabel hash terdistribusi (DHT)**.

152 BAB 2 • LAPISAN APLIKASI



VideoNote

Berjalan melalui
tabel hash yang didistribusikan

Sebelum menjelaskan cara membuat DHT, pertama-tama mari kita jelaskan contoh spesifik layanan DHT dalam konteks berbagi file P2P. Dalam hal ini, kunci adalah nama konten dan nilainya adalah alamat IP rekan yang memiliki salinan konten.

Jadi, jika Bob dan Charlie masing-masing memiliki salinan distribusi Linux terbaru, maka basis data DHT akan menyertakan dua pasangan nilai kunci berikut: (Linux, IPBob) dan (Linux, IPCharlie). Lebih khusus lagi, karena basis data DHT didistribusikan melalui peer, beberapa peer, katakanlah Dave, akan bertanggung jawab atas kunci "Linux" dan akan memiliki pasangan nilai kunci yang sesuai. Sekarang misalkan Alice ingin mendapatkan salinan Linux. Jelas, pertama-tama dia perlu mengetahui rekan mana yang memiliki salinan Linux sebelum dia dapat mulai mengunduhnya. Untuk tujuan ini, dia menanyakan DHT dengan "Linux" sebagai kuncinya.

DHT kemudian menentukan bahwa rekan Dave bertanggung jawab atas kunci "Linux." DHT kemudian menghubungi rekan Dave, memperoleh dari Dave pasangan nilai kunci (Linux, IPBob) dan (Linux, IPCharlie), dan meneruskannya ke Alice. Alice kemudian dapat mengunduh distribusi Linux terbaru dari IPBob atau IPCharlie.

Sekarang mari kita kembali ke masalah umum merancang DHT untuk pasangan nilai kunci umum. Salah satu pendekatan naif untuk membangun DHT adalah dengan menyebarkan pasangan (kunci, nilai) secara acak ke semua peer dan meminta setiap peer memelihara daftar alamat IP dari semua peer yang berpartisipasi. Dalam desain ini, rekan yang melakukan kueri mengirimkan kuerinya ke semua rekan lainnya, dan rekan yang berisi pasangan (kunci, nilai) yang cocok dengan kunci dapat merespons dengan pasangan yang cocok. Pendekatan seperti itu benar-benar tidak dapat diskalakan, tentu saja, karena akan mengharuskan setiap rekan untuk tidak hanya mengetahui tentang semua rekan lainnya (mungkin jutaan rekan seperti itu!) Tetapi lebih buruk lagi, meminta setiap kueri dikirim ke semua *rekan*.

Kami sekarang menjelaskan pendekatan yang elegan untuk merancang DHT. Untuk tujuan ini, pertama-tama mari kita tetapkan pengenal untuk setiap peer, di mana setiap pengenal adalah bilangan bulat dalam rentang $[0, 2n 1]$ untuk beberapa n tetap. Perhatikan bahwa setiap pengidentifikasi tersebut dapat diekspresikan oleh representasi n-bit. Mari kita juga mengharuskan setiap kunci menjadi bilangan bulat dalam rentang yang sama. Pembaca yang cerdik mungkin telah mengamati bahwa contoh kunci yang dijelaskan sebelumnya (nomor jaminan sosial dan nama konten) bukanlah bilangan bulat. Untuk membuat bilangan bulat dari kunci tersebut, kita akan menggunakan fungsi hash yang memetakan setiap kunci (misalnya, nomor jaminan sosial) ke bilangan bulat dalam kisaran $[0, 2n 1]$. Fungsi hash adalah fungsi banyak-ke-satu di mana dua input berbeda dapat memiliki output yang sama (integer yang sama), tetapi kemungkinan memiliki output yang sama sangat kecil. (Pembaca yang tidak terbiasa dengan fungsi hash mungkin ingin mengunjungi Bab 7, di mana fungsi hash dibahas secara mendetail.) Fungsi hash diasumsikan tersedia untuk semua peer dalam sistem. Selanjutnya, ketika kita mengacu pada "kunci", kita mengacu pada hash dari kunci aslinya. Jadi, misalnya, jika kunci aslinya adalah "Led Zeppelin IV", kunci yang digunakan dalam DHT adalah bilangan bulat yang sama dengan hash "Led Zeppelin IV". Seperti yang mungkin sudah Anda duga, inilah mengapa "Hash" digunakan dalam istilah "Fungsi Hash Terdistribusi".

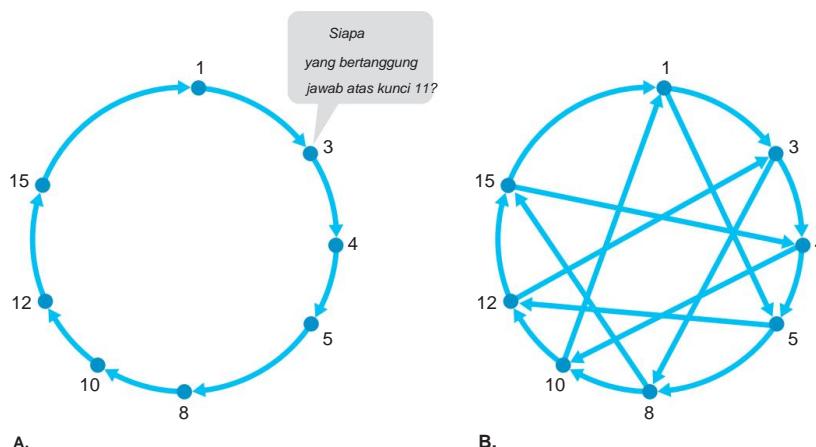
Sekarang mari kita pertimbangkan masalah menyimpan pasangan (kunci, nilai) di DHT. Masalah utama di sini adalah menentukan aturan untuk menetapkan kunci ke peer. Mengingat bahwa setiap rekan memiliki pengidentifikasi bilangan bulat dan bahwa setiap kunci juga merupakan bilangan bulat dalam rentang yang sama, pendekatan alami adalah dengan menetapkan setiap pasangan (kunci, nilai) ke rekan yang pengidentifikasinya paling dekat dengan *kunci*. Untuk menerapkan skema seperti itu, kita perlu mendefinisikan apa yang dimaksud dengan "terdekat", yang memungkinkan banyak konvensi. Untuk kenyamanan, mari kita tentukan

peer terdekat sebagai *penerus kunci terdekat*. Untuk mendapatkan beberapa wawasan di sini, mari kita lihat contoh spesifik. Misalkan $n = 4$ sehingga semua pengidentifikasi peer dan key berada dalam rentang $[0, 15]$. Lebih lanjut misalkan ada delapan peer dalam sistem dengan pengidentifikasi 1, 3, 4, 5, 8, 10, 12, dan 15. Terakhir, misalkan kita ingin menyimpan pasangan (kunci, nilai) (11, Johnny Wu) di salah satu dari delapan rekan. Tapi di rekan yang mana? Dengan menggunakan konvensi terdekat kita, karena rekan 12 adalah penerus terdekat untuk kunci 11, oleh karena itu kita menyimpan pasangan (11, Johnny Wu) di rekan 12. [Untuk melengkapi definisi terdekat kita, jika kuncinya persis sama dengan salah satu dari pengidentifikasi rekan, kami menyimpan pasangan (kunci, nilai) di rekan yang cocok itu; dan jika kuncinya lebih besar dari semua pengidentifikasi peer, kami menggunakan konvensi modulo-2n, menyimpan pasangan (kunci, nilai) di peer dengan pengidentifikasi terkecil.]

Sekarang misalkan peer, Alice, ingin memasukkan pasangan (kunci, nilai) ke dalam DHT. Secara konseptual, ini mudah: Dia pertama-tama menentukan rekan yang pengenalnya paling dekat dengan kunci; dia kemudian mengirim pesan ke rekan itu, menginstruskannya untuk menyimpan pasangan (kunci, nilai). Tapi bagaimana cara Alice menentukan rekan yang paling dekat dengan kunci? Jika Alice melacak semua peer dalam sistem (ID peer dan alamat IP yang sesuai), dia dapat menentukan peer terdekat secara lokal. Namun pendekatan semacam itu mengharuskan *setiap* rekan untuk melacak *semua* rekan lainnya di DHT—yang sama sekali tidak praktis untuk sistem skala besar dengan jutaan rekan.

DHT Edaran

Untuk mengatasi masalah skala ini, mari sekarang pertimbangkan untuk mengatur rekan-rekan ke dalam lingkaran. Dalam pengaturan melingkar ini, setiap rekan hanya melacak penerus langsung dan pendahulunya (modulo $2n$). Contoh lingkaran seperti itu ditunjukkan pada Gambar 2.27(a). Dalam contoh ini, n lagi 4 dan ada delapan yang sama



Gambar 2.27 (a) DHT berbentuk lingkaran. Rekan 3 ingin menentukan siapa yang bertanggung jawab atas kunci 11. (b) DHT melingkar dengan jalur pintas

154 BAB 2 • LAPISAN APLIKASI

rekan dari contoh sebelumnya. Setiap rekan hanya menyadari keberhasilan langsung dan pendahulunya; misalnya, rekan 5 mengetahui alamat IP dan pengidentifikasi untuk rekan 8 dan 4 tetapi belum tentu tahu apa pun tentang rekan lain yang mungkin ada di DHT. Susunan peer yang melingkar ini merupakan kasus khusus dari **jaringan overlay**. Dalam jaringan overlay, peer membentuk jaringan logika abstrak yang berada di atas jaringan komputer "underlay" yang terdiri dari link fisik, router, dan host. Tautan dalam jaringan overlay bukanlah tautan fisik, tetapi hanyalah penghubung virtual antara pasangan rekan. Pada overlay pada Gambar 2.27(a), terdapat delapan peer dan delapan link overlay; pada overlay pada Gambar 2.27(b) terdapat delapan peer dan 16 link overlay. Tautan overlay tunggal biasanya menggunakan banyak tautan fisik dan router fisik di jaringan underlay.

Dengan menggunakan overlay melingkar pada Gambar 2.27(a), sekarang misalkan peer 3 ingin menentukan peer mana di DHT yang bertanggung jawab atas kunci 11. Dengan menggunakan overlay melingkar, peer asal (peer 3) membuat pesan yang mengatakan "Siapa yang bertanggung jawab untuk kunci 11?" dan mengirimkan pesan ini searah jarum jam di sekitar lingkaran. Setiap kali rekan menerima pesan bijak seperti itu, karena mengetahui pengidentifikasi penerus dan pendahulunya, ia dapat menentukan apakah ia bertanggung jawab (yaitu, paling dekat dengan) kunci yang dimaksud. Jika rekan tidak bertanggung jawab atas kunci, itu hanya mengirimkan pesan ke penggantinya. Jadi, misalnya, ketika rekan 4 menerima pesan yang menanyakan tentang kunci 11, ia menentukan bahwa ia tidak bertanggung jawab atas kunci tersebut (karena penggantinya lebih dekat dengan kunci), sehingga hanya meneruskan pesan ke rekan 5. Proses ini berlanjut hingga pesan tiba di peer 12, yang menentukan bahwa itu adalah peer terdekat ke kunci 11. Pada titik ini, peer 12 dapat mengirim pesan kembali ke peer kueri, peer 3, yang menunjukkan bahwa itu bertanggung jawab atas kunci.

DHT melingkar memberikan solusi yang sangat elegan untuk mengurangi jumlah informasi overlay yang harus dikelola oleh setiap peer. Secara khusus, setiap peer hanya perlu mengetahui dua peer, penerus langsungnya dan pendahulunya. Tapi solusi ini menimbulkan masalah baru. Meskipun setiap peer hanya mengetahui dua peer yang bertetangga, untuk menemukan node yang bertanggung jawab atas kunci (dalam kasus terburuk), semua N node di DHT harus meneruskan pesan di sekitar lingkaran; Rata-rata $N/2$ pesan dikirim.

Jadi, dalam mendesain DHT, ada pertukaran antara jumlah tetangga yang harus dilacak oleh setiap peer dan jumlah pesan yang perlu dikirim DHT untuk menyelesaikan satu kueri. Di satu sisi, jika setiap peer melacak semua peer lainnya (mesh overlay), maka hanya satu pesan yang dikirim per kueri, tetapi setiap peer harus melacak N peer. Di sisi lain, dengan DHT melingkar, setiap rekan hanya mengetahui dua rekan, tetapi $N/2$ pesan dikirim rata-rata untuk setiap kueri. Untungnya, kami dapat menyempurnakan desain DHT kami sehingga jumlah tetangga per peer serta jumlah pesan per kueri disimpan ke ukuran yang dapat diterima. Salah satu penyempurnaan tersebut adalah dengan menggunakan hamparan melingkar sebagai fondasi, tetapi menambahkan "pintasan" sehingga setiap peer tidak hanya melacak penerus langsung dan pendahulunya, tetapi juga sejumlah kecil peer pintasan yang tersebar di sekitar lingkaran. Sebuah contoh dari DHT melingkar dengan beberapa jalur pintasan ditunjukkan pada Gambar 2.27(b). Pintasan digunakan untuk mempercepat perutean pesan kueri.

Secara khusus, ketika peer menerima pesan yang meminta kunci, itu meneruskan

pesan ke tetangga (tetangga penerus atau salah satu tetangga pintasan) yang merupakan lemari kunci. Jadi, pada Gambar 2.27(b), ketika peer 4 menerima pesan yang menanyakan tentang kunci 11, ia menentukan bahwa peer closet ke kunci (di antara tetangganya) adalah shortcut tetangga 10 dan kemudian meneruskan pesan langsung ke peer 10. Jelas, pintasan dapat secara signifikan mengurangi jumlah pesan yang digunakan untuk memproses kueri.

Pertanyaan alami berikutnya adalah "Berapa banyak pintasan yang harus dimiliki peer, dan peer mana yang harus menjadi tetangga pintasan ini? Pertanyaan ini mendapat perhatian yang signifikan dalam komunitas penelitian [Balakrishnan 2003; Androulidakis Theotokis 2004]. Yang penting, telah ditunjukkan bahwa DHT dapat dirancang sehingga baik jumlah tetangga per peer maupun jumlah pesan per query adalah $O(\log N)$, di mana N adalah jumlah peer. Desain seperti itu menghasilkan kompromi yang memuaskan antara solusi ekstrem menggunakan topologi mesh dan overlay melingkar.

Peer Churn

Dalam sistem P2P, peer bisa datang atau pergi tanpa peringatan. Jadi, saat mendesain DHT, kita juga harus memperhatikan pemeliharaan overlay DHT di hadapan churn rekan tersebut. Untuk mendapatkan pemahaman gambaran besar tentang bagaimana hal ini dapat dicapai, mari kita sekali lagi mempertimbangkan DHT sirkular pada Gambar 2.27(a). Untuk menangani churn peer, kami sekarang akan meminta setiap peer untuk melacak (yaitu, mengetahui alamat IP) penerus pertama dan kedua; misalnya, peer 4 sekarang melacak peer 5 dan peer 8. Kami juga wajibkan setiap peer untuk memverifikasi secara berkala bahwa dua penerusnya masih hidup (misalnya, dengan mengirimkan pesan ping secara berkala kepada mereka dan meminta tanggapan). Sekarang mari kita perhatikan bagaimana DHT dipertahankan ketika rekan tiba-tiba pergi. Misalnya, rekan 5 pada Gambar 2.27(a) tiba-tiba pergi. Dalam hal ini, dua rekan sebelum rekan yang berangkat (4 dan 3) mengetahui bahwa 5 telah berangkat, karena tidak lagi menanggapi pesan ping. Oleh karena itu, peer 4 dan 3 perlu memperbarui informasi status penerus mereka. Mari pertimbangkan bagaimana peer 4 memperbarui statusnya:

1. Peer 4 mengantik penerus pertamanya (peer 5) dengan penerus keduanya (peer 8).
2. Peer 4 kemudian meminta penerus pertama yang baru (peer 8) untuk pengidentifikasi dan alamat IP penerus langsungnya (peer 10). Peer 4 kemudian menjadikan peer 10 sebagai penerus keduanya.

Dalam masalah pekerjaan rumah, Anda akan diminta untuk menentukan bagaimana rekan 3 memperbarui informasi perutuan overlaynya.

Setelah membahas secara singkat apa yang harus dilakukan ketika seorang rekan pergi, sekarang mari kita pertimbangkan apa yang terjadi ketika seorang rekan ingin bergabung dengan DHT. Katakanlah peer dengan pengenal 13 ingin bergabung dengan DHT, dan pada saat bergabung, ia hanya mengetahui keberadaan peer 1 di DHT. Rekan 13 pertama-tama akan mengirim pesan kepada rekan 1, mengatakan "apa yang akan menjadi pendahulu dan penerus 13?" Pesan ini diteruskan melalui DHT hingga mencapai rekan 12, yang menyadari bahwa itu akan menjadi penerus 13 dan penerusnya saat ini, rekan 15, akan menjadi penerus 13. Selanjutnya, rekan 12 mengirimkan informasi pendahulu dan penerus ini ke rekan 13. Peer 13 sekarang dapat bergabung

156 BAB 2 • LAPISAN APLIKASI

DHT dengan menjadikan rekan 15 penggantinya dan dengan memberi tahu rekan 12 bahwa ia harus mengubah penerus langsungnya menjadi 13.

DHT telah menemukan penggunaan luas dalam praktiknya. Misalnya, BitTorrent menggunakan Kademia DHT untuk membuat pelacak terdistribusi. Di BitTorrent, kuncinya adalah pengidentifikasi torrent dan nilainya adalah alamat IP semua peer yang saat ini berpartisipasi dalam torrent [Falkner 2007, Neglia 2007]. Dengan cara ini, dengan menanyakan DHT dengan pengidentifikasi torrent, peer BitTorrent yang baru tiba dapat menentukan peer yang bertanggung jawab atas pengidentifikasi (yaitu, untuk melacak peer dalam sewa tor). Setelah menemukan rekan itu, rekan yang datang dapat menanyakannya untuk daftar rekan lain di torrent.

2.7 Pemrograman Soket: Membuat Aplikasi Jaringan

Sekarang setelah kita melihat sejumlah aplikasi jaringan yang penting, mari kita telusuri bagaimana sebenarnya program aplikasi jaringan dibuat. Ingat dari Bagian 2.1 bahwa aplikasi jaringan biasanya terdiri dari sepasang program—program klien dan program server—yang berada di dua sistem akhir yang berbeda. Saat kedua program ini dijalankan, proses klien dan proses server dibuat, dan proses ini berkomunikasi satu sama lain dengan membaca dari, dan menulis ke soket. Saat membuat aplikasi jaringan, tugas utama pengembang adalah menulis kode untuk program klien dan server.

Ada dua jenis aplikasi jaringan. Satu jenis adalah implementasi yang operasinya ditentukan dalam standar protokol, seperti RFC atau beberapa dokumen standar lainnya; aplikasi semacam itu terkadang disebut sebagai "terbuka", karena aturan yang menentukan operasinya diketahui semua orang. Untuk implementasi seperti itu, program klien dan server harus sesuai dengan aturan yang ditentukan oleh RFC. Sebagai contoh, program klien bisa menjadi implementasi dari sisi klien dari protokol FTP, dijelaskan dalam Bagian 2.3 dan secara eksplisit didefinisikan dalam RFC 959; sama halnya, program server bisa menjadi implementasi protokol server FTP, juga secara eksplisit didefinisikan dalam RFC 959. Jika satu pengembang menulis kode untuk program klien dan pengembang lain menulis kode untuk program server, dan kedua pengembang dengan hati-hati mengikuti aturan dari RFC, maka kedua program tersebut akan dapat saling beroperasi. Memang, banyak aplikasi jaringan saat ini melibatkan komunikasi antara program klien dan server yang telah dibuat oleh pengembang independen—misalnya, browser Firefox yang berkomunikasi dengan server Web Apache, atau klien BitTorrent yang berkomunikasi dengan pelacak BitTorrent.

Jenis lain dari aplikasi jaringan adalah aplikasi jaringan berpemilik.

Dalam hal ini program klien dan server menggunakan protokol lapisan aplikasi yang belum *dipublikasikan* secara terbuka di RFC atau di tempat lain. Satu pengembang (atau

2.7 • PEMROGRAMAN SOCKET: MEMBUAT APLIKASI JARINGAN 157

tim pengembangan) membuat program klien dan server, dan pengembang memiliki kendali penuh atas apa yang ada dalam kode. Namun karena kode tersebut tidak mengimplementasikan protokol terbuka, pengembang independen lainnya tidak akan dapat mengembangkan kode yang saling beroperasi dengan aplikasi tersebut.

Pada bagian ini, kita akan memeriksa isu-isu utama dalam mengembangkan aplikasi client-server, dan kita akan "mengotori tangan kita" dengan melihat kode yang mengimplementasikan aplikasi client-server yang sangat sederhana. Selama fase pengembangan, salah satu keputusan pertama yang harus dibuat oleh pengembang adalah apakah aplikasi dijalankan melalui TCP atau UDP. Ingatlah bahwa TCP berorientasi pada koneksi dan menyediakan saluran aliran byte yang andal melalui mana data mengalir di antara dua sistem akhir. UDP tidak terhubung dan mengirimkan paket data independen dari satu sistem ujung ke ujung lainnya, tanpa jaminan apa pun tentang pengiriman. Ingat juga bahwa ketika program klien atau server mengimplementasikan protokol yang ditentukan oleh RFC, itu harus menggunakan nomor port terkenal yang terkait dengan protokol; sebaliknya, saat mengembangkan aplikasi milik sendiri, pengembang harus berhati-hati untuk menghindari penggunaan nomor port yang terkenal tersebut. (Nomor port dibahas secara singkat di Bagian 2.1. Dibahas lebih detail di Bab 3.)

Kami memperkenalkan pemrograman soket UDP dan TCP melalui aplikasi UDP sederhana dan aplikasi TCP sederhana. Kami menyajikan aplikasi UDP dan TCP sederhana dengan Python. Kami dapat menulis kode dalam Java, C, atau C++, tetapi kami memilih Python terutama karena Python dengan jelas memaparkan konsep soket kunci. Dengan Python ada lebih sedikit baris kode, dan setiap baris dapat dijelaskan kepada programmer pemula tanpa kesulitan. Tapi tidak perlu takut jika Anda tidak terbiasa dengan Python. Anda harus dapat dengan mudah mengikuti kode jika Anda memiliki pengalaman pemrograman di Java, C, atau C++.

Jika Anda tertarik dengan pemrograman client-server dengan Java, Anda dianjurkan untuk melihat situs Web pendamping untuk buku teks ini; sebenarnya, Anda dapat menemukan semua contoh di bagian ini (dan lab terkait) di Java. Untuk pembaca yang tertarik dengan pemrograman client-server di C, ada beberapa referensi bagus yang tersedia [Donahoo 2001; Stevens 1997; Frost 1994; Kurose 1996]; contoh Python kami di bawah ini memiliki tampilan dan nuansa yang mirip dengan C.

2.7.1 Pemrograman Soket dengan UDP

Pada subbagian ini, kita akan menulis program client-server sederhana yang menggunakan UDP; di bagian berikut, kami akan menulis program serupa yang menggunakan TCP.

Ingat dari Bagian 2.1 bahwa proses yang berjalan pada mesin yang berbeda berkomunikasi satu sama lain dengan mengirimkan pesan ke soket. Kami mengatakan bahwa setiap proses dianalogikan dengan sebuah rumah dan soket proses dianalogikan dengan sebuah pintu. Aplikasi berada di satu sisi pintu di dalam rumah; protokol transport-layer berada di sisi lain pintu di dunia luar. Pengembang aplikasi memiliki kendali atas semua yang ada di sisi lapisan aplikasi soket; namun, ia memiliki sedikit kendali atas sisi lapisan transpor.

158 BAB 2 • LAPISAN APLIKASI

Sekarang mari kita lihat lebih dekat interaksi antara dua proses komunikasi yang menggunakan soket UDP. Sebelum proses pengiriman dapat mendorong paket data keluar dari pintu soket, saat menggunakan UDP, terlebih dahulu harus melampirkan alamat tujuan ke paket tersebut. Setelah paket melewati soket pengirim, Internet akan menggunakan alamat tujuan ini untuk merutekan paket melalui Internet ke soket dalam proses penerimaan. Ketika paket tiba di soket penerima, proses penerima akan mengambil paket melalui soket, dan kemudian memeriksa isi paket dan mengambil tindakan yang sesuai.

Jadi Anda mungkin sekarang bertanya-tanya, apa yang masuk ke alamat tujuan yang dilampirkan pada paket? Seperti yang Anda duga, alamat IP host tujuan adalah bagian dari alamat tujuan. Dengan memasukkan alamat IP tujuan dalam paket, router di Internet akan dapat merutekan paket melalui Internet ke host negara tujuan. Tetapi karena sebuah host dapat menjalankan banyak proses aplikasi jaringan, masing-masing dengan satu atau lebih soket, juga perlu untuk mengidentifikasi soket tertentu di host tujuan. Saat soket dibuat, pengidentifikasi, yang disebut **nomor port**, ditugaskan padanya. Jadi, seperti yang Anda duga, alamat tujuan paket juga menyertakan nomor port soket. Singkatnya, proses pengiriman melampirkan ke paket alamat tujuan yang terdiri dari alamat IP host tujuan dan nomor port soket tujuan. Selain itu, seperti yang akan segera kita lihat, alamat sumber pengirim — terdiri dari alamat IP dari host sumber dan nomor port dari soket sumber — juga dilampirkan ke paket. Namun, melampirkan alamat sumber ke paket biasanya *tidak* dilakukan oleh kode aplikasi UDP; alih-alih itu dilakukan secara otomatis oleh sistem operasi yang mendasarinya.

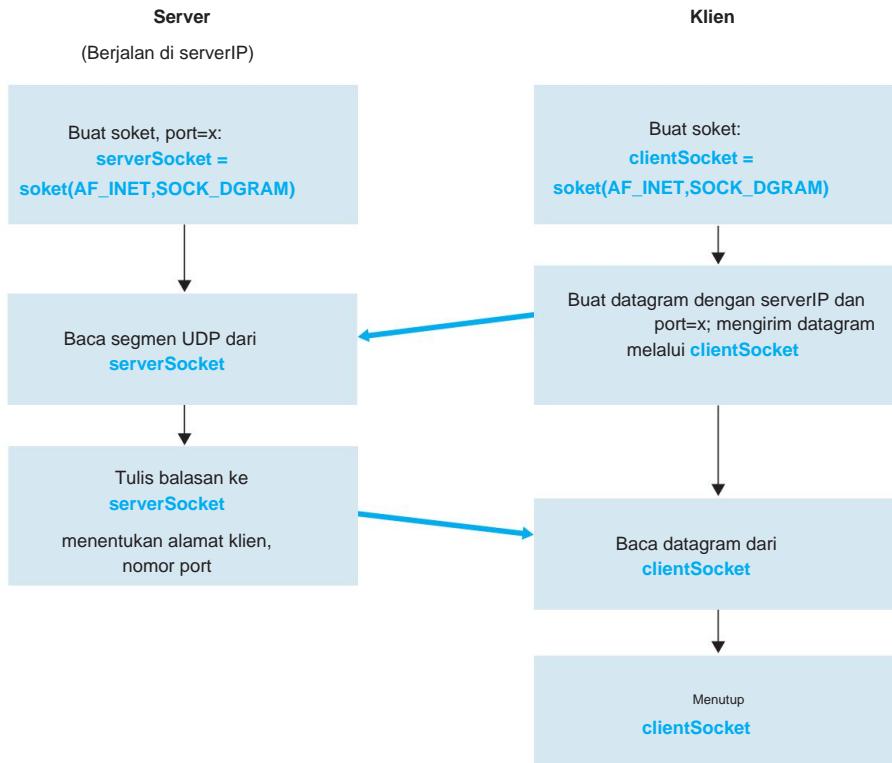
Kami akan menggunakan aplikasi client-server sederhana berikut untuk mendemonstrasikan socket pemrograman untuk UDP dan TCP:

1. Klien membaca sebaris karakter (data) dari keyboardnya dan mengirimkan data tersebut ke server.
2. Server menerima data dan mengubah karakter menjadi huruf besar.
3. Server mengirimkan data yang dimodifikasi ke klien.
4. Klien menerima data yang dimodifikasi dan menampilkan garis di layarnya.

Gambar 2.28 menyoroti aktivitas klien dan server terkait soket utama yang berkomunikasi melalui layanan transportasi UDP.

Sekarang mari kita tangani dan lihat pasangan program client-server untuk implementasi UDP dari aplikasi sederhana ini. Kami juga memberikan analisis terperinci, baris demi baris setelah setiap program. Kita akan mulai dengan klien UDP, yang akan mengirimkan pesan level aplikasi sederhana ke server. Agar server dapat menerima dan membalas pesan klien, server harus siap dan berjalan—yaitu, harus berjalan sebagai proses sebelum klien mengirimkan pesannya.

Program klien disebut `UDPClient.py`, dan program server disebut `UDPServer.py`. Untuk menekankan isu-isu utama, kami sengaja memberikan kode yang minimal. "Kode bagus" pasti akan memiliki beberapa baris tambahan, di



Gambar 2.28 Aplikasi client-server menggunakan UDP

khusus untuk menangani kasus error. Untuk aplikasi ini, kami telah memilih 12000 secara sewenang-wenang untuk nomor port server.

UDPClient.py

Berikut adalah kode untuk sisi klien aplikasi:

```

dari soket impor *
serverName = 'nama host'
serverPort = 12000
clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM)
pesan = raw_input('Masukkan kalimat huruf kecil:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
cetak pesan yang dimodifikasi
clientSocket.close()

```

160 BAB 2 • LAPISAN APLIKASI

Sekarang mari kita lihat berbagai baris kode di UDPClient.py.

```
dari impor soket *
```

Modul soket membentuk dasar dari semua komunikasi jaringan dengan Python. Dengan menyertakan baris ini, kita akan dapat membuat soket di dalam program kita.

```
serverName = 'nama host' serverPort  
= 12000
```

Baris pertama menetapkan string namaserver menjadi nama host. Di sini, kami menyediakan string yang berisi alamat IP server (misalnya, "128.138.32.126") atau nama host server (misalnya, "cis.poly.edu"). Jika kita menggunakan nama host, maka pencarian DNS akan dilakukan secara otomatis untuk mendapatkan alamat IP.) Baris kedua menetapkan serverPort variabel integer ke 12000.

```
clientSocket = soket(soket.AF_INET, soket.SOCK_DGRAM)
```

Baris ini membuat soket klien, disebut clientSocket. Parameter pertama menunjukkan keluarga alamat; khususnya, AF_INET menunjukkan bahwa jaringan yang mendasarinya menggunakan IPv4. (Jangan khawatir tentang hal ini sekarang—kami akan membahas IPv4 di Bab 4.) Parameter kedua menunjukkan bahwa soket bertipe SOCK_DGRAM, yang berarti soket UDP (bukan soket TCP). Perhatikan bahwa kami tidak menentukan nomor port soket klien saat kami membuatnya; kami malah membiarkan sistem operasi melakukan ini untuk kami. Sekarang pintu proses klien telah dibuat, kami ingin membuat pesan untuk dikirim melalui pintu.

```
pesan = raw_input('Masukkan kalimat dengan huruf kecil:')
```

raw_input() adalah fungsi bawaan di Python. Saat perintah ini dijalankan, pengguna di klien diminta dengan kata-kata "Masukkan data:" Pengguna kemudian menggunakan keyboardnya untuk memasukkan baris, yang dimasukkan ke dalam pesan variabel. Sekarang kami memiliki soket dan pesan, kami ingin mengirim pesan melalui soket ke host tujuan.

```
clientSocket.sendto(pesan,(serverName, serverPort))
```

Pada baris di atas, metode sendto() melampirkan alamat tujuan (serverName, serverPort) ke pesan dan mengirimkan paket yang dihasilkan ke dalam soket proses, clientSocket. (Seperti disebutkan sebelumnya, alamat sumber juga dilampirkan ke paket, meskipun ini dilakukan secara otomatis daripada secara eksplisit oleh kode.) Mengirim pesan klien-ke-server melalui soket UDP sesederhana itu!

Setelah mengirim paket, klien menunggu untuk menerima data dari server.

```
modifiedMessage, alamatserver = clientSocket.recvfrom(2048)
```

Dengan baris di atas, ketika sebuah paket datang dari Internet ke soket klien, data paket dimasukkan ke dalam variabel modifiedMessage dan alamat sumber paket dimasukkan ke dalam variabel serverAddress. Variabel serverAddress berisi alamat IP server dan nomor port server. Program UDPClient sebenarnya tidak memerlukan informasi alamat server ini, karena sudah mengetahui alamat server sejak awal; tetapi baris Python ini tetap menyediakan alamat server. Metode recvfrom juga mengambil ukuran buffer 2048 sebagai masukan. (Ukuran buffer ini berfungsi untuk sebagian besar tujuan.)

cetak pesan yang dimodifikasi

Baris ini mencetak ModifiedMessage pada tampilan pengguna. Itu harus menjadi baris asli yang diketik pengguna, tetapi sekarang dikapitalisasi.

```
clientSocket.close()
```

Baris ini menutup soket. Proses kemudian berakhir.

UDPServer.py

Sekarang mari kita lihat sisi server aplikasi:

```
dari soket impor * serverPort  
= 12000 serverSocket =  
  
socket(AF_INET, SOCK_DGRAM) serverSocket.bind((","  
serverPort)) cetak "Server siap menerima" sementara 1: pesan,  
clientAddress = serverSocket.recvfrom(2048) modifiedMessage  
= message.upper() serverSocket.sendto(modifiedMessage,  
clientAddress)
```

Perhatikan bahwa awal UDPServer mirip dengan UDPClient. Ia juga mengimpor modul soket, juga menyetel variabel integer serverPort ke 12000, dan juga membuat soket bertipe SOCK_DGRAM (soket UDP). Baris kode pertama yang sangat berbeda dari UDPClient adalah:

```
serverSocket.bind(("," serverPort))
```

Baris di atas mengikat (yaitu, menetapkan) nomor port 12000 ke soket server. Jadi di UDPServer, kode (ditulis oleh pengembang aplikasi) secara eksplisit

162 BAB 2 • LAPISAN APLIKASI

menetapkan nomor port ke soket. Dengan cara ini, ketika seseorang mengirim paket ke port 12000 di alamat IP server, paket itu akan diarahkan ke soket ini. UDPServer kemudian memasuki loop sementara; while loop akan memungkinkan UDPServer untuk menerima dan memproses paket dari klien tanpa batas. Di while loop, UDPServer menunggu paket tiba.

```
pesan, clientAddress = serverSocket.recvfrom(2048)
```

Baris kode ini mirip dengan yang kita lihat di UDPClient. Ketika sebuah paket tiba di soket server, data paket dimasukkan ke dalam pesan variabel dan alamat sumber paket dimasukkan ke dalam variabel clientAddress. Variabel clientAddress berisi alamat IP klien dan nomor port klien.

Di sini, UDPServer *akan* menggunakan informasi alamat ini, karena menyediakan alamat pengirim, mirip dengan alamat pengirim dengan surat pos biasa. Dengan informasi alamat sumber ini, server sekarang mengetahui ke mana ia harus mengarahkan balasannya.

```
modifiedMessage = pesan.upper()
```

Baris ini adalah inti dari aplikasi sederhana kami. Dibutuhkan baris yang dikirim oleh klien dan menggunakan metode upper() untuk mengkapitalisasinya.

```
serverSocket.sendto(modifiedMessage, clientAddress)
```

Baris terakhir ini melampirkan alamat klien (alamat IP dan nomor port) ke pesan yang dikapitalisasi, dan mengirimkan paket yang dihasilkan ke soket server. (Seperti disebutkan sebelumnya, alamat server juga dilampirkan ke paket, meskipun ini dilakukan secara otomatis daripada secara eksplisit oleh kode.) Internet kemudian akan mengirimkan paket ke alamat klien ini. Setelah server mengirim paket, paket tersebut tetap berada di while loop, menunggu paket UDP lain tiba (dari klien mana pun yang berjalan di host mana pun).

Untuk menguji pasangan program, Anda menginstal dan mengkompilasi UDPClient.py di satu host dan UDPServer.py di host lain. Pastikan untuk menyertakan nama host atau alamat IP yang tepat dari server di UDPClient.py. Selanjutnya, Anda menjalankan UDPServer.py, program server terkompilasi, di host server. Ini menciptakan proses di server yang menganggur hingga dihubungi oleh beberapa klien. Kemudian Anda menjalankan UDPClient.py, program klien terkompilasi, di klien. Ini menciptakan proses di klien. Terakhir, untuk menggunakan aplikasi di klien, Anda mengetik kalimat diikuti dengan carriage return.

Untuk mengembangkan aplikasi server-klien UDP Anda sendiri, Anda dapat memulai dengan sedikit memodifikasi program klien atau server. Misalnya, alih-alih mengonversi semua huruf menjadi huruf besar, server dapat menghitung berapa kali huruf s muncul dan mengembalikan angka ini. Atau Anda dapat memodifikasi klien sehingga setelah menerima kalimat yang dikapitalisasi, pengguna dapat terus mengirim lebih banyak kalimat ke server.

2.7.2 Pemrograman Soket dengan TCP

Tidak seperti UDP, TCP adalah protokol berorientasi koneksi. Ini berarti bahwa sebelum klien dan server dapat mulai mengirim data satu sama lain, mereka harus terlebih dahulu berjabat tangan dan membuat koneksi TCP. Salah satu ujung koneksi TCP terpasang ke soket klien dan ujung lainnya terpasang ke soket server. Saat membuat koneksi TCP, kami mengaitkannya dengan alamat soket klien (alamat IP dan nomor port) dan alamat soket server (alamat IP dan nomor port). Dengan koneksi TCP dibuat, ketika satu sisi ingin mengirim data ke sisi lain, itu hanya memasukkan data ke dalam koneksi TCP melalui soketnya. Ini berbeda dari UDP, di mana server harus melampirkan alamat tujuan ke paket sebelum memasukkannya ke dalam soket.

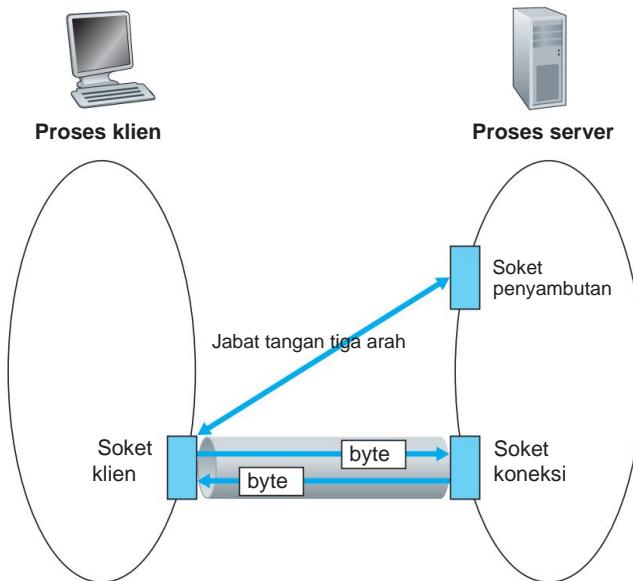
Sekarang mari kita lihat lebih dekat interaksi program klien dan server di TCP. Klien memiliki tugas memulai kontak dengan server. Agar server dapat bereaksi terhadap kontak awal klien, server harus siap.

Ini menyiratkan dua hal. Pertama, seperti dalam kasus UDP, server TCP harus dijalankan sebagai proses sebelum klien mencoba memulai kontak. Kedua, program server harus memiliki pintu khusus—lebih tepatnya, soket khusus—yang datang dari beberapa kontak awal dari proses klien yang berjalan pada host arbitrer. Dengan menggunakan analogi rumah/pintu kita untuk sebuah proses/soket, kita kadang-kadang menyebut kontak awal klien sebagai "mengetuk pintu penyambutan".

Dengan proses server berjalan, proses klien dapat memulai koneksi TCP ke server. Ini dilakukan dalam program klien dengan membuat soket TCP. Saat klien membuat soket TCP-nya, klien menentukan alamat soket penyambutan di server, yaitu alamat IP host server dan nomor port soket. Setelah membuat soketnya, klien memulai jabat tangan tiga arah dan membuat koneksi TCP dengan server. Jabat tangan tiga arah, yang terjadi di dalam lapisan transport, sama sekali tidak terlihat oleh program klien dan server.

Selama jabat tangan tiga arah, proses klien mengetuk pintu penyambutan dari proses server. Ketika server “mendengar” ketukan, itu membuat pintu baru — lebih tepatnya, soket baru yang didedikasikan untuk klien tertentu. Dalam contoh kami di bawah ini, pintu penyambutan adalah objek soket TCP yang kami sebut `serverSocket`; soket yang baru dibuat yang didedikasikan untuk klien yang membuat koneksi disebut soket koneksi. Siswa yang menemukan soket TCP untuk pertama kalinya beberapa kali membingungkan soket penyambutan (yang merupakan titik awal kontak untuk semua klien yang ingin berkomunikasi dengan server), dan setiap soket koneksi sisi server yang baru dibuat yang kemudian dibuat untuk berkomunikasi dengan setiap klien.

Dari perspektif aplikasi, soket klien dan soket koneksi server terhubung langsung dengan sebuah pipa. Seperti yang ditunjukkan pada Gambar 2.29, proses klien dapat mengirim sembarang byte ke dalam soketnya, dan TCP menjamin bahwa proses server akan menerima (melalui soket koneksi) setiap byte dalam urutan yang dikirim. TCP dengan demikian menyediakan layanan yang andal antara proses klien dan server. Selain itu, seperti halnya orang dapat masuk dan keluar dari pintu yang sama, proses klien tidak hanya mengirimkan byte



Gambar 2.29 Proses TCPServer memiliki dua soket

ke tetapi juga menerima byte dari soketnya; sama halnya, proses server tidak hanya menerima byte dari tetapi juga mengirimkan byte ke soket koneksinya.

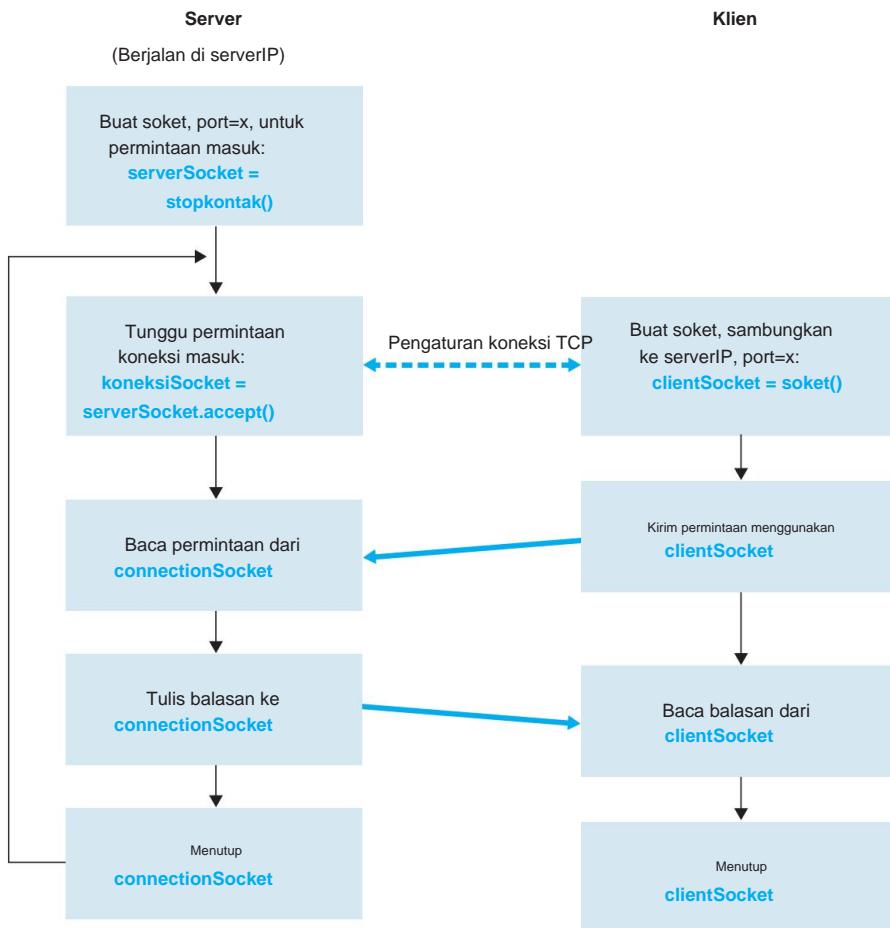
Kami menggunakan aplikasi klien-server sederhana yang sama untuk mendemonstrasikan penggabungan program soket dengan TCP: Klien mengirim satu baris data ke server, server mengkapitalisasi baris dan mengirimkannya kembali ke klien. Gambar 2.30 menyoroti aktivitas klien dan server terkait soket utama yang berkomunikasi melalui layanan transportasi TCP.

TCPClient.py

Berikut adalah kode untuk sisi klien aplikasi:

```
dari soket impor * namaserver
= 'namaserver'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort)) kalimat =
raw_input('Masukkan kalimat huruf kecil:') clientSocket.send(kalimat)
modifiedSentence = clientSocket.recv(1024) print 'Dari Server :',
modifiedSentence clientSocket.close()
```

2.7 • PEMROGRAMAN SOCKET: MEMBUAT APLIKASI JARINGAN 165



Gambar 2.30 Aplikasi client-server menggunakan TCP

Sekarang mari kita lihat berbagai baris kode yang berbeda secara signifikan dari implementasi UDP. Baris pertama adalah pembuatan soket klien.

`clientSocket = soket(AF_INET, SOCK_STREAM)`

Baris ini membuat soket klien, disebut `clientSocket`. Parameter pertama sekali lagi menunjukkan bahwa jaringan yang mendasarinya menggunakan IPv4. Parameter kedua menunjukkan bahwa soket bertipe `SOCK_STREAM`, yang berarti soket TCP (bukan soket UDP). Perhatikan bahwa kami sekali lagi tidak menentukan nomor port

166 BAB 2 • LAPISAN APLIKASI

soket klien saat kami membuatnya; kami malah membiarkan sistem operasi melakukan ini untuk kami. Sekarang baris kode berikutnya sangat berbeda dari yang kita lihat di UDPClient:

```
clientSocket.connect((serverName,serverPort))
```

Ingatlah bahwa sebelum klien dapat mengirim data ke server (atau sebaliknya) menggunakan soket TCP, koneksi TCP harus dibuat terlebih dahulu antara klien dan server. Baris di atas memulai koneksi TCP antara klien dan server. Parameter metode connect() adalah alamat sisi server dari koneksi. Setelah baris kode ini dieksekusi, jabat tangan tiga arah dilakukan dan koneksi TCP dibuat antara klien dan server.

```
kalimat = raw_input('Masukkan kalimat dengan huruf kecil:')
```

Seperti UDPClient, di atas mendapatkan kalimat dari pengguna. Kalimat string terus mengumpulkan karakter hingga pengguna mengakhiri baris dengan mengetikkan carriage return. Baris kode berikutnya juga sangat berbeda dengan UDPClient:

```
clientSocket.send(kalimat)
```

Baris di atas mengirimkan kalimat string melalui soket klien dan masuk ke koneksi TCP. Perhatikan bahwa program tidak secara eksplisit membuat paket dan melampirkan alamat tujuan ke paket, seperti yang terjadi pada soket UDP. Alih-alih, program klien hanya memasukkan byte dalam kalimat string ke dalam koneksi TCP. Klien kemudian menunggu untuk menerima byte dari server.

```
kalimat yang dimodifikasi = clientSocket.recv(2048)
```

Ketika karakter tiba dari server, mereka ditempatkan ke dalam kalimat yang dimodifikasi string. Karakter terus terakumulasi di modifiedSentence hingga baris diakhiri dengan karakter carriage return. Setelah mencetak kalimat dengan huruf besar, kami menutup soket klien:

```
clientSocket.close()
```

Baris terakhir ini menutup soket dan, karenanya, menutup koneksi TCP antara klien dan server. Ini menyebabkan TCP di klien mengirim pesan TCP ke TCP di server (lihat Bagian 3.5).

TCPServer.py

Sekarang mari kita lihat program server.

2.7 • PEMROGRAMAN SOCKET: MEMBUAT APLIKASI JARINGAN 167

```
dari impor soket * serverPort  
= 12000  
serverSocket = socket(AF_INET,SOCK_STREAM)  
serverSocket.bind((","+serverPort)) serverSocket.listen(1) print  
'Server siap menerima' sementara 1:
```

```
connectionSocket, addr = serverSocket.accept() kalimat =  
connectionSocket.recv(1024) capitalizedSentence = kalimat.upper()  
connectionSocket.send(capitalizedSentence) connectionSocket.close()
```

Sekarang mari kita lihat garis yang berbeda secara signifikan dari UDPServer dan Klien TCP. Seperti TCPClient, server membuat soket TCP dengan:

```
serverSocket=socket(AF_INET,SOCK_STREAM)
```

Mirip dengan UDPServer, kami mengaitkan nomor port server, serverPort, dengan soket ini:

```
serverSocket.bind((","+serverPort))
```

Tetapi dengan TCP, serverSocket akan menjadi soket penyambutan kami. Setelah memasang pintu penyambutan ini, kami akan menunggu dan mendengarkan beberapa klien mengetuk pintu:

```
serverSocket.mendengarkan(1)
```

Baris ini membuat server mendengarkan permintaan koneksi TCP dari klien. Parameter menentukan jumlah maksimum koneksi antrean (setidaknya 1).

```
connectionSocket, addr = serverSocket.accept()
```

Saat klien mengetuk pintu ini, program memanggil metode accept() untuk serverSocket, yang membuat soket baru di server, yang disebut connectionSocket, yang didedikasikan untuk klien khusus ini. Klien dan server kemudian menyelesaikan jabat tangan, membuat koneksi TCP antara clientSocket klien dan connectionSocket server. Dengan koneksi TCP dibuat, klien dan server sekarang dapat mengirim byte satu sama lain melalui koneksi tersebut. Dengan TCP, semua byte yang dikirim dari satu sisi tidak hanya dijamin tiba di sisi lain tetapi juga dijamin tiba secara berurutan.

```
connectionSocket.close()
```

168 BAB 2 • LAPISAN APLIKASI

Dalam program ini, setelah mengirimkan kalimat yang dimodifikasi ke klien, kami menutup soket koneksi. Tetapi karena serverSocket tetap terbuka, klien lain sekarang dapat mengetuk pintu dan mengirim kalimat ke server untuk dimodifikasi.

Ini melengkapi diskusi kami tentang pemrograman soket di TCP. Anda didorong untuk menjalankan dua program di dua host terpisah, dan juga memodifikasinya untuk mencapai tujuan yang sedikit berbeda. Anda harus membandingkan pasangan program UDP dengan pasangan program TCP dan melihat perbedaannya. Anda juga harus melakukan banyak tugas pemrograman soket yang dijelaskan di akhir Bab 2, 4, dan 7. Akhirnya, kami berharap suatu hari nanti, setelah menguasai program soket ini dan lebih lanjut, Anda akan menulis aplikasi jaringan populer Anda sendiri, menjadi sangat kaya dan terkenal, dan ingat penulis buku teks ini!

2.8 Ringkasan

Dalam bab ini, kita telah mempelajari aspek konseptual dan implementasi aplikasi kerja jaringan. Kami telah belajar tentang arsitektur client-server di mana-mana yang diadopsi oleh banyak aplikasi Internet dan melihat penggunaannya dalam protokol HTTP, FTP, SMTP, POP3, dan DNS. Kami telah mempelajari protokol tingkat aplikasi penting ini, dan aplikasi terkait yang sesuai (Web, transfer file, email, dan DNS) secara mendetail. Kami juga telah belajar tentang arsitektur P2P yang semakin umum dan bagaimana penggunaannya di banyak aplikasi. Kami telah memeriksa bagaimana socket API dapat digunakan untuk membangun aplikasi jaringan. Kami telah menelusuri penggunaan soket untuk layanan transportasi end-to-end berorientasi koneksi (TCP) dan connectionless (UDP). Langkah pertama dalam perjalanan kita menyusuri arsitektur jaringan berlapis kini telah selesai!

Di awal buku ini, di Bagian 1.1, kami memberikan definisi protokol yang agak kabur dan sederhana: "format dan urutan pesan yang dipertukarkan antara dua atau lebih entitas yang berkomunikasi, serta tindakan yang diambil pada trans misi dan/atau penerimaan pesan atau peristiwa lain." Materi dalam bab ini, dan khususnya studi terperinci kami tentang protokol HTTP, FTP, SMTP, POP3, dan DNS, kini telah menambah substansi definisi ini. Protokol adalah konsep kunci dalam jaringan; studi kami tentang protokol aplikasi sekarang telah memberi kami kesempatan untuk mengembangkan perasaan yang lebih intuitif tentang apa itu protokol.

Di Bagian 2.1, kami menjelaskan model layanan yang ditawarkan TCP dan UDP ke aplikasi yang memanggilnya. Kami melihat lebih dekat pada model layanan ini ketika kami mengembangkan aplikasi sederhana yang berjalan di atas TCP dan UDP di Bagian 2.7. Namun, kami telah berbicara sedikit tentang bagaimana TCP dan UDP menyediakan model layanan ini. Misalnya, kita tahu bahwa TCP menyediakan layanan data yang andal, tetapi kita belum mengatakan bagaimana melukannya. Pada bab selanjutnya kita akan melihat dengan hati-hati tidak hanya apa , tetapi juga *bagaimana* dan *mengapa* protokol transport.

Dilengkapi dengan pengetahuan tentang struktur aplikasi Internet dan protokol tingkat aplikasi, kita sekarang siap untuk melangkah lebih jauh ke tumpukan protokol dan menguji lapisan transport di Bab 3.



Masalah dan Pertanyaan Pekerjaan Rumah

Bab 2 Tinjau Pertanyaan

BAGIAN 2.1

- R1. Buat daftar lima aplikasi Internet nonproprietary dan lapisan aplikasi protokol yang mereka gunakan.
- R2. Apa perbedaan antara arsitektur jaringan dan arsitektur aplikasi?
- R3. Untuk sesi komunikasi antara sepasang proses, yang merupakan proses klien dan mana servernya?
- R4. Untuk aplikasi berbagi file P2P, apakah Anda setuju dengan pernyataan, "Tidak ada gagasan sisi klien dan server dari sesi komunikasi"? Mengapa atau mengapa tidak?
- R5. Informasi apa yang digunakan oleh proses yang berjalan di satu host untuk mengidentifikasi proses yang berjalan di host lain?
- R6. Misalkan Anda ingin melakukan transaksi dari klien jarak jauh ke server sebagai secepat mungkin. Apakah Anda menggunakan UDP atau TCP? Mengapa?
- R7. Mengacu pada Gambar 2.4, kita melihat bahwa tidak ada aplikasi yang tercantum pada Gambar 2.4 tidak memerlukan kehilangan data dan waktu. Bisakah Anda membayangkan aplikasi yang tidak memerlukan kehilangan data dan juga sangat sensitif terhadap waktu?
- R8. Buat daftar empat kelas luas layanan yang dapat disediakan oleh protokol transport. Untuk setiap kelas layanan, tunjukkan apakah UDP atau TCP (atau keduanya) menyediakan layanan tersebut.
- R9. Ingatlah bahwa TCP dapat ditingkatkan dengan SSL untuk menyediakan layanan keamanan proses-ke-proses, termasuk enkripsi. Apakah SSL beroperasi pada lapisan transport atau lapisan aplikasi? Jika pengembang aplikasi ingin TCP ditingkatkan dengan SSL, apa yang harus dilakukan pengembang?

BAGIAN 2.2–2.5

- R10. Apa yang dimaksud dengan protokol jabat tangan?
- R11. Mengapa HTTP, FTP, SMTP, dan POP3 berjalan di atas TCP daripada di UDP?
- R12. Pertimbangkan situs e-niaga yang ingin menyimpan catatan pembelian untuk setiap pelanggannya. Jelaskan bagaimana hal ini dapat dilakukan dengan cookie.

170 BAB 2 • LAPISAN APLIKASI

- R13. Jelaskan bagaimana Web caching dapat mengurangi keterlambatan dalam menerima permintaan obyek. Akankah caching Web mengurangi penundaan untuk semua objek yang diminta oleh pengguna atau hanya beberapa objek? Mengapa?
- R14. Telnet ke server Web dan mengirim pesan permintaan multiline. Sertakan dalam pesan permintaan baris header If-modified-since: untuk memaksa pesan respons dengan kode status 304 Not Modified.
- R15. Mengapa dikatakan bahwa FTP mengirimkan informasi kontrol "out-of-band"?
- R16. Misalkan Alice, dengan akun email berbasis Web (seperti Hotmail atau gmail), mengirim pesan ke Bob, yang mengakses emailnya dari server emailnya menggunakan POP3. Diskusikan bagaimana pesan tersebut sampai dari host Alice ke host Bob. Pastikan untuk membuat daftar rangkaian protokol lapisan aplikasi yang digunakan untuk memindahkan pesan antara dua host.
- R17. Cetak header pesan email yang baru saja Anda terima. Berapa banyak baris header Diterima: yang ada? Analisis setiap baris header dalam pesan.
- R18. Dari sudut pandang pengguna, apa perbedaan antara mode unduh dan hapus dan mode unduh dan simpan di POP3?
- R19. Mungkinkah server Web dan server email organisasi memiliki alias yang persis sama untuk nama host (misalnya, foo.com)? Apa jenis RR yang berisi nama host dari server email?
- R20. Periksa email yang Anda terima, dan periksa header pesan yang dikirim dari pengguna dengan alamat email .edu. Apakah mungkin untuk menentukan dari header alamat IP host tempat pesan dikirim? Lakukan hal yang sama untuk pesan yang dikirim dari akun gmail.

BAGIAN 2.6

- R21. Di BitTorrent, misalkan Alice memberikan potongan kepada Bob selama interval 30 detik. Akankah Bob membalas budi dan memberikan bongkahan kepada Alice dalam interval yang sama ini? Mengapa atau mengapa tidak?
- R22. Pertimbangkan rekan baru Alice yang bergabung dengan BitTorrent tanpa memilikinya potongan. Tanpa potongan apa pun, dia tidak dapat menjadi pengunggah empat teratas untuk rekan lainnya, karena dia tidak memiliki apa pun untuk diunggah. Lalu bagaimana Alice mendapatkan potongan pertamanya?
- R23. Apa itu jaringan overlay? Apakah itu termasuk router? Apa tepi dalam jaringan overlay?
- R24. Pertimbangkan DHT dengan topologi mesh overlay (yaitu, setiap peer melacak semua peer dalam sistem). Apa keuntungan dan kerugian dari desain seperti itu? Apa kelebihan dan kekurangan DHT sirkuler (tanpa jalan pintas)?

- R25. Sebutkan setidaknya empat aplikasi berbeda yang secara alami cocok untuk arsitektur P2P. (*Petunjuk:* Distribusi file dan pesan instan adalah dua.)

BAGIAN 2.7

- R26. Dalam Bagian 2.7, server UDP dijelaskan hanya membutuhkan satu soket, sedangkan server TCP membutuhkan dua soket. Mengapa? Jika server TCP mendukung n koneksi simultan, masing-masing dari host klien yang berbeda, berapa banyak soket yang dibutuhkan server TCP?
- R27. Untuk aplikasi client-server melalui TCP yang dijelaskan pada Bagian 2.7, mengapa program server harus dijalankan sebelum program klien? Untuk aplikasi client server melalui UDP, mengapa program client dapat dijalankan sebelum program server?



Masalah

P1. Benar atau salah?

- A. Seorang pengguna meminta halaman Web yang terdiri dari beberapa teks dan tiga gambar. Untuk halaman ini, klien akan mengirimkan satu pesan permintaan dan menerima empat pesan tanggapan.
- B. Dua halaman Web yang berbeda (misalnya, www.mit.edu/research.html dan www.mit.edu/students.html) dapat dikirim melalui koneksi yang sama per sistent.
- C. Dengan koneksi nonpersistent antara browser dan server asal, dimungkinkan untuk satu segmen TCP untuk membawa dua pesan permintaan HTTP yang berbeda.
- D. Tanggal: tajuk dalam pesan respons HTTP menunjukkan kapan objek dalam respons terakhir diubah.
- e. Pesan respons HTTP tidak pernah memiliki badan pesan kosong.

P2. Baca RFC 959 untuk FTP. Daftar semua perintah klien yang didukung oleh RFC.

P3. Pertimbangkan klien HTTP yang ingin mengambil dokumen Web pada saat tertentu URL. Alamat IP server HTTP awalnya tidak diketahui. Protokol transport dan lapisan aplikasi apa selain HTTP yang diperlukan dalam skenario ini?

P4. Pertimbangkan rangkaian karakter ASCII berikut yang ditangkap oleh Wireshark ketika browser mengirim pesan HTTP GET (yaitu, ini adalah isi sebenarnya dari pesan HTTP GET). Karakter `<cr><lf>` adalah karakter carriage return dan line-feed (yaitu, string karakter yang dicetak miring `<cr>` dalam teks di bawah ini mewakili karakter carriage-return tunggal yang terdapat pada titik tersebut di header HTTP) . Jawab pertanyaan berikut, tunjukkan di bagian mana pesan HTTP GET di bawah ini Anda menemukan jawabannya.

172 BAB 2 • LAPISAN APLIKASI

```
DAPATKAN /cs453/index.html HTTP/1.1<br><lf>Host: gai.a.cs.umass.edu<br><lf>User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804  
Netscape/7.2 (ax)<br><lf>Terima: ex t/xml, aplikasi/xml, aplikasi/xhtml+xml, teks /  
html;q=0.9, teks/plain;q=0.8,image/png,*/*;q=0.5<br><lf>Bahasa Terima: en-US,en;q=0.5<br><lf>Terima Pengodean: zip,deflate<br><lf>Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7<br><lf>Keep-Alive: 300<br><lf>Koneksi:keep-alive <br><lf><br><lf>
```

- A. Apa URL dokumen yang diminta oleh browser? B. Versi HTTP apa yang dijalankan browser? C. Apakah browser meminta koneksi non-persisten atau persisten? D. Apa alamat IP host tempat browser berjalan? e. Jenis browser apa yang memulai pesan ini? Mengapa jenis browser

diperlukan dalam pesan permintaan HTTP?

- P5. Teks di bawah menunjukkan balasan yang dikirim dari server sebagai tanggapan atas pesan HTTP GET pada pertanyaan di atas. Jawablah pertanyaan-pertanyaan berikut, dengan menunjukkan di mana dalam pesan di bawah ini Anda menemukan jawabannya.

```
HTTP/1.1 200 OK<br><lf>Tanggal: Sel, 07 Mar 2008 12:39:45 GMT<br><lf>Server:  
Apache/2.0.52 (Fedora)<br><lf>Terakhir Dimodifikasi: Sabtu, 10 Des 2005 18:27:46  
GMT<br><lf>ETag: "526c3-f22-a88a4c80"<br><lf>Jangkauan Terima : byte<br><lf>Panjang Konten: 3874<br><lf>Keep-Alive: timeout=max=100<br><lf>Koneksi: Keep-Alive<br><lf>Content-Type: text/html; charset=ISO-8859-1<br><lf><cr><lf><!doctype html publik "-//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><judul>CMPSCI 453 / 591 / Beranda NTU-ST550A Musim Semi 2005</title><lf></head><lf><lebih banyak teks dokumen berikut di sini (tidak ditampilkan)>
```

- A. Apakah server berhasil menemukan dokumen atau tidak? Jam berapa balasan dokumen diberikan?
B. Kapan dokumen terakhir diubah?
C. Berapa banyak byte yang ada dalam dokumen yang dikembalikan? D. Berapa 5 byte pertama dari dokumen yang dikembalikan? Apakah server menyetujui koneksi yang persisten?

P6. Dapatkan spesifikasi HTTP/1.1 (RFC 2616). Jawablah pertanyaan-pertanyaan berikut: a.

Jelaskan mekanisme yang digunakan untuk pensinyalan antara klien dan server untuk menunjukkan bahwa koneksi persisten sedang ditutup. Bisakah klien, server, atau keduanya memberi sinyal penutupan koneksi? B. Layanan enkripsi apa yang disediakan oleh HTTP? C. Bisakah klien membuka tiga atau lebih koneksi simultan dengan yang diberikan pelayan?

D. Baik server atau klien dapat menutup koneksi transportasi di antara mereka jika salah satu mendeteksi koneksi telah menganggur selama beberapa waktu. Mungkinkah satu sisi mulai menutup koneksi sementara sisi lain mengirimkan data melalui koneksi ini? Menjelaskan.

P7. Misalkan dalam browser Web Anda, Anda mengklik tautan untuk mendapatkan halaman Web.

Alamat IP untuk URL terkait tidak di-cache di host lokal Anda, sehingga pencarian DNS diperlukan untuk mendapatkan alamat IP. Misalkan n server DNS dikunjungi sebelum host Anda menerima alamat IP dari DNS; kunjungan berturut-turut menimbulkan RTT sebesar RTT_1, \dots, RTT_n . Selanjutnya anggaplah bahwa halaman Web yang terkait dengan tautan tersebut berisi tepat satu objek, yang terdiri dari sejumlah kecil teks HTML. Biarkan RTT_0 menunjukkan RTT antara host lokal dan server yang berisi objek. Dengan asumsi nol waktu transmisi objek, berapa banyak waktu yang berlalu sejak klien mengklik tautan hingga klien menerima objek?

P8. Mengacu pada Soal P7, misalkan file HTML mereferensikan delapan objek yang sangat kecil pada server yang sama. Mengabaikan waktu transmisi, berapa banyak waktu berlalu dengan

a. HTTP non-persisten tanpa koneksi TCP paralel? B. HTTP non-persisten dengan browser yang dikonfigurasi untuk 5 koneksi paralel?

C. HTTP terus-menerus?

P9. Perhatikan Gambar 2.12, di mana ada jaringan kelembagaan yang terhubung ke Internet.

Misalkan ukuran objek rata-rata adalah 850.000 bit dan tingkat permintaan rata-rata dari browser institusi ke server asal adalah 16 permintaan per detik. Anggap juga bahwa jumlah waktu yang diperlukan dari saat router di sisi Internet dari link akses meneruskan permintaan HTTP hingga menerima respons adalah rata-rata tiga detik (lihat Bagian 2.2.5).

Model total waktu respons rata-rata sebagai jumlah rata-rata penundaan akses (yaitu, penundaan dari router Internet ke router institusi) dan rata-rata penundaan Internet. Untuk tundaan akses rata-rata, gunakan $\bar{y}/(1 - \bar{y})$, di mana \bar{y} adalah waktu rata-rata yang diperlukan untuk mengirim objek melalui tautan akses dan adalah laju kedatangan objek ke tautan akses.

A. Temukan total waktu respons rata-rata. B. Sekarang misalkan cache dipasang di LAN institusional. Misalkan tingkat kesalahan adalah 0,4. Temukan total waktu respons.

174 BAB 2 • LAPISAN APLIKASI

P10. Pertimbangkan tautan pendek sepanjang 10 meter, di mana pengirim dapat mengirimkan dengan kecepatan 150 bit/detik di kedua arah. Misalkan paket yang berisi data panjangnya 100.000 bit, dan paket yang hanya berisi kontrol (misalnya, ACK atau jabat tangan) panjangnya 200 bit. Asumsikan bahwa N koneksi paralel masing-masing mendapatkan $1/N$ dari bandwidth tautan. Sekarang pertimbangkan protokol HTTP, dan misalkan setiap objek yang diunduh panjangnya 100 Kbit, dan objek yang diunduh awal berisi 10 objek referensi dari pengirim yang sama. Apakah unduhan paralel melalui contoh paralel HTTP non-persisten masuk akal dalam kasus ini? Sekarang pertimbangkan HTTP persisten. Apakah Anda mengharapkan keuntungan yang signifikan atas kasus yang tidak terus-menerus? Berikan alasan dan jelaskan jawaban Anda.

P11. Pertimbangkan skenario yang diperkenalkan pada masalah sebelumnya. Sekarang misalkan tautan tersebut dibagikan oleh Bob dengan empat pengguna lainnya. Bob menggunakan contoh paralel HTTP non-persisten, dan empat pengguna lainnya menggunakan HTTP non-persisten tanpa unduhan paralel. A. Apakah koneksi paralel Bob membantunya mendapatkan halaman Web lebih cepat?

Mengapa atau mengapa tidak?

B. Jika kelima pengguna membuka lima instance paralel dari HTTP non-persisten, apakah koneksi paralel Bob masih bermanfaat? Mengapa atau mengapa tidak?

P12. Tulis program TCP sederhana untuk server yang menerima jalur masukan dari a klien dan mencetak baris ke output standar server. (Anda dapat melakukannya dengan memodifikasi program TCPServer.py dalam teks.) Kompilasi dan jalankan program Anda. Di mesin lain mana pun yang berisi browser Web, setel server proxy di browser ke host yang menjalankan program server Anda; juga mengkonfigurasi nomor port dengan tepat. Browser Anda sekarang harus mengirim pesan permintaan GET ke server Anda, dan server Anda harus menampilkan pesan pada keluaran standarnya. Gunakan platform ini untuk menentukan apakah browser Anda menghasilkan pesan GET bersyarat untuk objek yang di-cache secara lokal.

P13. Apa perbedaan antara MAIL FROM: di SMTP dan Dari: di pesan surat itu sendiri?

P14. Bagaimana SMTP menandai akhir dari badan pesan? Bagaimana dengan HTTP? Bisakah HTTP menggunakan metode yang sama dengan SMTP untuk menandai akhir dari badan pesan? Menjelaskan.

P15. Baca RFC 5321 untuk SMTP. Apa kepanjangan dari MTA? Pertimbangkan email spam yang diterima berikut (dimodifikasi dari email spam asli). Dengan asumsi hanya pembuat email spam ini yang berbahaya dan semua host lainnya jujur, identifikasi host jahat yang telah menghasilkan email spam ini.

Dari - Jum 07 Nov 13:41:30 2008 Return-Path:

<tennis5@pp33head.com> Diterima: dari barmail.cs.umass.edu
(barmail.cs.umass.edu [128.119.240.3]) oleh cs.umass.edu

(8.13.1/8.12.6) untuk <hg@cs.umass.edu>; Jum, 7 Nov 2008 13:27:10 -0500

Diterima: dari asusus-4b96 (localhost [127.0.0.1]) oleh barmail.cs.umass.edu (Spam Firewall) untuk <hg@cs.umass.edu>; Jum, 7 Nov 2008 13:27:07 -0500 (EST)

Diterima: dari asusus-4b96 ([58.88.21.177]) oleh barmail.cs.umass.edu untuk <hg@cs.umass.edu>; Jum, 07 Nov 2008 13:27:07 -0500 (EST)

Diterima: dari [58.88.21.177] oleh inbnd55.exchangedd.com; Sab, 8 Nov 2008 01:27:07 +0700 Dari: "Jonny" <tennis5@pp33head.com> Kepada: <hg@cs.umass.edu> Perihal: Bagaimana mengamankan tabungan Anda

P16. Baca RFC POP3, RFC 1939. Apa tujuan dari UIDL POP3 memerintah?

P17. Pertimbangkan untuk mengakses email Anda dengan POP3.

- A. Misalkan Anda telah mengonfigurasi klien surat POP Anda untuk beroperasi dalam mode unduh-dan-hapus. Selesaikan transaksi berikut:

C: daftar

S: 1 498

S: 2 912

S: .

C: retr 1 **S:**

bla bla...

S:bla **S:** .

?

?

- B. Misalkan Anda telah mengonfigurasi klien surat POP Anda untuk beroperasi dalam mode unduh dan simpan. Selesaikan transaksi berikut:

C: daftar

S: 1 498

S: 2 912

S: .

C: retr 1 **S:**

bla bla...

S:bla **S:** .

?

?

176 BAB 2 • LAPISAN APLIKASI

C. Misalkan Anda telah mengonfigurasi klien surat POP Anda untuk beroperasi di mode unduh-dan-simpan. Menggunakan transkrip Anda di bagian (b), misalkan Anda mengambil pesan 1 dan 2, keluar dari POP, lalu lima menit kemudian Anda mengakses POP lagi untuk mengambil email baru. Misalkan dalam interval lima menit tidak ada pesan baru yang dikirimkan kepada Anda. Berikan transkrip sesi POP kedua ini.

P18. A. Apa itu database *whois* ?

B. Gunakan berbagai database whois di Internet untuk mendapatkan dua nama Server DNS. Tunjukkan database whois mana yang Anda gunakan.

C. Gunakan nslookup pada host lokal Anda untuk mengirim kueri DNS ke tiga server DNS: server DNS lokal Anda dan dua server DNS yang Anda temukan di bagian (b). Coba buat kueri untuk laporan Tipe A, NS, dan MX. Ringkas temuan Anda. D. Gunakan nslookup untuk menemukan server Web yang memiliki beberapa alamat IP. Apakah server Web institusi Anda (sekolah atau perusahaan) memiliki banyak alamat IP?

e. Gunakan database whois ARIN untuk menentukan rentang alamat IP yang digunakan oleh universitas Anda. F. Jelaskan bagaimana penyerang dapat menggunakan database whois dan alat nslookup untuk melakukan pengintaian pada sebuah institusi sebelum meluncurkan serangan.

G. Diskusikan mengapa database whois harus tersedia untuk umum.

P19. Dalam masalah ini, kami menggunakan alat *menggali* berguna yang tersedia di host Unix dan Linux untuk menjelajahi hierarki server DNS. Ingatlah bahwa pada Gambar 2.21, server DNS yang lebih tinggi dalam hierarki DNS mendeklasikan kueri DNS ke server DNS yang lebih rendah dalam hierarki, dengan mengirimkan kembali ke klien DNS nama server DNS dengan level yang lebih rendah. Pertama baca halaman manual untuk *menggali*, lalu jawab pertanyaan berikut. A. Dimulai dengan server DNS root (dari salah satu server root [am].root servers.net), memulai urutan kueri untuk alamat IP untuk server Web departemen Anda dengan menggunakan *dig*. Tampilkan daftar nama server DNS di rantai delegasi dalam menjawab kueri Anda. B. Ulangi bagian a) untuk beberapa situs Web populer, seperti google.com, yahoo.com, atau amazon.com.

P20. Misalkan Anda dapat mengakses cache di server DNS lokal departemen Anda.

Bisakah Anda mengusulkan cara untuk secara kasar menentukan server Web (di luar departemen Anda) yang paling populer di kalangan pengguna di departemen Anda? Menjelaskan.

P21. Misalkan departemen Anda memiliki server DNS lokal untuk semua komputer di departemen tersebut. Anda adalah pengguna biasa (yaitu, bukan administrator jaringan/sistem). Bisakah Anda menentukan apakah situs Web eksternal mungkin diakses dari komputer di departemen Anda beberapa detik yang lalu? Menjelaskan.

P22. Pertimbangkan untuk mendistribusikan file $F = 15$ Gbit ke N peer. Server memiliki kecepatan upload $us = 30$ Mbps, dan setiap peer memiliki kecepatan download $di = 2$ Mbps dan kecepatan upload u . Untuk $N = 10, 100$, dan 1.000 dan $u = 300$ Kbps, 700 Kbps, dan 2 Mbps, siapkan bagan yang memberikan waktu distribusi minimum untuk setiap kombinasi N dan u untuk distribusi client-server dan distribusi P2P.

P23. Pertimbangkan untuk mendistribusikan file bit F ke N peer menggunakan arsitektur client-server. Asumsikan model fluid di mana server dapat secara bersamaan mengirimkan ke beberapa peer, mentransmisikan ke setiap peer dengan kecepatan yang berbeda, selama kecepatan gabungan tidak melebihi $kita$. A. Misalkan $kita / N \leq dmin$. Tentukan skema distribusi yang memiliki waktu distribusi NF/us . B. Misalkan $kita / N > dmin$. Tentukan skema distribusi yang memiliki waktu distribusi $F/dmin$.

C. Simpulkan bahwa waktu distribusi minimum pada umumnya diberikan oleh $\max\{NF/(us), F/dmin\}$.

P24. Pertimbangkan untuk mendistribusikan file F bit ke N peer menggunakan arsitektur P2P. Asumsikan model fluida. Untuk kesederhanaan, asumsikan bahwa $dmin$ sangat besar, sehingga bandwidth pengunduhan rekan tidak pernah menjadi hambatan. A. Misalkan $us \leq (us + u_1 + \dots + u_N)/N$. Tentukan skema distribusi $(us + u_1 + \dots + u_N)/N$.

B. Misalkan $kita \leq (us + u_1 + \dots + u_N)/N$. Tentukan skema distribusi $(us + u_1 + \dots + u_N)/N$.

Simpulkan bahwa waktu distribusi minimum pada umumnya diberikan oleh $\max\{F/(us + u_1 + \dots + u_N), NF/(us + u_1 + \dots + u_N)\}$.

P25. Pertimbangkan jaringan overlay dengan N peer aktif, dengan setiap pasangan peer memiliki koneksi TCP aktif. Selain itu, misalkan koneksi TCP melewati total M router. Berapa banyak node dan edge yang ada di jaringan overlay yang sesuai?

P26. Misalkan Bob bergabung dengan torrent BitTorrent, tetapi dia tidak ingin mengunggah data apa pun ke rekan lain (disebut free-riding). A. Bob mengklaim bahwa dia dapat menerima salinan lengkap dari file yang dibagikan oleh gerombolan tersebut. Apakah klaim Bob mungkin? Mengapa atau mengapa tidak? B. Bob lebih lanjut mengklaim bahwa dia dapat lebih jauh lagi membuat "free-riding"-nya lebih efisien dengan menggunakan kumpulan beberapa komputer (dengan alamat IP yang berbeda) di lab komputer di departemennya. Bagaimana dia bisa melakukan itu?

P27. Dalam contoh DHT melingkar di Bagian 2.6.2, misalkan peer 3 mengetahui bahwa peer 5 telah pergi. Bagaimana rekan 3 memperbarui informasi status pengantinya? Peer mana yang sekarang menjadi penerus pertamanya? Penerus keduanya?

178 BAB 2 • LAPISAN APLIKASI

P28. Dalam contoh DHT sirkular di Bagian 2.6.2, misalkan peer baru 6 ingin bergabung dengan DHT dan peer 6 awalnya hanya mengetahui alamat IP peer 15. Langkah apa yang diambil?

P29. Karena bilangan bulat dalam $[0, 2^n - 1]$ dapat dinyatakan sebagai bilangan biner n-bit dalam DHT, setiap kunci dapat dinyatakan sebagai $k = (k_0, k_1, \dots, k_{n-1})$, dan setiap peer identifier dapat dinyatakan $p = (p_0, p_1, \dots, p_{n-1})$. Sekarang mari kita tentukan jarak XOR antara kunci k dan peer p sebagai

$$d(k, p) = \min_{j=0}^{n-1} |k_j - p_j| 2^j$$

Jelaskan bagaimana metrik ini dapat digunakan untuk menetapkan pasangan (kunci, nilai) ke peer. (Untuk mempelajari tentang cara membuat DHT yang efisien menggunakan metrik natural ini, lihat [Maymounkov 2002] yang menjelaskan Kademia DHT.)

P30. Karena DHT adalah jaringan overlay, mereka mungkin tidak cocok dengan jaringan fisik underlay dengan baik dalam artian bahwa dua rekan tetangga mungkin secara fisik sangat jauh; misalnya, satu rekan bisa berada di Asia dan tetangganya bisa berada di Amerika Utara. Jika kita menetapkan pengidentifikasi secara acak dan seragam ke peer yang baru bergabung, apakah skema penugasan ini akan menyebabkan ketidakcocokan? Menjelaskan. Dan bagaimana ketidaksesuaian tersebut akan mempengaruhi kinerja DHT?

P31. Instal dan kompilasi program Python TCPClient dan UDPClient menjadi satu host dan TCPServer dan UDPServer di host lain.

A. Misalkan Anda menjalankan TCPClient sebelum Anda menjalankan TCPServer. Apa yang terjadi?
Mengapa?

B. Misalkan Anda menjalankan UDPClient sebelum Anda menjalankan UDPServer. Apa yang terjadi?
Mengapa?

C. Apa yang terjadi jika Anda menggunakan nomor port yang berbeda untuk klien dan server sisi?

P32. Misalkan di UDPClient.py, setelah kita membuat soket, kita tambahkan baris:

```
clientSocket.bind(("0.0.0.0", 5432))
```

Apakah perlu mengubah UDPServer.py? Berapa nomor port untuk soket di UDPClient dan UDPServer? Apa yang mereka sebelum membuat perubahan ini?

P33. Bisakah Anda mengonfigurasi browser Anda untuk membuka beberapa koneksi simultan ke situs Web? Apa keuntungan dan kerugian memiliki sejumlah besar koneksi TCP simultan?

P34 Kita telah melihat bahwa soket TCP Internet memperlakukan data yang dikirim sebagai aliran byte tetapi soket UDP mengenali batas pesan. Apa satu

keuntungan dan satu kerugian dari API berorientasi byte versus memiliki API yang secara eksplisit mengenali dan mempertahankan batas pesan yang ditentukan aplikasi?

P35. Apa itu server Web Apache? Harganya berapa? Fungsi apa yang dimiliki saat ini? Anda mungkin ingin melihat Wikipedia untuk menjawab pertanyaan ini.

P36. Banyak klien BitTorrent menggunakan DHT untuk membuat pelacak terdistribusi. Untuk ini DHT, apa itu "kunci" dan apa "nilainya"?



Tugas Pemrograman Soket

Situs Web pendamping mencakup enam tugas pemrograman soket. Empat tugas pertama dirangkum di bawah ini. Tugas kelima memanfaatkan protokol ICMP dan dirangkum di akhir Bab 4. Tugas keenam menggunakan protokol multimedia dan dirangkum di akhir Bab 7. Sangat disarankan agar siswa menyelesaikan beberapa, jika tidak semua, dari ini tugas.

Siswa dapat menemukan detail lengkap dari tugas ini, serta cuplikan penting dari kode Python, di situs web <http://www.awl.com/kurose-ross>.

Tugas 1: Server Web

Dalam tugas ini, Anda akan mengembangkan server Web sederhana dengan Python yang hanya mampu memproses satu permintaan. Secara khusus, server Web Anda akan (i) membuat soket koneksi saat dihubungi oleh klien (browser); (ii) menerima permintaan HTTP dari koneksi ini; (iii) menguraikan permintaan untuk menentukan file spesifik yang diminta; (iv) mendapatkan file yang diminta dari sistem file server; (v) membuat pesan respons HTTP yang terdiri dari file yang diminta didahului dengan baris header; dan (vi) mengirim respons melalui koneksi TCP ke browser yang meminta. Jika browser meminta file yang tidak ada di server Anda, server Anda akan mengembalikan pesan kesalahan "404 Tidak Ditemukan".

Di situs Web pendamping, kami menyediakan kode kerangka untuk server Anda. Tugas Anda adalah menyelesaikan kode, menjalankan server Anda, dan kemudian menguji server Anda dengan mengirimkan permintaan dari browser yang berjalan di host yang berbeda. Jika Anda menjalankan server Anda di host yang sudah memiliki server Web yang berjalan di dalamnya, maka Anda harus menggunakan port yang berbeda dari port 80 untuk server Web Anda.

Tugas 2: Pinger UDP

Dalam tugas pemrograman ini, Anda akan menulis program ping klien dengan Python. Klien Anda akan mengirim pesan ping sederhana ke server, menerima kembali pesan pong yang sesuai dari server, dan menentukan penundaan antara saat klien mengirim pesan ping dan menerima pesan pong. Penundaan ini disebut Round Trip Time (RTT). Fungsionalitas yang disediakan oleh klien dan server adalah

180 BAB 2 • LAPISAN APLIKASI

mirip dengan fungsionalitas yang disediakan oleh program ping standar yang tersedia di sistem operasi modern. Namun, program ping standar menggunakan Internet Control Message Protocol (ICMP) (yang akan kita pelajari di Bab 4). Di sini kita akan membuat program ping berbasis UDP yang tidak standar (tapi sederhana).

Program ping Anda adalah mengirim 10 pesan ping ke server target melalui UDP. Untuk setiap pesan, klien Anda harus menentukan dan mencetak RTT saat pesan ding pong yang sesuai dikembalikan. Karena UDP adalah protokol yang tidak dapat diandalkan, paket yang dikirim oleh klien atau server mungkin akan hilang. Untuk alasan ini, klien tidak dapat menunggu tanpa batas waktu untuk membalas pesan ping. Anda harus meminta klien menunggu hingga satu detik untuk balasan dari server; jika tidak ada balasan yang diterima, klien harus berasumsi bahwa paket tersebut hilang dan mencetak pesan yang sesuai.

Dalam tugas ini, Anda akan diberikan kode lengkap untuk server (tersedia di situs Web pendamping). Tugas Anda adalah menulis kode klien, yang akan sangat mirip dengan kode server. Disarankan agar Anda terlebih dahulu mempelajari kode server dengan cermat. Anda kemudian dapat menulis kode klien Anda, dengan bebas memotong dan menempelkan baris dari kode server.

Tugas 3: Klien Surat

Tujuan dari tugas pemrograman ini adalah untuk membuat klien email sederhana yang mengirim email ke penerima mana pun. Klien Anda perlu membuat koneksi TCP dengan server surat (misalnya, server surat Google), berdialog dengan server surat menggunakan protokol SMTP, mengirim pesan email ke penerima (misalnya, teman Anda) melalui server surat, dan terakhir tutup koneksi TCP dengan server email.

Untuk tugas ini, situs Web pendamping menyediakan kode kerangka untuk klien Anda. Tugas Anda adalah menyelesaikan kode dan mengujinya dengan mengirim email ke akun pengguna yang berbeda. Anda juga dapat mencoba mengirim melalui server yang berbeda (misalnya, melalui server surat Google dan melalui server surat universitas Anda).

Tugas 4: Proksi Web Multi-Utas

Dalam tugas ini, Anda akan mengembangkan proxy Web. Saat proxy Anda menerima permintaan HTTP untuk suatu objek dari browser, proxy akan menghasilkan permintaan HTTP baru untuk objek yang sama dan mengirimkannya ke server asal. Saat proxy menerima respons HTTP yang sesuai dengan objek dari server asal, proxy membuat respons HTTP baru, termasuk objek, dan mengirimkannya ke klien. Proksi ini akan multi-utas, sehingga dapat menangani banyak permintaan sekaligus.

Untuk tugas ini, situs Web pendamping menyediakan kode kerangka untuk server proxy. Tugas Anda adalah menyelesaikan kode, lalu mengujinya dengan meminta browser yang berbeda meminta objek Web melalui proxy Anda.



Lab Wireshark: HTTP

Setelah kaki kami basah dengan paket sniffer Wireshark di Lab 1, kami sekarang siap menggunakan Wireshark untuk menyelidiki protokol yang sedang beroperasi. Di lab ini, kita akan mempelajari beberapa aspek protokol HTTP: interaksi dasar GET/reply, format pesan HTTP, mengambil file HTML berukuran besar, mengambil file HTML dengan URL tersemat, koneksi persisten dan non-persisten, serta autentikasi dan keamanan HTTP .

Seperti halnya semua lab Wireshark, deskripsi lengkap tentang lab ini tersedia di situs Web buku ini, <http://www.awl.com/kurose-ross>.



VideoNote

Menggunakan
Wireshark untuk menyelidiki
Protokol HTTP



Lab Wireshark: DNS

Di lab ini, kita melihat lebih dekat sisi klien DNS, protokol yang menerjemahkan nama host Internet ke alamat IP. Ingat dari Bagian 2.5 bahwa peran klien dalam DNS relatif sederhana—klien mengirim kueri ke server DNS lokalnya dan menerima respons kembali. Banyak yang bisa terjadi di bawah penutup, tidak terlihat oleh klien DNS, karena server DNS hierarkis berkomunikasi satu sama lain untuk menyelesaikan permintaan DNS klien secara rekursif atau iteratif. Namun, dari sudut pandang klien DNS, protokolnya cukup sederhana—kueri diformulasikan ke server DNS lokal dan respons diterima dari server tersebut. Kami mengamati DNS beraksi di lab ini.

Seperti halnya semua lab Wireshark, deskripsi lengkap tentang lab ini tersedia di situs Web buku ini, <http://www.awl.com/kurose-ross>.

WAWANCARA DENGAN...

Marc Andreessen

Marc Andreessen adalah co-creator Mosaic, browser Web yang mempopulerkan World Wide Web pada tahun 1993. Mosaic memiliki antarmuka yang bersih dan mudah dipahami dan merupakan browser pertama yang menampilkan gambar sejajar dengan teks. Pada tahun 1994, Marc Andreessen dan Jim Clark mendirikan Netscape, yang perambannya sejauh ini merupakan peramban paling populer hingga pertengahan 1990-an. Netscape juga mengembangkan protokol Secure Sockets Layer (SSL) dan banyak produk server Internet, termasuk server surat dan server Web berbasis SSL. Dia sekarang menjadi salah satu pendiri dan mitra umum firma modal venture Andreessen Horowitz, karena melihat pengembangan portofolio dengan kepemilikan yang meliputi Facebook, Foursquare, Groupon, Jawbone, Twitter, dan Zynga. Dia melayani di banyak papan, termasuk Bump, eBay, Glam Media, Facebook, dan Hewlett-Packard. Dia memegang gelar BS dalam Ilmu Komputer dari University of Illinois di Urbana-Champaign.



Bagaimana Anda menjadi tertarik pada komputasi? Apakah Anda selalu tahu bahwa Anda ingin bekerja di bidang teknologi informasi?

Revolusi video game dan komputasi personal terjadi tepat ketika saya tumbuh dewasa—komputasi personal adalah batas teknologi baru di akhir tahun 70-an dan awal 80-an. Dan bukan hanya Apple dan IBM PC, tetapi juga ratusan perusahaan baru seperti Commodore dan Atari. Saya belajar sendiri memprogram dari sebuah buku berjudul "Instant Freeze-Dried BASIC" pada usia 10 tahun, dan mendapatkan komputer pertama saya (Komputer Warna TRS-80—lihatlah!) pada usia 12 tahun.

Tolong jelaskan satu atau dua proyek paling menarik yang pernah Anda kerjakan selama karir Anda. Apa tantangan terbesarnya?

Tidak diragukan lagi, proyek yang paling menarik adalah browser web Mosaic asli pada tahun '92-'93—and tantangan terbesar adalah membuat siapa pun menganggapnya serius saat itu. Pada saat itu, semua orang mengira masa depan interaktif akan disampaikan sebagai "televisi interaktif" oleh perusahaan besar, bukan sebagai Internet oleh perusahaan rintisan.

Apa yang membuat Anda bersemangat tentang masa depan jaringan dan Internet? Apa kekhawatiran terbesar Anda?

Hal yang paling menarik adalah batas besar aplikasi dan layanan yang belum dijelajahi yang dapat dijelajahi oleh programmer dan pengusaha—Internet telah melepaskan kreativitas di

tingkat yang menurut saya belum pernah kita lihat sebelumnya. Kekhawatiran terbesar saya adalah prinsip konsekuensi yang tidak diinginkan—kita tidak selalu tahu implikasi dari apa yang kita lakukan, seperti Internet yang digunakan oleh pemerintah untuk menjalankan tingkat baru pengawasan terhadap warga negara.

Apakah ada sesuatu yang harus diperhatikan siswa secara khusus seiring dengan kemajuan teknologi Web?

Laju perubahan—hal terpenting untuk dipelajari adalah cara belajar—cara beradaptasi secara fleksibel terhadap perubahan dalam teknologi tertentu, dan cara tetap berpikiran terbuka tentang peluang dan kemungkinan baru saat Anda menjalani karier.

Siapa orang yang menginspirasi Anda secara profesional?

Vannevar Bush, Ted Nelson, Doug Engelbart, Nolan Bushnell, Bill Hewlett dan Dave Packard, Ken Olsen, Steve Jobs, Steve Wozniak, Andy Grove, Grace Hopper, Hedy Lamarr, Alan Turing, Richard Stallman.

Apa rekomendasi Anda untuk siswa yang ingin mengejar karir di bidang komputasi dan teknologi informasi?

Masuklah sedalam mungkin untuk memahami bagaimana teknologi dibuat, dan kemudian lengkapi dengan mempelajari cara kerja bisnis.

Bisakah teknologi memecahkan masalah dunia?

Tidak, tapi kami memajukan standar hidup orang-orang melalui pertumbuhan ekonomi, dan sebagian besar pertumbuhan ekonomi sepanjang sejarah berasal dari teknologi—jadi itu bagus.

Halaman ini sengaja dikosongkan

BAB 3

Mengangkut Lapisan

Berada di antara lapisan aplikasi dan jaringan, lapisan transport adalah bagian sentral dari arsitektur jaringan berlapis. Ini memiliki peran penting dalam menyediakan layanan komunikasi langsung ke proses aplikasi yang berjalan di host yang berbeda.

Pendekatan pedagogik yang kami ambil dalam bab ini adalah untuk bergantian antara diskusi tentang prinsip-prinsip transport-layer dan diskusi tentang bagaimana prinsip-prinsip ini diterapkan dalam protokol yang ada; seperti biasa, penekanan khusus akan diberikan pada protokol Internet, khususnya protokol lapisan transport TCP dan UDP.

Kita akan mulai dengan mendiskusikan hubungan antara transport dan network layer. Ini menetapkan tahapan untuk memeriksa fungsi kritis pertama dari lapisan transport—memperluas layanan pengiriman lapisan jaringan antara dua sistem akhir menjadi layanan pengiriman antara dua proses lapisan aplikasi yang berjalan pada sistem akhir. Kami akan mengilustrasikan fungsi ini dalam cakupan kami tentang protokol transport tanpa koneksi Internet, UDP.

Kami kemudian akan kembali ke prinsip dan menghadapi salah satu masalah paling mendasar dalam jaringan komputer—bagaimana dua entitas dapat berkomunikasi dengan andal melalui media yang dapat kehilangan dan merusak data. Melalui serangkaian skenario yang semakin rumit (dan realistik!), kami akan membangun serangkaian teknik yang digunakan protokol transportasi untuk memecahkan masalah ini. Kami kemudian akan menunjukkan bagaimana prinsip-prinsip ini diwujudkan dalam TCP, protokol transport berorientasi koneksi Internet.

Kami selanjutnya akan beralih ke masalah fundamental kedua yang penting dalam jaringan — mengendalikan laju transmisi entitas lapisan transpor untuk menghindari, atau

186 BAB 3 • LAPISAN TRANSPORTASI

pulih dari, kemacetan dalam jaringan. Kami akan mempertimbangkan penyebab dan konsekuensi kemacetan, serta teknik pengendalian kemacetan yang umum digunakan.

Setelah mendapatkan pemahaman yang kuat tentang isu-isu di balik kontrol kemacetan, kita akan mempelajari pendekatan TCP untuk kontrol kemacetan.

3.1 Pendahuluan dan Layanan Transport-Layer

Dalam dua bab sebelumnya kita menyentuh peran lapisan transport dan layanan yang disediakannya. Mari kita tinjau dengan cepat apa yang telah kita pelajari tentang lapisan transport.

Protokol transport-layer menyediakan **komunikasi logis** antara proses aplikasi yang berjalan pada host yang berbeda. Yang kami maksud dengan *komunikasi logis* adalah bahwa dari perspektif aplikasi, seolah-olah host yang menjalankan proses terhubung secara langsung; pada kenyataannya, host mungkin berada di sisi berlawanan dari planet ini, terhubung melalui banyak router dan berbagai jenis tautan. Proses aplikasi menggunakan komunikasi logis yang disediakan oleh lapisan transport untuk mengirim pesan satu sama lain, bebas dari kekhawatiran detail infrastruktur fisik yang digunakan untuk membawa pesan ini. Gambar 3.1 mengilustrasikan pengertian komunikasi logis.

Seperi yang ditunjukkan pada Gambar 3.1, protokol transport-layer diimplementasikan pada sistem akhir tetapi tidak pada router jaringan. Di sisi pengirim, lapisan transport mengubah pesan lapisan aplikasi yang diterimanya dari proses pengiriman aplikasi menjadi paket lapisan transpor, yang dikenal sebagai **segmen** lapisan transpor dalam terminologi Internet.

Hal ini dilakukan dengan (mungkin) memecah pesan aplikasi menjadi potongan yang lebih kecil dan menambahkan header transport-layer ke setiap potongan untuk membuat segmen transport-layer. Lapisan transport kemudian meneruskan segmen ke lapisan jaringan pada sistem akhir pengiriman, di mana segmen tersebut dienkapsulasi dalam paket lapisan jaringan (data gram) dan dikirim ke tujuan. Penting untuk dicatat bahwa router jaringan hanya bertindak pada bidang lapisan jaringan datagram; yaitu, mereka tidak memeriksa bidang segmen transport-layer yang dienkapsulasi dengan datagram. Di sisi penerima, lapisan jaringan mengekstraksi segmen lapisan transpor dari datagram dan meneruskan segmen tersebut ke lapisan transpor. Lapisan transport kemudian memproses segmen yang diterima, membuat data dalam segmen tersebut tersedia untuk aplikasi penerima.

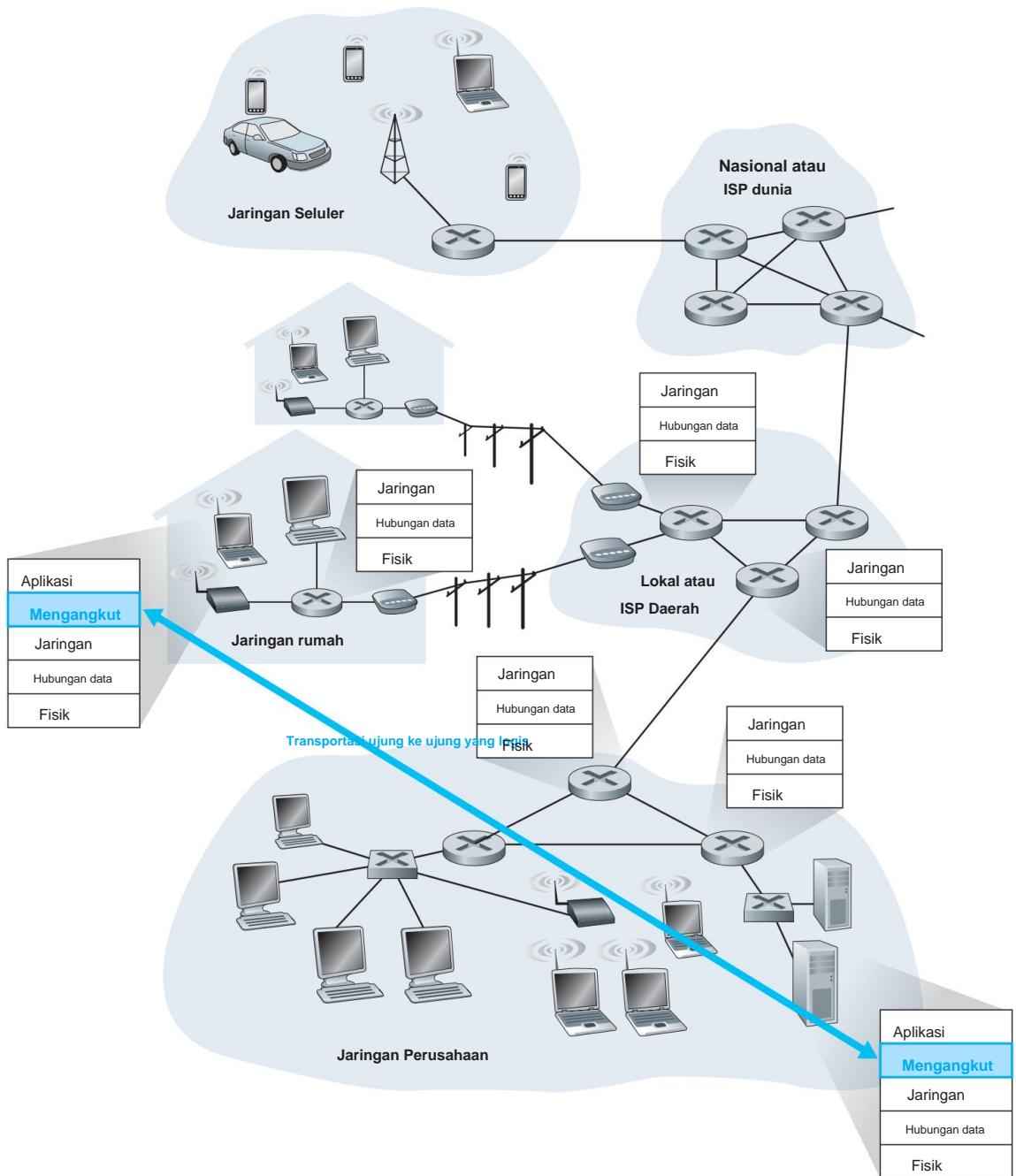
Lebih dari satu protokol transport-layer mungkin tersedia untuk aplikasi jaringan. Misalnya, Internet memiliki dua protokol—TCP dan UDP. Masing-masing protokol ini menyediakan satu set layanan transport-layer yang berbeda untuk aplikasi pemanggilan.

3.1.1 Hubungan Antara Lapisan Transport dan Jaringan

Ingatlah bahwa lapisan transport terletak tepat di atas lapisan jaringan dalam tumpukan protokol.

Sedangkan protokol lapisan transport menyediakan komunikasi logis antara proses yang berjalan pada host yang berbeda, protokol lapisan jaringan menyediakan logis

3.1 • PENDAHULUAN DAN JASA LAPISAN TRANSPORTASI 187



Gambar 3.1 Lapisan transport menyediakan logika daripada fisik komunikasi antar proses aplikasi

188 BAB 3 • LAPISAN TRANSPORTASI

komunikasi antar *host*. Perbedaan ini tidak kentara tetapi penting. Mari kita periksa perbedaan ini dengan bantuan analogi rumah tangga.

Pertimbangkan dua rumah, satu di Pantai Timur dan yang lainnya di Pantai Barat, dengan masing-masing rumah menjadi rumah bagi selusin anak. Anak-anak di rumah tangga Pantai Timur adalah sepupu dari anak-anak di rumah tangga Pantai Barat. Anak-anak di dua rumah tangga suka menulis satu sama lain — setiap anak menulis surat kepada sepupu setiap minggu, dengan setiap surat dikirimkan oleh layanan pos tradisional dalam amplop terpisah. Jadi, setiap rumah tangga mengirim 144 surat ke rumah tangga lainnya setiap minggu. (Anak-anak ini akan menghemat banyak uang jika mereka memiliki email!) Di setiap rumah ada satu anak—Ann di rumah West Coast dan Bill di rumah East Coast—bertanggung jawab atas pengumpulan surat dan distribusi surat. Setiap minggu Ann mengunjungi semua saudara laki-laki dan perempuannya, mengumpulkan surat, dan memberikan surat ke pembawa surat layanan pos, yang melakukan kunjungan harian ke rumah. Ketika surat tiba di rumah Pantai Barat, Ann juga bertugas mendistribusikan surat kepada saudara laki-laki dan perempuannya. Bill memiliki pekerjaan serupa di Pantai Timur.

Dalam contoh ini, layanan pos menyediakan komunikasi logis antara dua rumah—layanan pos memindahkan surat dari rumah ke rumah, bukan dari orang ke orang. Di sisi lain, Ann dan Bill menyediakan komunikasi yang logis di antara sepupu—Ann dan Bill mengambil surat dari, dan mengirimkan surat ke, saudara laki-laki dan perempuan mereka. Perhatikan bahwa dari sudut pandang sepupu, Ann dan Bill *adalah* layanan surat, meskipun Ann dan Bill hanyalah bagian (bagian sistem akhir) dari proses pengiriman ujung ke ujung. Contoh rumah tangga ini berfungsi sebagai analogi yang bagus untuk menjelaskan bagaimana lapisan transport berhubungan dengan lapisan jaringan:

pesan aplikasi = surat dalam amplop proses =
 sepupu host (juga disebut sistem akhir) = rumah
 protokol lapisan transportasi = protokol lapisan
 jaringan Ann dan Bill = layanan pos (termasuk
 operator surat)

Melanjutkan analogi ini, perhatikan bahwa Ann dan Bill melakukan semua pekerjaan mereka di rumah masing-masing; mereka tidak terlibat, misalkan, dalam menyortir surat di pusat surat perantara mana pun atau dalam memindahkan surat dari satu pusat surat ke pusat surat lainnya. Demikian pula, protokol transport-layer hidup di sistem akhir. Di dalam sistem akhir, protokol transport memindahkan pesan dari proses aplikasi ke tepi jaringan (yaitu, lapisan jaringan) dan sebaliknya, tetapi protokol ini tidak menentukan bagaimana pesan dipindahkan di dalam inti jaringan. Faktanya, seperti yang diilustrasikan pada Gambar 3.1, router perantara tidak bertindak, atau mengenali, informasi apa pun yang mungkin telah ditambahkan oleh lapisan transport ke pesan aplikasi.

Melanjutkan saga keluarga kita, misalkan sekarang ketika Ann dan Bill pergi berlibur, pasangan sepupu lainnya—katakanlah, Susan dan Harvey—menggantikan mereka dan menyediakan pengambilan dan pengiriman surat untuk keperluan rumah tangga. Sayangnya untuk kedua keluarga tersebut, Susan dan Harvey tidak melakukan pengambilan dan pengiriman dengan cara yang persis sama seperti Ann dan Bill. Menjadi anak-anak yang lebih muda, Susan dan Harvey lebih jarang mengambil dan mengirim surat dan kadang-kadang kehilangan surat (yang kadang-kadang

dikunyah oleh anjing keluarga). Jadi, pasangan sepupu Susan dan Harvey tidak menyediakan rangkaian layanan yang sama (yaitu, model layanan yang sama) seperti Ann dan Bill. Dengan cara yang analog, jaringan komputer dapat menyediakan beberapa protokol transportasi, dengan masing-masing protokol menawarkan model layanan yang berbeda untuk aplikasi.

Kemungkinan layanan yang dapat disediakan oleh Ann dan Bill secara jelas dibatasi oleh kemungkinan layanan yang disediakan oleh layanan pos. Misalnya, jika layanan pos tidak memberikan batasan maksimum berapa lama waktu yang diperlukan untuk mengirimkan surat antara dua rumah (misalnya, tiga hari), maka Ann dan Bill tidak dapat menjamin penundaan maksimum untuk pengiriman surat antara salah satu pasangan sepupu. Dengan cara yang sama, layanan yang dapat disediakan oleh protokol transport seringkali dibatasi oleh model layanan dari protokol lapisan jaringan yang mendasarinya. Jika protokol lapisan jaringan tidak dapat memberikan penundaan atau jaminan bandwidth untuk segmen lapisan transport yang dikirim antar host, maka protokol lapisan transport tidak dapat memberikan jaminan penundaan atau bandwidth untuk pesan aplikasi yang dikirim antar proses.

Namun demikian, layanan tertentu *dapat* ditawarkan oleh protokol transport bahkan ketika protokol jaringan yang mendasarinya tidak menawarkan layanan yang sesuai pada lapisan kerja jaringan. Misalnya, seperti yang akan kita lihat di bab ini, protokol transport dapat menawarkan layanan transfer data yang andal ke aplikasi bahkan ketika protokol jaringan yang mendasarinya tidak dapat diandalkan, bahkan ketika protokol jaringan kehilangan, mengacaukan, atau menggandakan paket. Sebagai contoh lain (yang akan kita jelajahi di Bab 8 ketika kita membahas keamanan jaringan), protokol transport dapat menggunakan enkripsi untuk menjamin bahwa pesan aplikasi tidak dibaca oleh penyusup, bahkan ketika lapisan jaringan tidak dapat menjamin kerahasiaan segmen lapisan transport. .

3.1.2 Tinjauan tentang Transport Layer di Internet

Ingat bahwa Internet, dan lebih umum jaringan TCP/IP, membuat dua protokol lapisan transport yang berbeda tersedia untuk lapisan aplikasi. Salah satu protokol ini adalah **UDP** (User Datagram Protocol), yang menyediakan layanan tanpa koneksi yang tidak dapat diandalkan untuk aplikasi pemanggil. Yang kedua dari protokol ini adalah **TCP** (Transmission Control Protocol), yang menyediakan layanan yang dapat diandalkan dan berorientasi koneksi ke aplikasi pemanggil. Saat mendesain aplikasi jaringan, pengembang aplikasi harus menentukan salah satu dari dua protokol transport ini. Seperti yang kita lihat di Bagian 2.7, pengembang aplikasi memilih antara UDP dan TCP saat membuat soket.

Untuk menyederhanakan terminologi, ketika dalam konteks Internet, kami menyebut paket lapisan transport sebagai segmen. Kami menyebutkan, bagaimanapun, bahwa literatur Internet (misalnya, RFC) juga mengacu pada paket transport-layer untuk TCP sebagai segmen tetapi sering mengacu pada paket untuk UDP sebagai datagram. Tetapi literatur Internet yang sama ini juga menggunakan istilah *datagram* untuk paket lapisan jaringan! Untuk buku pengantar tentang jaringan komputer seperti ini, kami percaya bahwa tidak terlalu membingungkan untuk menyebut paket TCP dan UDP sebagai segmen, dan mencadangkan istilah *datagram* untuk paket lapisan jaringan.

Sebelum melanjutkan pengenalan singkat kami tentang UDP dan TCP, akan berguna untuk mengatakan beberapa kata tentang lapisan jaringan Internet. (Kita akan mempelajari tentang lapisan kerja jaringan secara rinci di Bab 4.) Protokol lapisan jaringan Internet memiliki

190 BAB 3 • LAPISAN TRANSPORTASI

nama—IP, untuk Protokol Internet. IP menyediakan komunikasi logis antara host.

Model layanan IP adalah **layanan pengiriman upaya terbaik**. Ini berarti bahwa IP melakukan "upaya terbaik" untuk mengirimkan segmen antara host yang berkomunikasi, *tetapi tidak memberikan jaminan*. Secara khusus, ini tidak menjamin pengiriman segmen, tidak menjamin pengiriman segmen secara teratur, dan tidak menjamin integritas data dalam segmen. Untuk alasan ini, IP dikatakan sebagai **layanan yang tidak dapat diandalkan**. Kami juga menyebutkan di sini bahwa setiap host memiliki setidaknya satu alamat lapisan jaringan, yang disebut alamat IP. Kami akan memeriksa pengalamatan IP secara rinci di Bab 4; untuk bab ini kita hanya perlu mengingat bahwa *setiap host memiliki alamat IP*.

Setelah melihat sekilas model layanan IP, sekarang mari kita meringkas model layanan yang disediakan oleh UDP dan TCP. Tanggung jawab paling mendasar dari UDP dan TCP adalah memperluas layanan pengiriman IP antara dua sistem akhir menjadi layanan pengiriman antara dua proses yang berjalan di sistem akhir. Memperluas pengiriman host-ke-host ke pengiriman proses-ke-proses disebut **transport-layer multiplexing** dan **demultiplexing**. Kita akan membahas multiplexing dan demultiplexing transport-layer di bagian berikutnya. UDP dan TCP juga menyediakan pemeriksaan integritas dengan menyertakan bidang deteksi kesalahan di header segmennya. Dua layanan transport-layer minimal ini—pengiriman data proses-ke-proses dan pemeriksaan kesalahan—adalah satu-satunya dua layanan yang disediakan UDP! Secara khusus, seperti IP, UDP adalah layanan yang tidak dapat diandalkan—itu tidak menjamin bahwa data yang dikirim oleh satu proses akan tiba utuh (atau sama sekali!) ke proses tujuan. UDP dibahas secara rinci di Bagian 3.3.

TCP, di sisi lain, menawarkan beberapa layanan tambahan untuk aplikasi. Pertama dan terpenting, ini menyediakan **transfer data yang andal**. Dengan menggunakan flow control, sequence num bers, acknowledgment, dan timer (teknik yang akan kita telusuri secara detail dalam bab ini), TCP memastikan bahwa data dikirimkan dari proses pengiriman ke proses penerimaan, dengan benar dan teratur. TCP dengan demikian mengubah layanan IP yang tidak dapat diandalkan antara sistem akhir menjadi layanan transportasi data yang dapat diandalkan antar proses. TCP juga menyediakan **kontrol kemacetan**. Kontrol kemacetan bukanlah layanan yang disediakan untuk aplikasi pemanggil karena ini adalah layanan untuk Internet secara keseluruhan, layanan untuk kebaikan umum. Secara umum, kontrol kongesti TCP mencegah salah satu koneksi TCP membanjiri link dan router antara host yang berkomunikasi dengan jumlah lalu lintas yang berlebihan. TCP berusaha untuk memberikan setiap koneksi yang melintasi tautan padat bagian yang sama dari bandwidth tautan. Ini dilakukan dengan mengatur kecepatan di mana sisi pengirim koneksi TCP dapat mengirim lalu lintas ke jaringan.

Lalu lintas UDP, di sisi lain, tidak diatur. Aplikasi yang menggunakan transportasi UDP dapat mengirim dengan tarif berapa pun yang diinginkan, selama yang diinginkan.

Sebuah protokol yang menyediakan transfer data yang andal dan kontrol kemacetan sangatlah rumit. Kami memerlukan beberapa bagian untuk mencakup prinsip-prinsip transfer data yang andal dan kontrol kemacetan, dan bagian tambahan untuk mencakup protokol TCP itu sendiri. Topik-topik ini diselidiki dalam Bagian 3.4 sampai 3.8. Pendekatan yang diambil dalam bab ini adalah bergantian antara prinsip dasar dan protokol TCP. Misalnya, pertama-tama kita akan membahas transfer data yang andal dalam pengaturan umum dan kemudian membahas bagaimana TCP secara khusus menyediakan transfer data yang andal. Demikian pula, kami akan terlebih dahulu

diskusikan kontrol kemacetan dalam pengaturan umum dan kemudian diskusikan bagaimana TCP melakukan kontrol kemacetan. Namun sebelum masuk ke semua hal bagus ini, pertama-tama mari kita lihat multiplexing dan demultiplexing transport-layer.

3.2 Multiplexing dan Demultiplexing

Pada bagian ini, kita membahas transport-layer multiplexing dan demultiplexing, yaitu memperluas layanan pengiriman host-ke-host yang disediakan oleh lapisan jaringan ke layanan pengiriman proses-ke-proses untuk aplikasi yang berjalan di host. Agar diskusi tetap konkret, kita akan membahas layanan lapisan transportasi dasar ini dalam konteks Internet. Kami menekankan, bagaimanapun, bahwa layanan multiplexing/demultiplexing diperlukan untuk semua jaringan komputer.

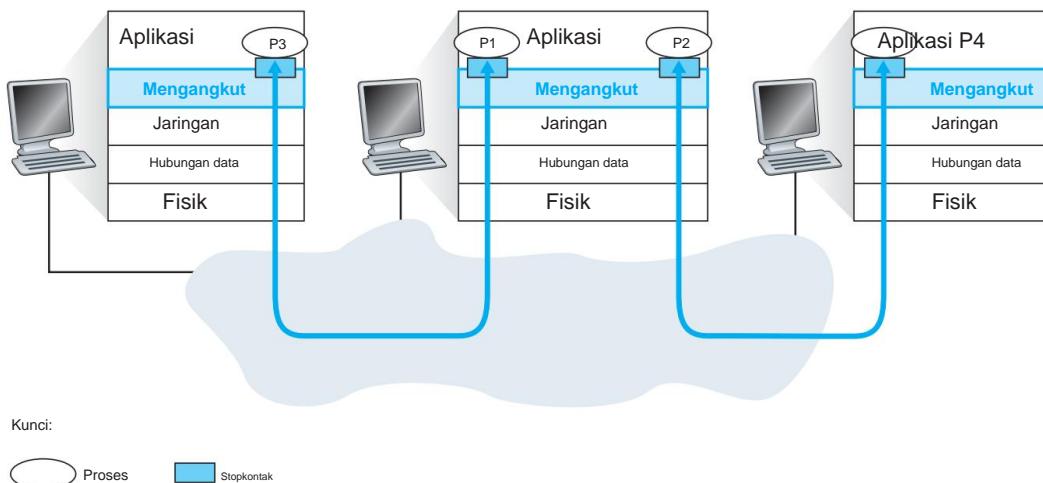
Di host tujuan, lapisan transport menerima segmen dari lapisan jaringan tepat di bawahnya. Lapisan transport bertanggung jawab mengirimkan data di segmen ini ke proses aplikasi yang sesuai yang berjalan di host. Mari kita lihat sebuah contoh. Misalkan Anda sedang duduk di depan komputer Anda, dan Anda mengunduh halaman Web sambil menjalankan satu sesi FTP dan dua sesi Telnet.

Oleh karena itu, Anda memiliki empat proses aplikasi jaringan yang berjalan—dua proses Telnet, satu proses FTP, dan satu proses HTTP. Saat lapisan transport di komputer Anda menerima data dari lapisan jaringan di bawahnya, lapisan ini perlu mengarahkan data yang diterima ke salah satu dari empat proses ini. Sekarang mari kita periksa bagaimana ini dilakukan.

Ingin pertama dari Bagian 2.7 bahwa suatu proses (sebagai bagian dari aplikasi jaringan) dapat memiliki satu atau lebih **soket**, pintu yang dilalui data dari jaringan ke proses dan melalui mana data lewat dari proses ke jaringan. Jadi, seperti yang ditunjukkan pada Gambar 3.2, lapisan transport di host penerima sebenarnya tidak mengirimkan data secara langsung ke proses, melainkan ke soket perantara. Karena pada waktu tertentu bisa terdapat lebih dari satu soket di host penerima, setiap soket memiliki pengidentifikasi unik. Format pengidentifikasi bergantung pada apakah soketnya adalah soket UDP atau TCP, seperti yang akan segera kita bahas.

Sekarang mari kita pertimbangkan bagaimana host penerima mengarahkan segmen transport-layer yang masuk ke soket yang sesuai. Setiap segmen transport-layer memiliki satu set bidang di segmen untuk tujuan ini. Di ujung penerima, lapisan transport memeriksa bidang-bidang ini untuk mengidentifikasi soket penerima dan kemudian mengarahkan segmen ke soket itu. Pekerjaan mengirimkan data dalam segmen transport-layer ke soket yang benar disebut **demultiplexing**. Pekerjaan mengumpulkan potongan data pada host sumber dari soket yang berbeda, merangkum setiap potongan data dengan informasi header (yang nantinya akan digunakan dalam demultiplexing) untuk membuat segmen, dan meneruskan segmen ke lapisan jaringan disebut multiplexing . Perhatikan bahwa lapisan transport pada host tengah pada Gambar 3.2 harus memisahkan segmen-segmen yang tiba dari lapisan jaringan di bawah untuk memproses P1 atau P2 di atas; ini dilakukan dengan mengarahkan data segmen yang datang ke soket proses yang sesuai. Lapisan transport di host tengah juga harus

192 BAB 3 • LAPISAN TRANSPORTASI

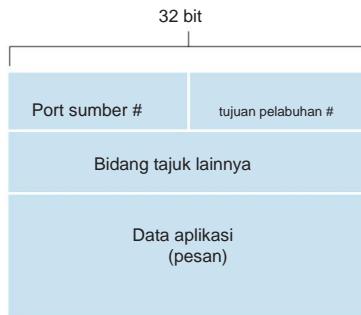


Gambar 3.2 Multiplexing dan demultiplexing transport-layer

mengumpulkan data keluar dari soket ini, membentuk segmen lapisan transportasi, dan meneruskan segmen ini ke lapisan jaringan. Meskipun kami telah memperkenalkan multiplexing dan demultiplexing dalam konteks protokol transport Internet, penting untuk menyadari bahwa mereka menjadi perhatian setiap kali satu protokol pada satu lapisan (pada lapisan transport atau di tempat lain) digunakan oleh banyak protokol pada lapisan berikutnya yang lebih tinggi. lapisan.

Untuk mengilustrasikan pekerjaan demultiplexing, ingat analogi rumah tangga di bagian sebelumnya. Setiap anak diidentifikasi dengan namanya. Ketika Bill menerima sekumpulan surat dari pembawa surat, dia melakukan operasi demultiplexing dengan mengamati kepada siapa surat-surat itu ditujukan dan kemudian menyerahkan surat itu kepada saudara laki-laki dan perempuannya. Ann melakukan operasi multiplexing ketika dia mengumpulkan surat dari saudara laki-laki dan perempuannya dan memberikan surat yang terkumpul ke petugas pos.

Sekarang setelah kita memahami peran transport-layer multiplexing dan demultiplexing, mari kita periksa bagaimana hal itu sebenarnya dilakukan di host. Dari pembahasan di atas, kita tahu bahwa transport-layer multiplexing mensyaratkan (1) soket memiliki pengidentifikasi unik, dan (2) setiap segmen memiliki bidang khusus yang menunjukkan soket tujuan pengiriman segmen. Bidang khusus ini, diilustrasikan pada Gambar 3.3, adalah **bidang nomor port sumber** dan **bidang nomor port tujuan**. (Segmen UDP dan TCP memiliki bidang lain juga, seperti yang dibahas di bagian selanjutnya dari bab ini.) Setiap nomor port adalah nomor 16-bit, mulai dari 0 hingga 65535. Nomor port mulai dari 0 hingga 1023 disebut **baik -nomor port yang dikenal** dan dibatasi, yang berarti bahwa mereka dicadangkan untuk digunakan oleh protokol aplikasi terkenal seperti HTTP (yang menggunakan nomor port 80) dan FTP (yang menggunakan nomor port 21). Daftar nomor port terkenal diberikan dalam RFC 1700 dan diperbarui di <http://www.iana.org> [RFC 3232]. Saat kami mengembangkan yang baru



Gambar 3.3 Bidang nomor port sumber dan tujuan dalam segmen lapisan transport

aplikasi (seperti aplikasi sederhana yang dikembangkan di Bagian 2.7), kita harus menetapkan nomor port untuk aplikasi tersebut.

Sekarang harus jelas bagaimana lapisan transport *dapat* mengimplementasikan layanan demultiplexing: Setiap soket di host dapat diberi nomor port, dan ketika segmen tiba di host, lapisan transport memeriksa nomor port tujuan di segmen dan mengarahkan segmen ke soket yang sesuai. Data segmen kemudian melewati soket ke dalam proses yang terpasang. Seperti yang akan kita lihat, pada dasarnya inilah cara UDP melakukannya. Namun, kita juga akan melihat bahwa multiplexing/demultiplexing di TCP lebih halus.

Multiplexing dan Demultiplexing Tanpa Koneksi

Ingat dari Bagian 2.7.1 bahwa program Python yang berjalan di host dapat membuat soket UDP dengan garis

```
clientSocket = soket(soket.AF_INET, soket.SOCK_DGRAM)
```

Saat soket UDP dibuat dengan cara ini, lapisan transport secara otomatis memberikan nomor port ke soket. Secara khusus, lapisan transport menetapkan nomor port dalam kisaran 1024 hingga 65535 yang saat ini tidak digunakan oleh port UDP lainnya di host. Alternatifnya, kita dapat menambahkan baris ke dalam program Python kita setelah kita membuat soket untuk mengasosiasikan nomor port tertentu (katakanlah, 19157) ke soket UDP ini melalui metode socket bind() :

```
clientSocket.bind(("19157"))
```

Jika pengembang aplikasi yang menulis kode mengimplementasikan sisi server dari "protokol terkenal", maka pengembang harus menetapkan yang sesuai

194 BAB 3 • LAPISAN TRANSPORTASI

nomor port terkenal. Biasanya, sisi klien aplikasi memungkinkan lapisan port trans secara otomatis (dan transparan) menetapkan nomor port, sedangkan sisi server aplikasi menetapkan nomor port tertentu.

Dengan nomor port yang ditetapkan ke soket UDP, kami sekarang dapat menjelaskan multiplexing/demultiplexing UDP dengan tepat. Misalkan sebuah proses di Host A, dengan port UDP 19157, ingin mengirim potongan data aplikasi ke proses dengan port UDP 46428 di Host B. Lapisan transport di Host A membuat segmen lapisan transport yang menyertakan data aplikasi, nomor port sumber (19157), nomor port tujuan (46428), dan dua nilai lainnya (yang akan dibahas nanti, tetapi penting untuk pembahasan saat ini). Lapisan transport kemudian melewati segmen yang dihasilkan ke lapisan jaringan. Lapisan jaringan mengenkapsulasi segmen dalam datagram IP dan melakukan upaya terbaik untuk mengirimkan segmen ke host penerima. Jika segmen tiba di Host B penerima, lapisan transport di host penerima memeriksa nomor port tujuan di segmen (46428) dan mengirimkan segmen ke soketnya yang diidentifikasi oleh port 46428. Perhatikan bahwa Host B dapat menjalankan banyak proses, masing-masing dengan soket UDP sendiri dan nomor port terkait. Saat segmen UDP tiba dari jaringan, Host B mengarahkan (demultiplex) setiap segmen ke soket yang sesuai dengan memeriksa nomor port tujuan segmen tersebut.

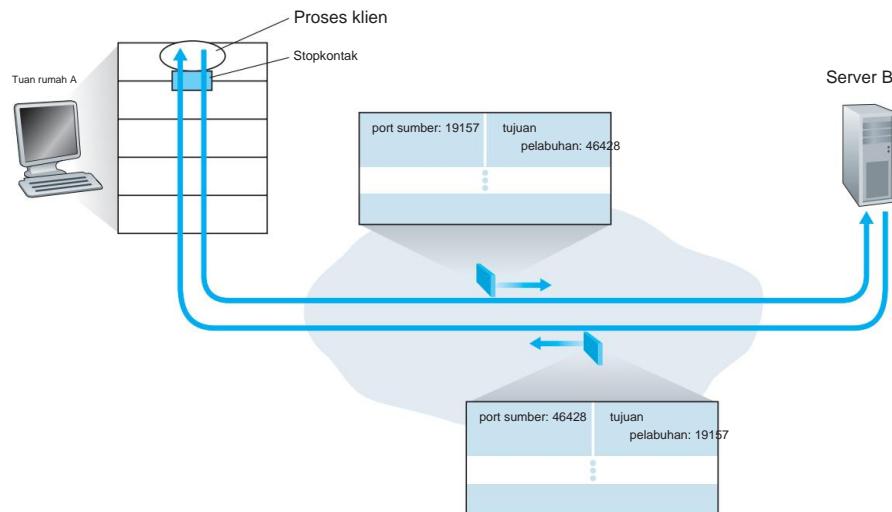
Penting untuk dicatat bahwa soket UDP sepenuhnya diidentifikasi oleh dua tuple yang terdiri dari alamat IP tujuan dan nomor port tujuan. Akibatnya, jika dua segmen UDP memiliki alamat IP sumber dan/atau nomor port sumber yang berbeda, tetapi memiliki alamat IP *tujuan* dan nomor port *tujuan* yang sama , maka kedua segmen tersebut akan diarahkan ke proses tujuan yang sama melalui soket tujuan yang sama.

Anda mungkin bertanya-tanya sekarang, apa tujuan dari nomor port sumber? Seperti yang ditunjukkan pada Gambar 3.4, di segmen A-ke-B, nomor port sumber berfungsi sebagai bagian dari "alamat pengirim"—ketika B ingin mengirim segmen kembali ke A, port tujuan di segmen B-ke-A akan mengambil nilainya dari nilai port sumber segmen A-ke-B. (Alamat pengirim yang lengkap adalah alamat IP A dan nomor port sumber.) Sebagai contoh, ingat program server UDP yang dipelajari di Bagian 2.7. Di UDPServer.py, server menggunakan metode `recvfrom()` untuk mengekstrak nomor port (sumber) sisi klien dari segmen yang diterimanya dari klien; itu kemudian mengirimkan segmen baru ke klien, dengan nomor port sumber yang diekstraksi berfungsi sebagai nomor port tujuan di segmen baru ini.

Multiplexing dan Demultiplexing Berorientasi Koneksi

Untuk memahami demultiplexing TCP, kita harus melihat dari dekat soket TCP dan pembentukan koneksi TCP. Satu perbedaan halus antara soket TCP dan soket UDP adalah bahwa soket TCP diidentifikasi oleh empat tupel: (alamat IP sumber, nomor port sumber, alamat IP tujuan, nomor port tujuan).

Jadi, ketika segmen TCP datang dari jaringan ke host, host menggunakan keempat nilai untuk mengarahkan (demultiplex) segmen ke soket yang sesuai. Secara khusus, dan berbeda dengan UDP, dua segmen TCP tiba dengan IP sumber yang berbeda



Gambar 3.4 Inversi nomor port sumber dan tujuan

alamat atau nomor port sumber akan (dengan pengecualian segmen TCP yang membawa permintaan pembuatan koneksi asli) diarahkan ke dua soket yang berbeda. Untuk mendapatkan wawasan lebih lanjut, mari pertimbangkan kembali contoh pemrograman client-server TCP di Bagian 2.7.2:

- Aplikasi server TCP memiliki "soket penyambutan", yang menunggu permintaan pembentukan koneksi dari klien TCP (lihat Gambar 2.29) pada nomor port 12000. •

Klien TCP membuat soket dan mengirimkan segmen permintaan pembentukan koneksi dengan baris:

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,12000))
```

- Permintaan pembuatan sambungan tidak lebih dari segmen TCP dengan nomor port tujuan 12000 dan bit pembuatan sambungan khusus yang diatur dalam header TCP (dibahas di Bagian 3.5). Segmen tersebut juga menyertakan nomor port sumber yang dipilih oleh klien.
- Ketika sistem operasi host dari komputer yang menjalankan proses server menerima segmen permintaan sambungan masuk dengan port tujuan 12000, ia menempatkan proses server yang sedang menunggu untuk menerima sambungan pada nomor port 12000. Proses server kemudian membuat yang baru stopkontak:

```
connectionSocket, addr = serverSocket.accept()
```

196 BAB 3 • LAPISAN TRANSPORTASI

- Juga, lapisan transport pada server mencatat empat nilai berikut dalam segmen permintaan koneksi: (1) nomor port sumber di segmen, (2) alamat IP dari host sumber, (3) port tujuan nomor di segmen, dan (4) alamat IP-nya sendiri. Soket koneksi yang baru dibuat diidentifikasi oleh empat nilai ini; semua segmen yang tiba selanjutnya yang port sumber, alamat IP sumber, port tujuan, dan alamat IP tujuan cocok dengan keempat nilai ini akan didemultipleks ke soket ini. Dengan koneksi TCP sekarang, klien dan server sekarang dapat mengirim data satu sama lain.

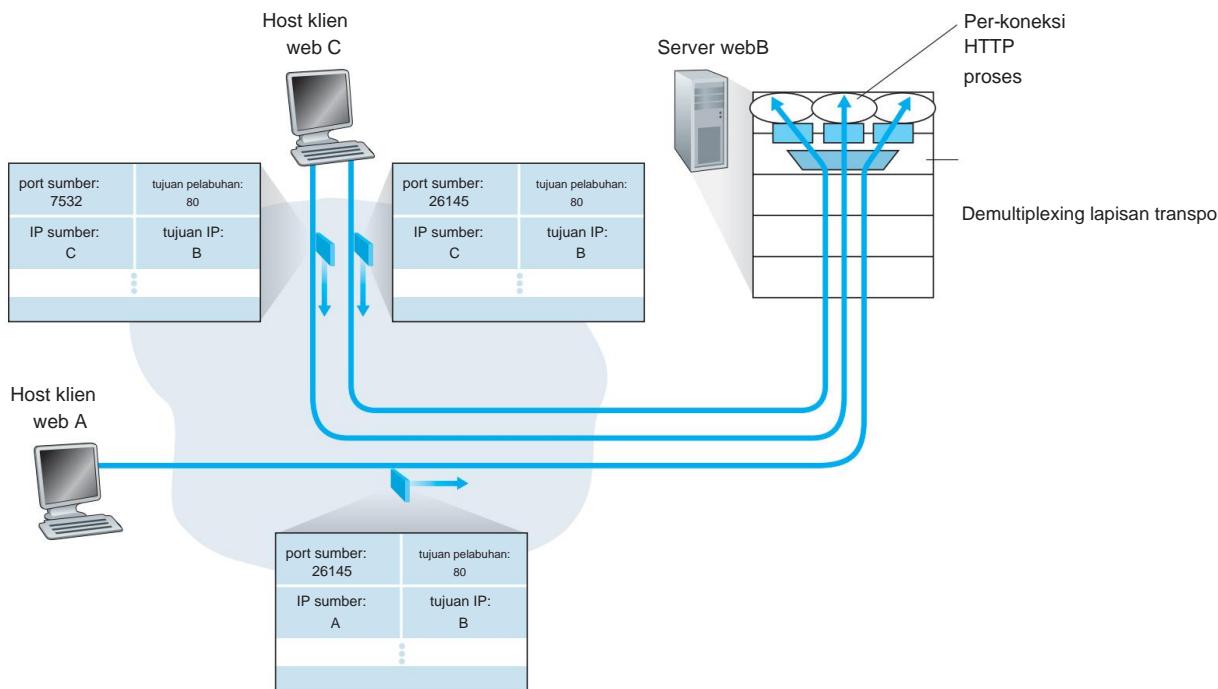
Tuan rumah server dapat mendukung banyak soket koneksi TCP secara bersamaan, dengan setiap soket terpasang pada suatu proses, dan dengan setiap soket diidentifikasi oleh empat tupelnya sendiri. Ketika segmen TCP tiba di host, keempat bidang (alamat IP sumber, port sumber, alamat IP tujuan, port tujuan) digunakan untuk mengarahkan (demultiplex) segmen ke soket yang sesuai.

FOKUS PADA KEAMANAN

PEMINDAIAN PELABUHAN

Kami telah melihat bahwa proses server menunggu dengan sabar di port terbuka untuk dihubungi oleh klien jarak jauh. Beberapa port dicadangkan untuk aplikasi terkenal (misalnya, Web, FTP, DNS, dan server SMTP); port lain digunakan oleh konvensi oleh aplikasi populer (misalnya, server SQL Microsoft 2000 mendengarkan permintaan pada port UDP 1434). Jadi, jika kami menentukan bahwa sebuah port terbuka di host, kami mungkin dapat memetakan port tersebut ke aplikasi tertentu yang berjalan di host. Ini sangat berguna untuk administrator sistem, yang sering tertarik untuk mengetahui aplikasi jaringan mana yang berjalan di host di jaringan mereka. Tetapi penyerang, untuk "menyelesaikan masalah bersama", juga ingin mengetahui port mana yang terbuka di host target. Jika sebuah host ditemukan menjalankan aplikasi dengan kelemahan keamanan yang diketahui (misalnya, server SQL yang mendengarkan pada port 1434 terkena buffer overflow, memungkinkan pengguna jarak jauh untuk mengeksekusi kode arbitrer pada host yang rentan, sebuah kelemahan yang dieksloitasi oleh Slammer worm [CERT 2003–04]), maka inang itu siap untuk diserang.

Menentukan aplikasi mana yang mendengarkan port mana adalah tugas yang relatif mudah. Memang ada sejumlah program domain publik, yang disebut pemindai port, yang melakukan hal itu. Mungkin yang paling banyak digunakan adalah nmap, tersedia secara gratis di <http://nmap.org> dan disertakan di sebagian besar distribusi Linux. Untuk TCP, nmap secara berurutan memindai port, mencari port yang menerima koneksi TCP. Untuk UDP, nmap kembali memindai port secara berurutan, mencari port UDP yang merespons segmen UDP yang ditransmisikan. Dalam kedua kasus tersebut, nmap mengembalikan daftar port terbuka, tertutup, atau tidak terjangkau. Sebuah host yang menjalankan nmap dapat mencoba memindai setiap host target di Internet. Kita akan mengunjungi kembali nmap di Bagian 3.5.6, saat kita membahas manajemen koneksi TCP. di mana saja



Gambar 3.5 Dua klien, menggunakan nomor port tujuan yang sama (80) untuk berkomunikasi dengan aplikasi server Web yang sama

Situasi diilustrasikan pada Gambar 3.5, di mana Host C memulai dua sesi HTTP ke server B, dan Host A memulai satu sesi HTTP ke B. Host A dan C dan server B masing-masing memiliki alamat IP unik mereka sendiri—A, C, dan B, masing-masing. Host C menetapkan dua nomor port sumber yang berbeda (26145 dan 7532) ke dua koneksi HTTP-nya.

Karena Host A memilih nomor port sumber secara terpisah dari C, Host A mungkin juga menetapkan port sumber 26145 ke koneksi HTTP-nya. Namun ini bukan masalah—server B akan tetap dapat melakukan demultiplex dengan benar pada dua koneksi yang memiliki nomor port sumber yang sama, karena kedua koneksi tersebut memiliki IP sumber yang berbeda.

Server Web dan TCP

Sebelum menutup diskusi ini, ada baiknya untuk mengatakan beberapa kata tambahan tentang server Web dan bagaimana mereka menggunakan nomor port. Pertimbangkan host yang menjalankan server Web, seperti server Web Apache, pada port 80. Ketika klien (misalnya, browser) mengirim segmen ke server, semua segmen *akan* memiliki port tujuan 80. Khususnya, baik koneksi awal- segmen pendirian dan segmen yang membawa pesan permintaan HTTP akan memiliki port tujuan 80. Seperti yang baru saja kami jelaskan,

198 BAB 3 • LAPISAN TRANSPORTASI

server membedakan segmen dari klien yang berbeda menggunakan alamat IP sumber dan nomor port sumber.

Gambar 3.5 menunjukkan server Web yang memunculkan proses baru untuk setiap koneksi. Seperti yang ditunjukkan pada Gambar 3.5, masing-masing proses ini memiliki soket koneksinya sendiri di mana permintaan HTTP datang dan respons HTTP dikirim. Kami menyebutkan, bagaimanapun, bahwa tidak selalu ada korespondensi satu-ke-satu antara soket koneksi dan proses. Faktanya, server Web berkinerja tinggi saat ini sering hanya menggunakan satu proses, dan membuat utas baru dengan soket koneksi baru untuk setiap koneksi klien baru. (Utas dapat dilihat sebagai subproses yang ringan.) Jika Anda melakukan tugas pemrograman pertama di Bab 2, Anda membuat server Web yang melakukan hal ini. Untuk server seperti itu, pada waktu tertentu mungkin ada banyak soket koneksi (dengan pengidentifikasi berbeda) yang dilampirkan ke proses yang sama.

Jika klien dan server menggunakan HTTP persisten, maka sepanjang durasi koneksi persisten, klien dan server bertukar pesan HTTP melalui soket server yang sama. Namun, jika klien dan server menggunakan HTTP non-persistent, maka koneksi TCP baru dibuat dan ditutup untuk setiap permintaan/respons, dan karenanya soket baru dibuat dan kemudian ditutup untuk setiap permintaan/respons. Seringnya pembuatan dan penutupan soket ini dapat sangat memengaruhi kinerja server Web yang sibuk (walaupun sejumlah trik sistem operasi dapat digunakan untuk mengurangi masalah). Pembaca yang tertarik dengan masalah sistem operasi seputar HTTP persistent dan non-persistent dianjurkan untuk melihat [Nielsen 1997; Nahum 2002].

Sekarang kita telah membahas multiplexing dan demultiplexing transport-layer, mari kita lanjutkan dan diskusikan salah satu protokol transport Internet, UDP. Pada bagian selanjutnya kita akan melihat bahwa UDP menambahkan sedikit lebih banyak ke protokol lapisan jaringan daripada layanan multiplexing/demultiplexing.

3.3 Transportasi Tanpa Koneksi: UDP

Pada bagian ini, kita akan mencermati UDP, cara kerjanya, dan apa fungsinya.

Kami mendorong Anda untuk merujuk kembali ke Bagian 2.1, yang mencakup ikhtisar model layanan UDP, dan Bagian 2.7.1, yang membahas pembuatan program soket menggunakan UDP.

Untuk memotivasi diskusi kita tentang UDP, misalkan Anda tertarik untuk merancang protokol transport sederhana dan tanpa embel-embel. Bagaimana Anda bisa melakukan ini? Pertama-tama Anda mungkin mempertimbangkan untuk menggunakan protokol transport kosong. Secara khusus, di sisi pengiriman, Anda dapat mempertimbangkan untuk mengambil pesan dari proses aplikasi dan meneruskannya langsung ke lapisan jaringan; dan di sisi penerima, Anda dapat mempertimbangkan untuk mengambil pesan yang datang dari lapisan jaringan dan meneruskannya langsung ke proses aplikasi. Tapi seperti yang kita pelajari di bagian sebelumnya, kita harus melakukan lebih dari tidak sama sekali! Paling tidak, lapisan transport harus

3.3 • TRANSPORTASI TANPA KONEKSI: UDP 199

menyediakan layanan multiplexing/demultiplexing untuk meneruskan data antara lapisan jaringan dan proses tingkat aplikasi yang benar.

UDP, didefinisikan dalam [RFC 768], hanya melakukan sesedikit yang dapat dilakukan oleh protokol transport. Selain dari fungsi multiplexing/demultiplexing dan beberapa pemeriksaan kesalahan ringan, itu tidak menambahkan apa pun ke IP. Faktanya, jika pengembang aplikasi memilih UDP daripada TCP, maka aplikasi tersebut hampir langsung berbicara dengan IP. UDP mengambil pesan dari proses aplikasi, melampirkan bidang nomor port sumber dan tujuan untuk layanan multiplexing/demultiplexing, menambahkan dua bidang kecil lainnya, dan meneruskan segmen yang dihasilkan ke lapisan jaringan. Lapisan jaringan merangkum segmen transport ke dalam datagram IP dan kemudian melakukan upaya terbaik untuk mengirimkan segmen ke host penerima. Jika segmen tiba di host penerima, UDP menggunakan nomor port tujuan untuk mengirimkan data segmen ke proses aplikasi yang benar. Perhatikan bahwa dengan UDP tidak ada handshaking antara mengirim dan menerima entitas transport-layer sebelum mengirim segmen. Untuk alasan ini, UDP dikatakan *tanpa koneksi*.

DNS adalah contoh protokol lapisan aplikasi yang biasanya menggunakan UDP.

Ketika aplikasi DNS di host ingin membuat kueri, aplikasi tersebut membuat pesan kueri DNS dan meneruskan pesan tersebut ke UDP. Tanpa melakukan jabat tangan apa pun dengan entitas UDP yang berjalan pada sistem akhir tujuan, UDP sisi host menambahkan kolom header ke pesan dan meneruskan segmen yang dihasilkan ke lapisan jaringan. Lapisan jaringan merangkum segmen UDP ke dalam datagram dan mengirimkan datagram ke server nama. Aplikasi DNS pada host kueri kemudian menunggu jawaban atas kuerinya. Jika tidak menerima balasan (mungkin karena jaringan yang mendasarinya kehilangan kueri atau balasan), baik itu mencoba mengirim kueri ke server nama lain, atau memberi tahu aplikasi pemanggil bahwa aplikasi tidak bisa mendapatkan balasan.

Sekarang Anda mungkin bertanya-tanya mengapa pengembang aplikasi memilih untuk membangun aplikasi melalui UDP daripada melalui TCP. Bukankah TCP selalu lebih disukai, karena TCP menyediakan layanan transfer data yang andal, sedangkan UDP tidak?

Jawabannya adalah tidak, karena banyak aplikasi yang lebih cocok untuk UDP karena alasan berikut:

- *Kontrol tingkat aplikasi yang lebih baik atas data apa yang dikirim, dan kapan*. Di bawah UDP, segera setelah proses aplikasi meneruskan data ke UDP, UDP akan mengemas data di dalam segmen UDP dan segera meneruskan segmen tersebut ke lapisan jaringan. TCP, di sisi lain, memiliki mekanisme kontrol kongesti yang menghambat pengiriman TCP lapisan transport ketika satu atau lebih tautan antara host sumber dan tujuan menjadi terlalu padat. TCP juga akan terus mengirim ulang segmen sampai penerimaan segmen telah diakui oleh tujuan, terlepas dari berapa lama waktu pengiriman yang dapat diandalkan. Karena aplikasi real-time sering memerlukan tingkat pengiriman minimum, tidak ingin terlalu menunda transmisi segmen, dan dapat mentolerir beberapa kehilangan data, model layanan TCP tidak terlalu cocok dengan kebutuhan aplikasi ini. Seperti dibahas di bawah, aplikasi ini dapat menggunakan UDP dan mengimplementasikan, sebagai bagian dari aplikasi, fungsionalitas tambahan apa pun yang diperlukan di luar layanan pengiriman segmen tanpa embel-embel UDP.

200 BAB 3 • LAPISAN TRANSPORTASI

- *Tidak ada pembentukan koneksi.* Seperti yang akan kita bahas nanti, TCP menggunakan jabat tangan tiga arah sebelum mulai mentransfer data. UDP meledak begitu saja tanpa persiapan formal. Jadi UDP tidak menimbulkan penundaan apa pun untuk membuat koneksi. Ini mungkin alasan utama mengapa DNS berjalan di atas UDP daripada TCP—DNS akan jauh lebih lambat jika dijalankan di atas TCP. HTTP menggunakan TCP daripada UDP, karena keandalan sangat penting untuk halaman Web dengan teks. Namun, seperti yang telah kita bahas secara singkat di Bagian 2.2, penundaan pembentukan koneksi TCP di HTTP merupakan kontributor penting untuk penundaan terkait dengan pengunduhan dokumen Web.
- *Tidak ada status koneksi.* TCP mempertahankan status koneksi di sistem akhir. Status koneksi ini mencakup buffer terima dan kirim, parameter kontrol kongesti, dan parameter nomor urutan dan pengakuan. Kita akan melihat di Bagian 3.5 bahwa informasi status ini diperlukan untuk mengimplementasikan layanan transfer data TCP yang andal dan untuk menyediakan kontrol kongesti. UDP, di sisi lain, tidak mempertahankan status koneksi dan tidak melacak salah satu dari parameter ini. Untuk alasan ini, server yang dikhususkan untuk aplikasi tertentu biasanya dapat mendukung lebih banyak klien aktif saat aplikasi dijalankan melalui UDP daripada TCP. •
Overhead header paket kecil. Segmen TCP memiliki 20 byte header di atas kepala di setiap segmen, sedangkan UDP hanya memiliki 8 byte overhead.

Gambar 3.6 daftar aplikasi Internet populer dan protokol transport yang mereka gunakan. Seperti yang kita harapkan, e-mail, akses terminal jarak jauh, Web, dan transfer file dijalankan melalui TCP—semua aplikasi ini memerlukan layanan transfer data TCP yang andal. Namun demikian, banyak aplikasi penting berjalan di atas UDP daripada TCP. UDP digunakan untuk update tabel routing RIP (lihat Bagian 4.6.1). Karena pembaruan RIP dikirim secara berkala (biasanya setiap lima menit), pembaruan yang hilang akan digantikan oleh pembaruan yang lebih baru, sehingga pembaruan yang hilang dan kedaluwarsa menjadi tidak berguna. UDP juga digunakan untuk membawa data manajemen jaringan (SNMP; lihat Bab 9). UDP lebih disukai daripada TCP dalam hal ini, karena aplikasi manajemen jaringan harus sering dijalankan saat jaringan berada dalam kondisi tertekan—tepatnya saat transfer data yang dapat diandalkan dan dikendalikan kemacetan sulit dicapai. Juga, seperti yang kami sebutkan sebelumnya, DNS berjalan di atas UDP, sehingga menghindari penundaan pembentukan koneksi TCP.

Seperti yang ditunjukkan pada Gambar 3.6, baik UDP maupun TCP digunakan saat ini dengan aplikasi multimedia, seperti telepon Internet, konferensi video real-time, dan streaming audio dan video yang disimpan. Kami akan mencermati aplikasi ini di Bab 7. Kami baru saja menyebutkan bahwa semua aplikasi ini dapat mentolerir sejumlah kecil kehilangan paket, sehingga transfer data yang andal tidak mutlak penting untuk keberhasilan aplikasi. Selain itu, aplikasi real-time, seperti telepon Internet dan konferensi video, bereaksi sangat buruk terhadap kontrol kongesti TCP. Untuk alasan ini, pengembang aplikasi multimedia dapat memilih untuk menjalankan aplikasinya melalui UDP daripada TCP. Namun, TCP semakin banyak digunakan untuk transportasi media streaming. Misalnya, [Sripanidkulchai 2004] menemukan bahwa hampir 75% on-demand dan live streaming menggunakan TCP. Ketika tingkat kehilangan paket rendah, dan dengan beberapa organisasi

3.3 • TRANSPORTASI TANPA KONEKSI: UDP 201

Aplikasi	Aplikasi-Lapisan Protokol	Transportasi Dasar Protokol
Surat elektronik	SMTP	TCP
Akses terminal jarak jauh	Telnet	TCP
Web	HTTP	TCP
Transfer file	FTP	TCP
Server file jarak jauh	NFS	Biasanya UDP
Streaming multimedia	biasanya berpemilik	UDP atau TCP
telepon internet	biasanya berpemilik	UDP atau TCP
Manajemen jaringan	SNMP	Biasanya UDP
Protokol perutean	---	Biasanya UDP
Terjemahan nama	DNS	Biasanya UDP

Gambar 3.6 Aplikasi Internet populer dan protokol transport yang mendasarinya

memblokir lalu lintas UDP untuk alasan keamanan (lihat Bab 8), TCP menjadi protokol yang semakin menarik untuk transportasi media streaming.

Meski sudah umum dilakukan saat ini, menjalankan aplikasi multimedia melalui UDP masih kontroversial. Seperti yang kami sebutkan di atas, UDP tidak memiliki kontrol kemacetan. Tetapi kontrol kemacetan diperlukan untuk mencegah jaringan memasuki keadaan padat di mana sangat sedikit pekerjaan yang berguna dilakukan. Jika semua orang memulai streaming video kecepatan bit tinggi tanpa menggunakan kontrol kongesti, akan ada begitu banyak paket yang meluap di router sehingga sangat sedikit paket UDP yang berhasil melintasi jalur sumber ke tujuan. Selain itu, tingkat kerugian tinggi yang disebabkan oleh pengirim UDP yang tidak terkendali akan menyebabkan pengirim TCP (yang, seperti yang akan kita lihat, *menurunkan* tingkat pengiriman mereka dalam menghadapi kemacetan) secara dramatis menurunkan tingkat mereka.

Dengan demikian, kurangnya kontrol kongesti di UDP dapat mengakibatkan tingkat kerugian yang tinggi antara pengirim dan penerima UDP, dan crowding out dari sesi TCP—masalah yang berpotensi serius [Floyd 1999]. Banyak peneliti telah mengusulkan mekanisme baru untuk memaksa semua sumber, termasuk sumber UDP, untuk melakukan kontrol kongesti adaptif [Mahdavi 1997; Floyd 2000; Kohler 2006: RFC 4340].

Sebelum membahas struktur segmen UDP, kami menyebutkan bahwa aplikasi dapat memiliki transfer data yang andal saat menggunakan UDP. Hal ini dapat dilakukan jika keandalan dibangun ke dalam aplikasi itu sendiri (misalnya, dengan menambahkan mekanisme pengakuan dan pengiriman ulang, seperti yang akan kita pelajari di bagian berikutnya). Tapi ini adalah tugas nontrivial yang akan membuat pengembangan aplikasi sibuk melakukan debug.

202 BAB 3 • LAPISAN TRANSPORTASI

waktu yang lama. Namun demikian, membangun keandalan secara langsung ke dalam aplikasi memungkinkan aplikasi untuk “mendapatkan kuenya dan memakannya juga”. Artinya, proses aplikasi dapat berkomunikasi dengan andal tanpa tunduk pada batasan laju transmisi yang dipaksakan oleh mekanisme kontrol kongesti TCP.

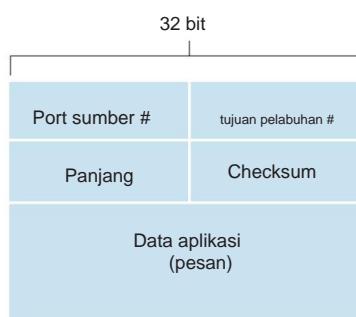
3.3.1 Struktur Segmen UDP

Struktur segmen UDP, ditunjukkan pada Gambar 3.7, didefinisikan dalam RFC 768. Data aplikasi menempati bidang data segmen UDP. Misalnya, untuk DNS, bidang data berisi pesan kueri atau pesan respons. Untuk aplikasi audio streaming, sampel audio mengisi kolom data. Header UDP hanya memiliki empat bidang, masing-masing terdiri dari dua byte. Seperti dibahas di bagian sebelumnya, nomor port memungkinkan host tujuan untuk meneruskan data aplikasi ke proses yang benar yang berjalan di sistem akhir tujuan (yaitu, untuk melakukan fungsi demultiplexing). Bidang panjang menentukan jumlah byte di segmen UDP (header plus data). Nilai panjang eksplisit diperlukan karena ukuran bidang data mungkin berbeda dari satu segmen UDP ke segmen berikutnya. Checksum digunakan oleh host penerima untuk memeriksa apakah kesalahan telah dimasukkan ke dalam segmen. Sebenarnya, checksum juga dihitung pada beberapa bidang di header IP selain segmen UDP. Tapi kami mengabaikan detail ini untuk melihat hutan melalui pepohonan. Kami akan membahas perhitungan checksum di bawah ini. Prinsip dasar deteksi kesalahan dijelaskan di Bagian 5.2. Bidang panjang menentukan panjang segmen UDP, termasuk header, dalam byte.

3.3.2 Pemeriksaan UDP

Checksum UDP menyediakan deteksi kesalahan. Artinya, checksum digunakan untuk menentukan apakah bit di dalam segmen UDP telah diubah (misalnya, oleh noise di tautan atau saat disimpan di router) saat dipindahkan dari sumber ke tujuan.

UDP di sisi pengirim melakukan komplemen 1s dari jumlah semua kata 16-bit di segmen, dengan luapan apa pun yang ditemui selama penjumlahan



Gambar 3.7 Struktur segmen UDP

3.3 • TRANSPORTASI TANPA KONEKSI: UDP 203

dibungkus. Hasil ini dimasukkan ke dalam bidang checksum segmen UDP. Berikut kami berikan contoh sederhana perhitungan checksum. Anda dapat menemukan detail tentang implementasi perhitungan yang efisien di RFC 1071 dan kinerja atas data nyata di [Stone 1998; Batu 2000]. Sebagai contoh, misalkan kita memiliki tiga kata 16-bit berikut:

```
0110011001100000
0101010101010101
1000111100001100
```

Jumlah dari dua pertama dari kata-kata 16-bit ini adalah

```
0110011001100000
0101010101010101
1011101110110101
```

Menambahkan kata ketiga ke jumlah di atas memberi

```
1011101110110101
1000111100001100
01001010101000010
```

Perhatikan bahwa penambahan terakhir ini memiliki luapan, yang melilit. Pelengkap 1s diperoleh dengan mengubah semua 0s menjadi 1s dan mengubah semua 1s menjadi 0s.

Jadi komplemen 1 dari penjumlahan 0100101011000010 adalah 1011010100111101, yang menjadi checksum. Di penerima, keempat kata 16-bit ditambahkan, termasuk checksum. Jika tidak ada kesalahan yang dimasukkan ke dalam paket, maka jelas jumlah pada penerima adalah 1111111111111111. Jika salah satu bit adalah 0, maka kita tahu bahwa kesalahan telah dimasukkan ke dalam paket.

Anda mungkin bertanya-tanya mengapa UDP menyediakan checksum sejak awal, karena banyak protokol lapisan tautan (termasuk protokol Ethernet yang populer) juga menyediakan pemeriksaan kesalahan. Alasannya adalah tidak ada jaminan bahwa semua tautan antara sumber dan tujuan memberikan pemeriksaan kesalahan; yaitu, salah satu tautan mungkin menggunakan protokol lapisan tautan yang tidak menyediakan pemeriksaan kesalahan. Selain itu, bahkan jika segmen ditransfer dengan benar melalui tautan, kesalahan bit dapat terjadi saat segmen disimpan dalam memori router. Mengingat bahwa keandalan link-by-link atau deteksi kesalahan dalam memori tidak dijamin, UDP harus menyediakan deteksi kesalahan pada lapisan transport, *pada basis ujung-ujung*, jika layanan transfer data ujung-ujung menyediakan deteksi kesalahan. Ini adalah contoh dari **prinsip ujung-ujung** yang terkenal dalam desain sistem [Saltzer 1984], yang menyatakan bahwa karena fungsionalitas tertentu (deteksi kesalahan, dalam hal ini harus diimplementasikan pada basis ujung-ujung: "fungsi ditempatkan di tingkat yang lebih rendah mungkin berlebihan atau nilainya kecil jika dibandingkan dengan biaya untuk menyediakannya di tingkat yang lebih tinggi."

Karena IP seharusnya berjalan di atas hampir semua protokol layer-2, berguna bagi layer transport untuk menyediakan pemeriksaan kesalahan sebagai langkah keamanan. Meskipun UDP

204 BAB 3 • LAPISAN TRANSPORTASI

menyediakan pemeriksaan kesalahan, itu tidak melakukan apa pun untuk pulih dari kesalahan. Beberapa implementasi UDP hanya membuang segmen yang rusak; yang lain meneruskan segmen yang rusak ke aplikasi dengan peringatan.

Itu mengakhiri diskusi kita tentang UDP. Kami akan segera melihat bahwa TCP menawarkan transfer data yang andal ke aplikasinya serta layanan lain yang tidak ditawarkan UDP. Tentu saja, TCP juga lebih kompleks daripada UDP. Namun, sebelum membahas TCP, akan berguna untuk mundur dan pertama-tama membahas prinsip dasar transfer data yang andal.

3.4 Prinsip Transfer Data yang Andal

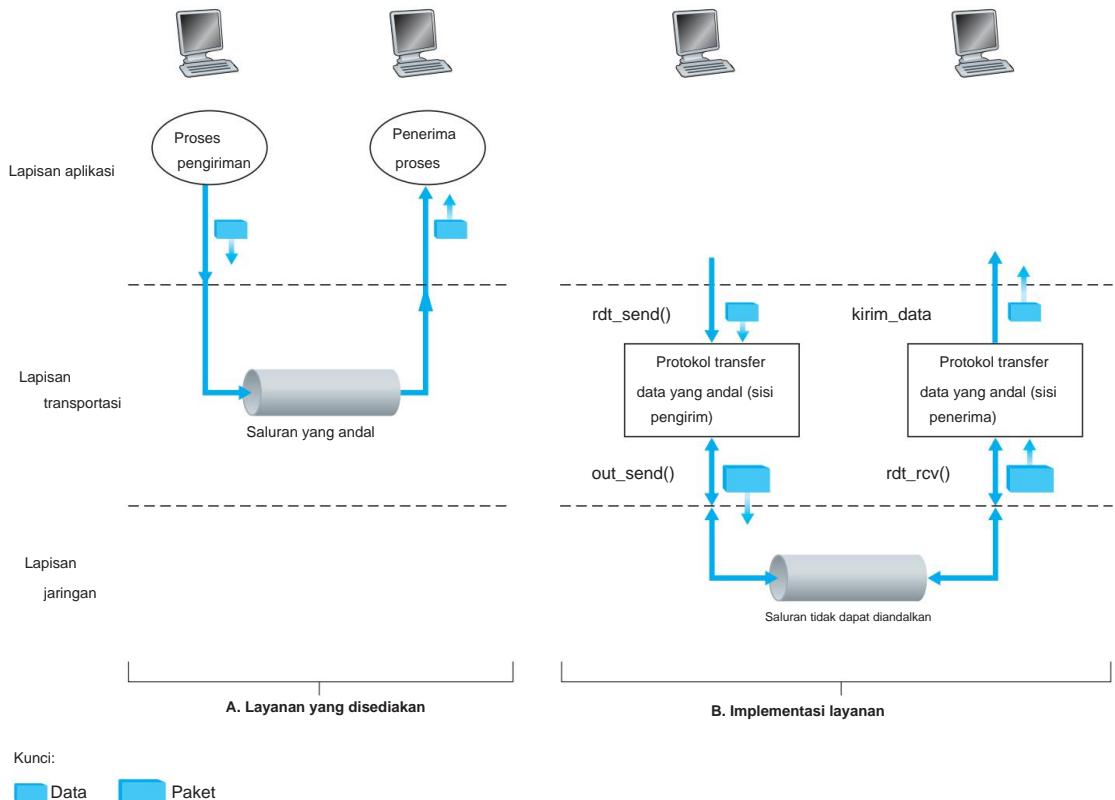
Pada bagian ini, kami mempertimbangkan masalah transfer data yang dapat diandalkan dalam konteks umum. Ini tepat karena masalah penerapan transfer data yang andal tidak hanya terjadi pada lapisan transport, tetapi juga pada lapisan tautan dan juga lapisan aplikasi. Masalah umum dengan demikian sangat penting untuk jaringan.

Memang, jika seseorang harus mengidentifikasi daftar "sepuluh besar" masalah fundamental penting di semua jaringan, ini akan menjadi kandidat untuk memimpin daftar. Pada bagian selanjutnya kita akan mempelajari TCP dan menunjukkan, khususnya, bahwa TCP mengeksploitasi banyak prinsip yang akan kita uraikan.

Gambar 3.8 mengilustrasikan kerangka kerja untuk studi kami tentang transfer data yang andal. Abstraksi layanan yang diberikan kepada entitas lapisan atas adalah saluran yang dapat diandalkan di mana data dapat ditransfer. Dengan saluran yang andal, tidak ada bit data yang ditransfer yang rusak (dibalik dari 0 ke 1, atau sebaliknya) atau hilang, dan semuanya dikirim sesuai urutan pengirimannya. Inilah model layanan yang ditawarkan oleh TCP ke aplikasi Internet yang memintanya.

Merupakan tanggung jawab **protokol transfer data yang andal** untuk mengimplementasikan abstraksi layanan ini. Tugas ini dipersulit oleh fakta bahwa lapisan *di bawah* protokol transfer data yang andal mungkin tidak dapat diandalkan. Misalnya, TCP adalah protokol transfer data andal yang diimplementasikan di atas lapisan kerja jaringan end-to-end (IP) yang tidak dapat diandalkan. Secara lebih umum, lapisan di bawah dua titik akhir komunikasi yang andal mungkin terdiri dari satu tautan fisik (seperti dalam kasus protokol transfer data tingkat tautan) atau jaringan kerja global (seperti dalam kasus protokol tingkat transportasi). . Namun, untuk tujuan kami, kami dapat melihat lapisan bawah ini hanya sebagai saluran titik-ke-titik yang tidak dapat diandalkan.

Pada bagian ini, kami akan secara bertahap mengembangkan sisi pengirim dan penerima dari protokol transfer data yang andal, dengan mempertimbangkan model saluran dasar yang semakin kompleks. Sebagai contoh, kami akan mempertimbangkan mekanisme protokol apa yang diperlukan ketika saluran yang mendasarinya dapat merusak bit atau kehilangan seluruh paket. Satu asumsi yang akan kita adopsi sepanjang diskusi kita di sini adalah bahwa paket akan dikirim sesuai urutan pengirimannya, dengan beberapa paket mungkin hilang; yaitu, saluran yang mendasarinya tidak akan menyusun ulang paket. Gambar 3.8(b) mengilustrasikan antarmuka untuk protokol transfer data kami. Sisi pengirim dari protokol transfer data akan dipanggil dari atas dengan panggilan ke `rdt_send()`. Ini akan meneruskan data yang akan dikirim ke lapisan atas di sisi penerima. (Di sini `rdt` adalah singkatan dari protokol *transfer data andal* dan *_send*



Gambar 3.8 Transfer data yang andal: Model layanan dan implementasi layanan

menunjukkan bahwa sisi pengirim rdt sedang dipanggil. Langkah pertama dalam mengembangkan protokol apa pun adalah memilih nama yang bagus!) Di sisi penerima, rdt_rcv() akan dipanggil ketika sebuah paket datang dari sisi penerima saluran. Ketika protokol rdt ingin mengirimkan data ke lapisan atas, ia akan melakukannya dengan memanggil pengiriman_data(). Berikut ini kami menggunakan terminologi "paket" daripada "segmen" lapisan transportasi. Karena teori yang dikembangkan di bagian ini berlaku untuk jaringan komputer secara umum dan tidak hanya untuk lapisan transport Internet, istilah generik "paket" mungkin lebih tepat di sini.

Pada bagian ini kami hanya mempertimbangkan kasus **transfer data searah**, yaitu transfer data dari sisi pengirim ke sisi penerima. Kasus **transfer data dua arah** (yaitu, full-duplex) yang andal secara konseptual tidak lebih sulit tetapi jauh lebih membosankan untuk dijelaskan. Meskipun kami hanya mempertimbangkan transfer data searah, penting untuk dicatat bahwa sisi pengirim dan penerima dari protokol kami tetap perlu mengirimkan paket di *kedua arah*, seperti yang ditunjukkan pada Gambar 3.8. Kita akan segera melihat bahwa, selain bertukar paket yang berisi data yang akan ditransfer, itu

206 BAB 3 • LAPISAN TRANSPORTASI

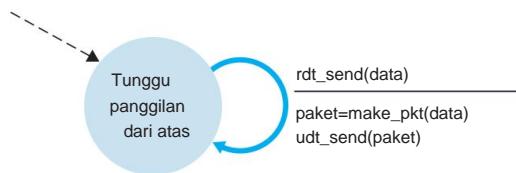
sisi pengirim dan penerima rdt juga perlu bertukar paket kontrol bolak-balik. Kedua sisi kirim dan terima dari rdt mengirim paket ke sisi lain dengan panggilan ke udt_send() (di mana udt berarti *transfer data yang tidak dapat diandalkan*).

3.4.1 Membangun Protokol Transfer Data yang Andal

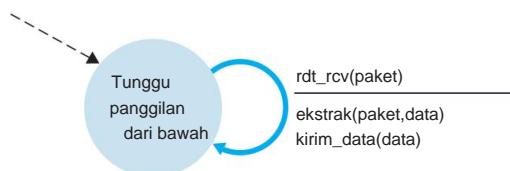
Kita sekarang melewati serangkaian protokol, masing-masing menjadi lebih kompleks, sampai pada protokol transfer data yang sempurna dan andal.

Transfer Data yang Andal melalui Saluran yang Sangat Andal: rdt1.0

Kami pertama-tama mempertimbangkan kasus paling sederhana, di mana saluran yang mendasarinya sepenuhnya dapat diandalkan. Protokol itu sendiri, yang akan kita sebut rdt1.0, sepele. Definisi **finite-state machine (FSM)** untuk pengirim dan penerima rdt1.0 ditunjukkan pada Gambar 3.9. FSM pada Gambar 3.9(a) mendefinisikan operasi pengirim, sedangkan FSM pada Gambar 3.9(b) mendefinisikan operasi penerima. Penting untuk dicatat bahwa ada **FSM terpisah** untuk pengirim dan penerima. FSM pengirim dan penerima pada Gambar 3.9 masing-masing hanya memiliki satu status. Panah dalam deskripsi FSM menunjukkan transisi protokol dari satu status ke status lainnya. (Karena setiap FSM pada Gambar 3.9 hanya memiliki satu keadaan, sebuah transisi harus dari satu keadaan kembali ke dirinya sendiri; kita akan segera melihat diagram keadaan yang lebih rumit.) Peristiwa yang menyebabkan transisi ditunjukkan di atas garis horizontal yang menandakan transisi, Dan



A. rdt1.0: sisi pengirim



B. rdt1.0: sisi penerima

Gambar 3.9 rdt1.0 – Protokol untuk saluran yang benar-benar andal

3.4 • PRINSIP TRANSFER DATA YANG ANDAL 207

tindakan yang diambil saat peristiwa terjadi ditampilkan di bawah garis horizontal. Saat tidak ada tindakan yang diambil pada suatu peristiwa, atau tidak ada peristiwa yang terjadi dan tindakan diambil, kami akan menggunakan simbol di bawah atau di atas garis horizontal, masing-masing, untuk secara eksplisit menunjukkan tidak adanya tindakan atau peristiwa. Status awal FSM ditunjukkan dengan panah putus-putus. Meskipun FSM pada Gambar 3.9 hanya memiliki satu status, FSM yang akan kita lihat sebentar lagi memiliki banyak status, jadi penting untuk mengidentifikasi status awal setiap FSM.

Sisi pengirim rdt hanya menerima data dari lapisan atas melalui peristiwa rdt_send(data), membuat paket berisi data (melalui tindakan make_pkt(data)) dan mengirimkan paket ke saluran. Dalam praktiknya, kejadian rdt_send(data) akan dihasilkan dari pemanggilan prosedur (misalnya, ke rdt_send()) oleh aplikasi lapisan atas.

Di sisi penerima, rdt menerima paket dari saluran yang mendasari melalui acara rdt_rcv(paket), menghapus data dari paket (melalui ekstrak tindakan (paket, data)) dan meneruskan data ke lapisan atas (melalui tindakan pengiriman_data(data)). Dalam praktiknya, kejadian rdt_rcv(paket) akan dihasilkan dari pemanggilan prosedur (misalnya, ke rdt_rcv()) dari protokol lapisan bawah.

Dalam protokol sederhana ini, tidak ada perbedaan antara unit data dan paket. Juga, semua aliran paket berasal dari pengirim ke penerima; dengan saluran yang sangat andal, pihak penerima tidak perlu memberikan umpan balik apa pun kepada pengirim karena tidak ada yang salah! Perhatikan bahwa kami juga berasumsi bahwa penerima dapat menerima data secepat pengirim mengirim data. Jadi, penerima tidak perlu meminta pengirim untuk melambat!

Transfer Data yang Andal melalui Saluran dengan Kesalahan Bit: rdt2.0

Sebuah model yang lebih realistik dari saluran yang mendasari adalah satu di mana bit dalam sebuah paket mungkin rusak. Kesalahan bit seperti itu biasanya terjadi pada komponen fisik jaringan saat paket ditransmisikan, disebarluaskan, atau disangga. Kami akan terus berasumsi untuk saat ini bahwa semua paket yang ditransmisikan diterima (walaupun bitnya mungkin rusak) sesuai urutan pengirimannya.

Sebelum mengembangkan protokol untuk komunikasi yang andal melalui saluran seperti itu, pertama-tama pertimbangkan bagaimana orang dapat menghadapi situasi seperti itu. Pertimbangkan bagaimana Anda sendiri dapat mendiktekan pesan panjang melalui telepon. Dalam skenario tipikal, penerima pesan mungkin mengatakan "OK" setelah setiap kalimat didengar, dipahami, dan direkam. Jika penerima pesan mendengar kalimat kacau, Anda diminta mengulangi kalimat kacau tersebut. Protokol pendiktean pesan ini menggunakan **ucapan terima kasih positif** ("OK") dan **ucapan terima kasih negatif** ("Silakan ulangi."). Pesan-pesan kontrol ini memungkinkan penerima untuk memberi tahu pengirim apa yang telah diterima dengan benar, dan apa yang telah diterima karena kesalahan sehingga perlu diulang. Dalam pengaturan jaringan komputer, protokol transfer data yang andal berdasarkan transmisi ulang tersebut dikenal sebagai **protokol ARQ** (Automatic Repeat ReQuest).

208 BAB 3 • LAPISAN TRANSPORTASI

Pada dasarnya, tiga kemampuan protokol tambahan diperlukan dalam protokol ARQ untuk menangani adanya kesalahan bit:

- *Deteksi kesalahan.* Pertama, diperlukan mekanisme untuk memungkinkan penerima mendeteksi ketika kesalahan bit telah terjadi. Ingat dari bagian sebelumnya bahwa UDP menggunakan bidang checksum Internet untuk tujuan ini. Dalam Bab 5 kita akan menguji teknik deteksi dan koreksi kesalahan secara lebih rinci; teknik ini memungkinkan penerima untuk mendeteksi dan mungkin memperbaiki kesalahan bit paket. Untuk saat ini, kita hanya perlu mengetahui bahwa teknik ini memerlukan bit tambahan (di luar bit data asli yang akan ditransfer) dikirim dari pengirim ke penerima; bit-bit ini akan dikumpulkan ke dalam bidang paket checksum dari paket data rdt2.0.

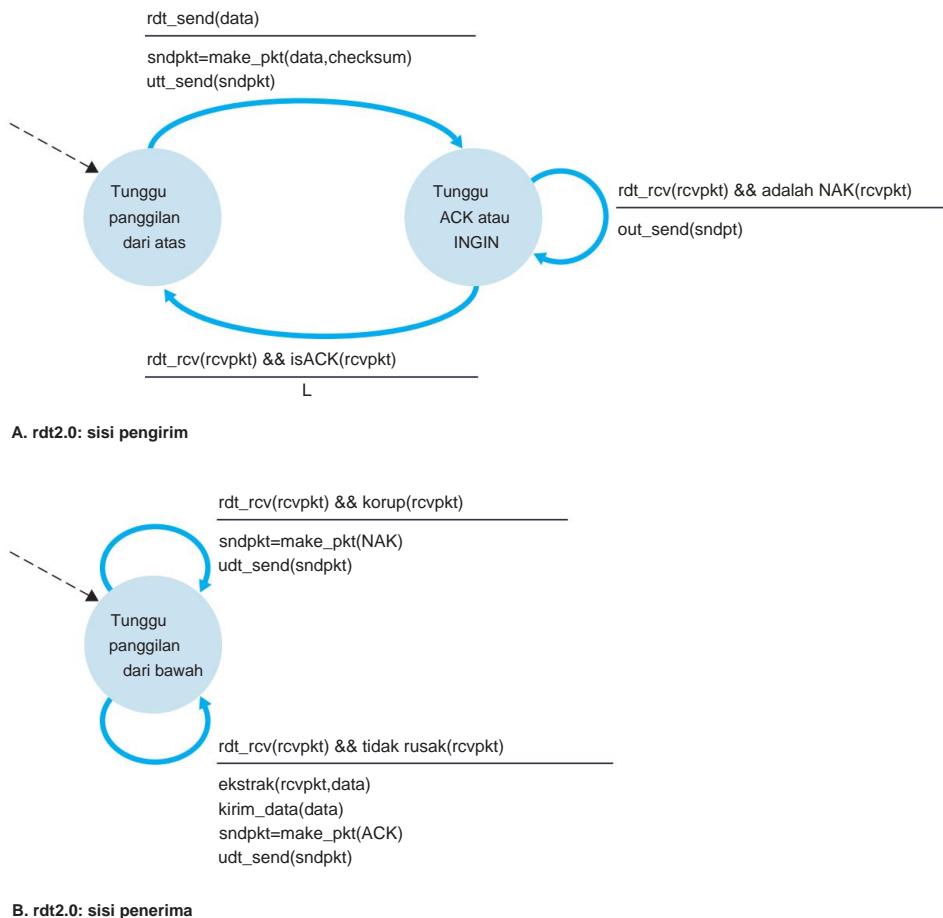
- *Umpam balik penerima.* Karena pengirim dan penerima biasanya mengeksekusi pada sistem akhir yang berbeda, mungkin terpisah ribuan mil, satu-satunya cara bagi pengirim untuk mempelajari pandangan dunia penerima (dalam hal ini, apakah paket diterima dengan benar atau tidak) adalah untuk penerima untuk memberikan umpan balik eksplisit kepada pengirim. Balasan pengakuan positif (ACK) dan negatif (NAK) dalam skenario dikte pesan adalah contoh umpan balik tersebut. Protokol rdt2.0 kami juga akan mengirimkan paket ACK dan NAK kembali dari penerima ke pengirim. Pada prinsipnya, paket-paket ini hanya membutuhkan panjang satu bit; misalnya, nilai 0 dapat menunjukkan NAK dan nilai 1 dapat menunjukkan ACK.

- *Transmisi ulang.* Paket yang diterima karena kesalahan pada penerima akan ditransmisikan ulang disampaikan oleh pengirim.

Gambar 3.10 menunjukkan representasi FSM dari rdt2.0, sebuah protokol transfer data yang menggunakan deteksi kesalahan, ucapan terima kasih positif, dan ucapan terima kasih negatif.

Sisi kirim rdt2.0 memiliki dua status. Di negara bagian paling kiri, protokol sisi kirim sedang menunggu data diturunkan dari lapisan atas. Ketika peristiwa `rdt_send(data)` terjadi, pengirim akan membuat paket (`sndpkt`) yang berisi data yang akan dikirim, bersama dengan paket checksum (misalnya, seperti yang dibahas di Bagian 3.3.2 untuk kasus segmen UDP), dan kemudian kirim paket melalui operasi `udt_send(sndpkt)`. Pada state paling kanan, protokol pengirim sedang menunggu paket ACK atau NAK dari penerima. Jika paket ACK diterima (notasi `rdt_rcv(rcvpkt) && isACK(rcvpkt)` pada Gambar 3.10 sesuai dengan kejadian ini), pengirim mengetahui bahwa paket yang terakhir dikirim telah diterima dengan benar dan dengan demikian protokol kembali ke keadaan menunggu untuk data dari lapisan atas. Jika NAK diterima, protokol mentransmisikan ulang paket terakhir dan menunggu ACK atau NAK dikembalikan oleh penerima sebagai respons terhadap paket data yang ditransmisikan ulang. Penting untuk diperhatikan bahwa saat pengirim berada dalam status `wait-for-ACK-or-NAK`, pengirim tidak *dapat* memperoleh lebih banyak data dari lapisan atas; yaitu, peristiwa `rdt_send()` tidak dapat terjadi; yang akan terjadi hanya setelah pengirim menerima ACK dan meninggalkan status ini.

Dengan demikian, pengirim tidak akan mengirim data baru sampai yakin bahwa penerima telah menerimanya



Gambar 3.10 rdt2.0—Protokol untuk saluran dengan kesalahan bit

menerima paket saat ini dengan benar. Karena perilaku ini, protokol seperti rdt2.0 dikenal sebagai protokol **stop-and-wait**.

FSM sisi penerima untuk rdt2.0 masih memiliki status tunggal. Pada kedatangan paket, penerima membalas dengan ACK atau NAK, tergantung apakah paket yang diterima rusak atau tidak. Pada Gambar 3.10, notasi $rdt_rcv(rcvpkt) \&& corrupt(rcvpkt)$ sesuai dengan peristiwa di mana sebuah paket diterima dan ditemukan kesalahan.

Protokol rdt2.0 mungkin terlihat berfungsi tetapi, sayangnya, memiliki kelemahan yang fatal. Secara khusus, kami belum memperhitungkan kemungkinan bahwa paket ACK atau NAK bisa rusak! (Sebelum melanjutkan, Anda harus memikirkan bagaimana ini

210 BAB 3 • LAPISAN TRANSPORTASI

masalah dapat diperbaiki.) Sayangnya, sedikit kekeliruan kami tidak berbahaya seperti kelihatannya. Minimal, kita perlu menambahkan bit checksum ke paket ACK/NAK untuk mendeteksi kesalahan tersebut. Pertanyaan yang lebih sulit adalah bagaimana protokol harus pulih dari kesalahan dalam paket ACK atau NAK. Kesulitan di sini adalah jika ACK atau NAK rusak, pengirim tidak memiliki cara untuk mengetahui apakah penerima telah menerima bagian terakhir dari data yang dikirimkan dengan benar atau tidak.

Pertimbangkan tiga kemungkinan untuk menangani ACK atau NAK yang rusak:

- Untuk kemungkinan pertama, pertimbangkan apa yang mungkin dilakukan manusia dalam skenario pendiktean pesan. Jika pembicara tidak mengerti jawaban "OK" atau "Please repeat that" dari penerima, pembicara mungkin akan bertanya, "Apa yang kamu katakan?" (sehingga memperkenalkan tipe baru paket pengirim-ke-penerima ke protokol kami). Penerima kemudian akan mengulangi balasannya. Tetapi bagaimana jika pembicara mengatakan "Apa yang kamu katakan?" rusak? Si penerima, yang tidak tahu apakah kalimat yang kacau itu adalah bagian dari dikte atau permintaan untuk mengulang jawaban terakhir, mungkin akan menjawab dengan "Apa yang *kamu* katakan?" Dan kemudian, tentu saja, tanggapan itu mungkin kacau. Jelas, kita sedang menuju jalan yang sulit.
- Alternatif kedua adalah menambahkan bit checksum yang cukup untuk memungkinkan pengirim tidak hanya mendeteksi, tetapi juga memulihkan dari, kesalahan bit. Ini memecahkan masalah langsung untuk saluran yang dapat merusak paket tetapi tidak menghilangkannya.
- Pendekatan ketiga adalah pengirim hanya mengirim ulang paket data saat ini ketika menerima paket ACK atau NAK yang rusak. Pendekatan ini, bagaimanapun, memperkenalkan **paket duplikat** ke dalam saluran pengirim-ke-penerima. Kesulitan mendasar dengan duplikasi paket adalah bahwa penerima tidak mengetahui apakah ACK atau NAK yang terakhir dikirim diterima dengan benar di pengirim. Dengan demikian, ia tidak dapat mengetahui secara *a priori* apakah paket yang datang berisi data baru atau merupakan transmisi ulang!

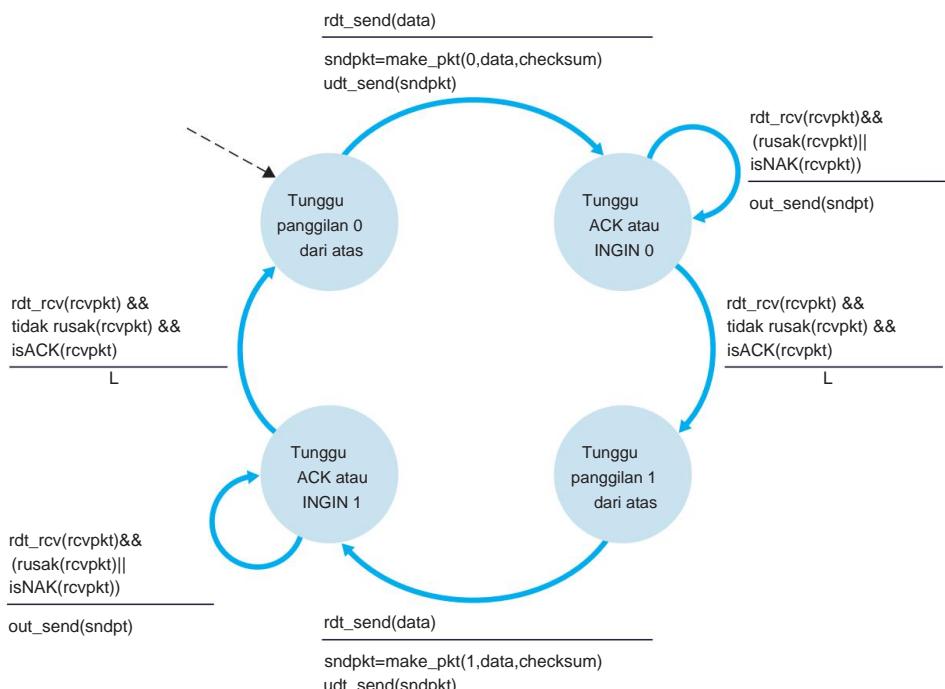
Solusi sederhana untuk masalah baru ini (dan yang diadopsi di hampir semua protokol transfer data yang ada, termasuk TCP) adalah menambahkan bidang baru ke paket data dan meminta pengirim memberi nomor paket datanya dengan memasukkan nomor urut ke bidang **ini**. Penerima kemudian hanya perlu memeriksa nomor urut ini untuk menentukan apakah paket yang diterima merupakan pengiriman ulang atau tidak. Untuk kasus sederhana dari protokol stop-and wait ini, nomor urut 1-bit sudah cukup, karena akan memungkinkan penerima untuk mengetahui apakah pengirim mengirim ulang paket yang ditransmisikan sebelumnya (nomor urut paket yang diterima memiliki urutan yang sama), nomor sebagai paket yang paling baru diterima) atau paket baru (nomor urut berubah, bergerak "maju" dalam aritmatika modulo-2). Karena saat ini kami mengasumsikan saluran yang tidak kehilangan paket, paket ACK dan NAK sendiri tidak perlu menunjukkan nomor urut paket yang mereka akui. Pengirim mengetahui bahwa paket ACK atau NAK yang diterima (baik kacau atau tidak) dihasilkan sebagai respons terhadap paket data yang terakhir dikirim.

3.4 • PRINSIP TRANSFER DATA YANG ANDAL 211

Gambar 3.11 dan 3.12 menunjukkan deskripsi FSM untuk rdt2.1, versi tetap rdt2.0 kami. FSM pengirim dan penerima rdt2.1 masing-masing sekarang memiliki status dua kali lebih banyak dari sebelumnya. Ini karena status protokol sekarang harus mencerminkan apakah paket yang saat ini sedang dikirim (oleh pengirim) atau yang diharapkan (di penerima) harus memiliki nomor urut 0 atau 1. Perhatikan bahwa tindakan di negara bagian tersebut di mana paket bernomor 0 sedang dikirim atau diharapkan adalah gambar cermin dari mereka di mana paket bernomor 1 sedang dikirim atau diharapkan; satu-satunya perbedaan berkaitan dengan penanganan nomor urut.

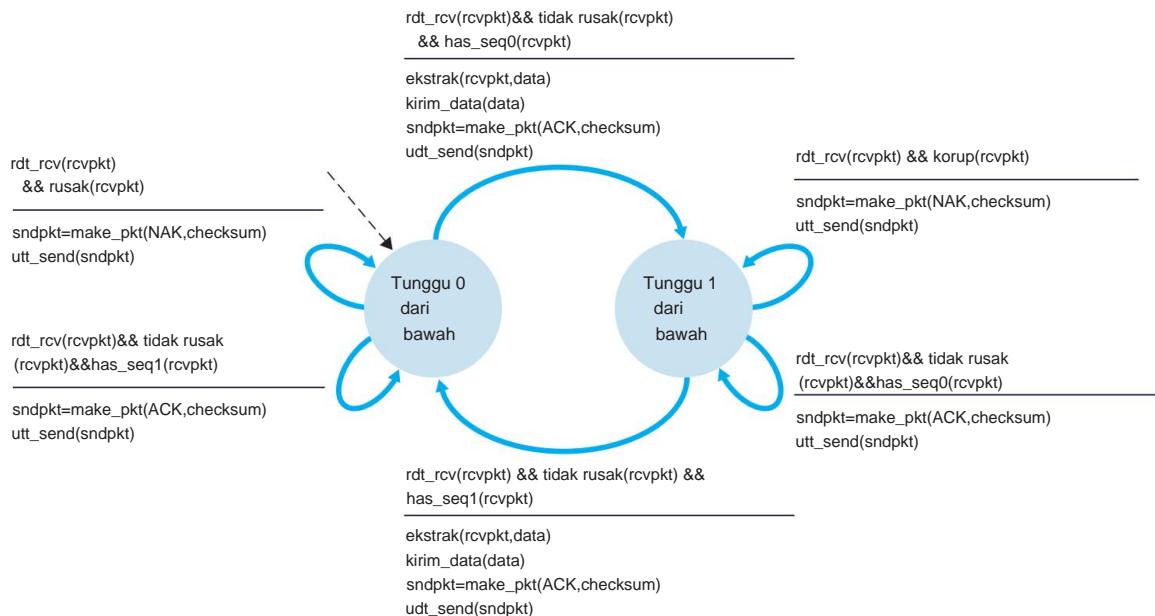
Protokol rdt2.1 menggunakan pengakuan positif dan negatif dari penerima ke pengirim. Ketika paket out-of-order diterima, penerima mengirimkan acknowledgment positif untuk paket yang telah diterimanya. Ketika paket yang rusak diterima, penerima mengirimkan pemberitahuan negatif. Kita dapat mencapai efek yang sama seperti NAK jika, alih-alih mengirim NAK, kita mengirim ACK untuk paket terakhir yang diterima dengan benar. Pengirim yang menerima dua ACK untuk paket yang sama (yaitu, menerima **duplikat ACK**) mengetahui bahwa penerima tidak menerima paket dengan benar setelah paket yang di-ACK dua kali.

Data andal bebas NAK kami



Gambar 3.11 pengirim rdt2.1

212 BAB 3 • LAPISAN TRANSPORTASI



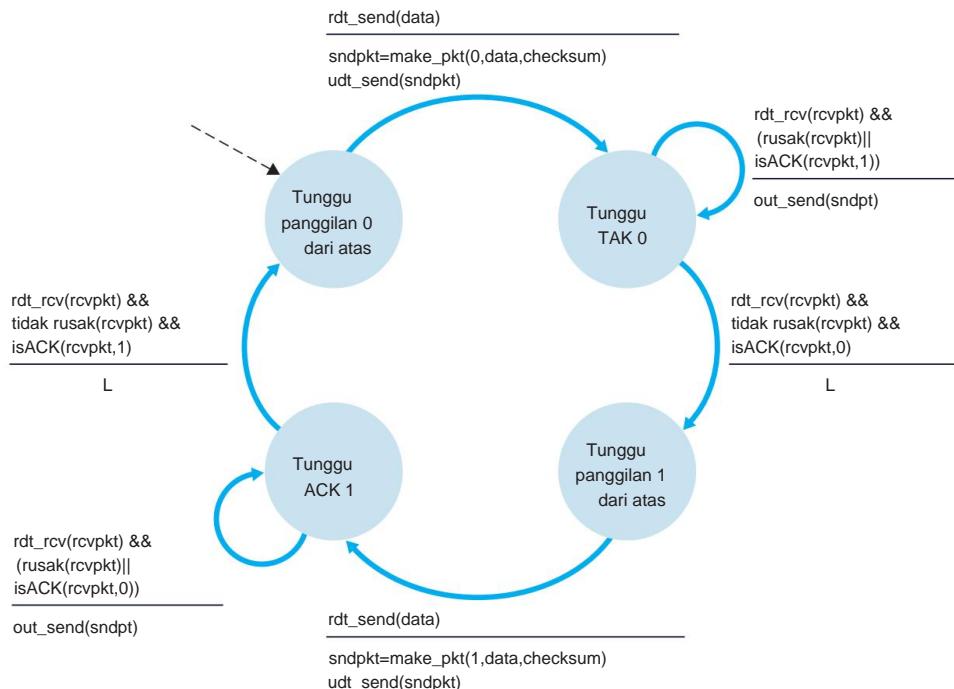
Gambar 3.12 penerima rdt2.1

protokol transfer untuk saluran dengan kesalahan bit adalah rdt2.2, ditunjukkan pada Gambar 3.13 dan 3.14. Satu perubahan halus antara rdt2.1 dan rdt2.2 adalah bahwa penerima sekarang harus menyertakan nomor urut paket yang diakui oleh pesan ACK (ini dilakukan dengan memasukkan argumen ACK,0 atau ACK,1 dalam make_pkt() di FSM penerima), dan pengirim sekarang harus memeriksa nomor urut paket yang diakui oleh pesan ACK yang diterima (ini dilakukan dengan menyertakan argumen 0 atau 1 di isACK() di FSM pengirim).

Transfer Data yang Andal melalui Saluran Rugi dengan Kesalahan Bit: rdt3.0

Misalkan sekarang selain merusak bit, saluran yang mendasarinya juga dapat *kehilangan* paket, peristiwa yang tidak biasa di jaringan komputer saat ini (termasuk Internet). Dua perhatian tambahan sekarang harus ditangani oleh protokol: bagaimana mendeteksi kehilangan paket dan apa yang harus dilakukan ketika terjadi kehilangan paket. Penggunaan ming checksum, nomor urut, paket ACK, dan transmisi ulang—teknik yang sudah dikembangkan di rdt2.2—akan memungkinkan kita untuk menjawab masalah terakhir. Penanganan masalah pertama akan membutuhkan penambahan mekanisme protokol baru.

Ada banyak kemungkinan pendekatan untuk mengatasi kehilangan paket (beberapa di antaranya dieksplorasi dalam latihan di akhir bab ini). Di sini, kami akan menempatkan beban untuk mendeteksi dan memulihkan dari paket yang hilang pada pengirim. Memperkirakan

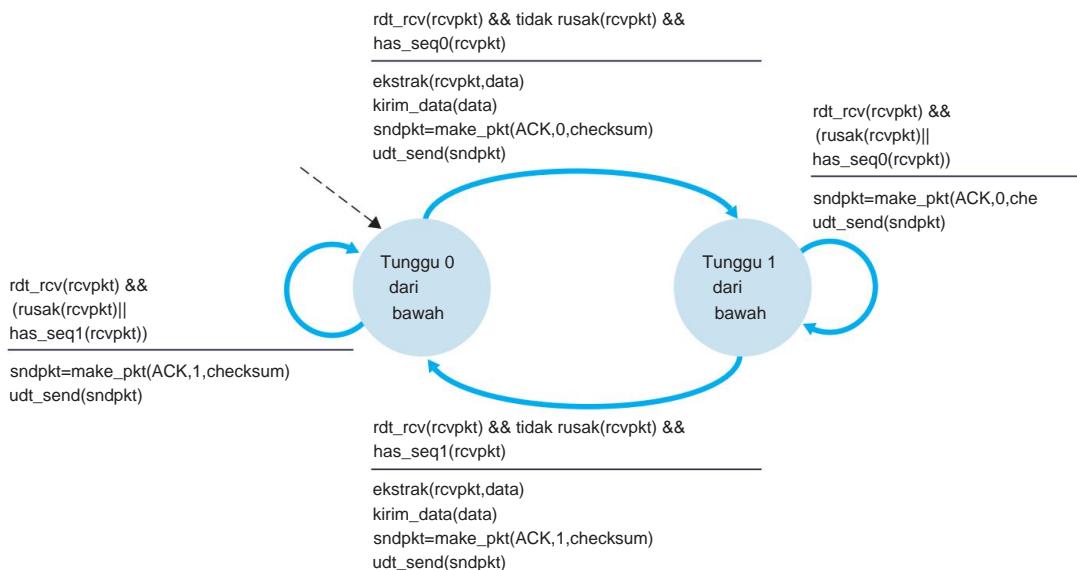


Gambar 3.13 pengirim rdt2.2

bawa pengirim mentransmisikan paket data dan paket itu, atau ACK penerima dari paket itu, hilang. Dalam kedua kasus tersebut, tidak ada balasan yang akan datang pada pengirim dari penerima. Jika pengirim bersedia menunggu cukup lama sehingga dipastikan ada paket yang hilang, maka cukup mentransmisikan ulang paket data tersebut. Anda harus meyakinkan diri sendiri bahwa protokol ini memang berfungsi.

Tetapi berapa lama pengirim harus menunggu untuk memastikan bahwa ada sesuatu yang hilang? Pengirim jelas harus menunggu setidaknya selama delay perjalanan bolak-balik antara pengirim dan penerima (yang mungkin termasuk buffering di router perantara) ditambah jumlah waktu yang diperlukan untuk memproses paket di penerima. Dalam banyak jaringan, keterlambatan maksimum kasus terburuk ini sangat sulit bahkan untuk diperkirakan, apalagi diketahui dengan pasti. Selain itu, protokol idealnya harus pulih dari kehilangan paket sesegera mungkin; menunggu penundaan terburuk bisa berarti menunggu lama sampai pemulihan kesalahan dimulai. Pendekatan yang diadopsi dalam praktiknya adalah agar pengirim secara bijaksana memilih nilai waktu sedemikian rupa sehingga kemungkinan kehilangan paket, meskipun tidak dijamin, telah terjadi. Jika ACK tidak diterima dalam waktu ini, paket ditransmisikan ulang. Perhatikan bahwa jika sebuah paket mengalami penundaan yang sangat besar, pengirim dapat mengirim ulang paket tersebut meskipun baik paket data maupun ACK-nya tidak hilang. Ini memperkenalkan kemungkinan duplikasi paket **data**

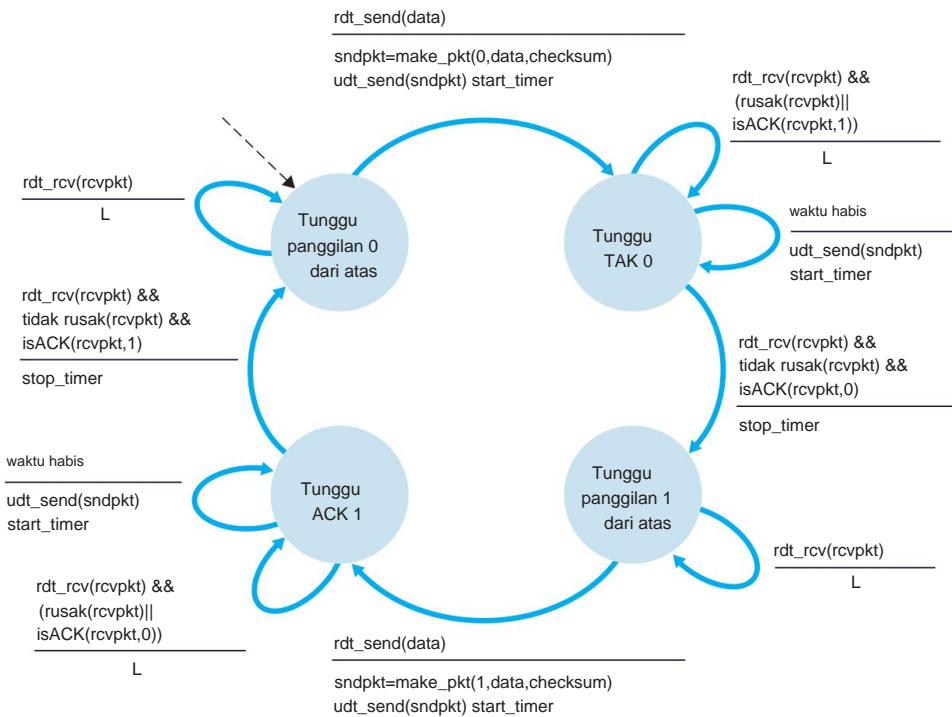
214 BAB 3 • LAPISAN TRANSPORTASI

**Gambar 3.14** penerima rdt2.2

saluran pengirim ke penerima. Untungnya, protokol rdt2.2 sudah memiliki fungsionalitas yang cukup (yaitu, nomor urut) untuk menangani kasus duplikasi paket.

Dari sudut pandang pengirim, pengiriman ulang adalah obat mujarab. Pengirim tidak tahu apakah paket data hilang, ACK hilang, atau jika paket atau ACK terlalu tertunda. Dalam semua kasus, aksinya sama: transmisi ulang. Menerapkan mekanisme pengiriman ulang berbasis waktu memerlukan **penghitung waktu mundur** yang dapat mengganggu pengirim setelah waktu tertentu habis. Pengirim dengan demikian harus dapat (1) memulai timer setiap kali paket (baik paket pertama kali atau transmisi ulang) dikirim, (2) menanggapi interupsi timer (mengambil tindakan yang sesuai), dan (3) hentikan pengatur waktu.

Gambar 3.15 menunjukkan FSM pengirim untuk rdt3.0, sebuah protokol yang secara andal mentransfer data melalui saluran yang dapat merusak atau kehilangan paket; dalam masalah pekerjaan rumah, Anda akan diminta untuk memberikan FSM receiver untuk rdt3.0. Gambar 3.16 menunjukkan bagaimana protokol beroperasi tanpa paket yang hilang atau tertunda dan bagaimana menangani paket data yang hilang. Pada Gambar 3.16, waktu bergerak maju dari bagian atas diagram menuju bagian bawah diagram; perhatikan bahwa waktu penerimaan untuk sebuah paket harus lebih lambat dari waktu pengiriman untuk sebuah paket sebagai akibat dari penundaan transmisi dan propagasi. Pada Gambar 3.16(b)–(d), tanda kurung sisi kirung menunjukkan waktu di mana pengatur waktu disetel dan waktu habis setelahnya. Beberapa aspek yang lebih halus dari protokol ini dieksplorasi dalam latihan di akhir bab ini. Karena nomor urut paket bergantian antara 0 dan 1, protokol rdt3.0 terkadang dikenal sebagai **protokol alternating-bit**.



Gambar 3.15 pemancar rdt3.0

Kami sekarang telah menyusun elemen kunci dari protokol transfer data. Jumlah cek, nomor urut, pengatur waktu, dan paket pengakuan positif dan negatif masing-masing memainkan peran penting dan perlu dalam pengoperasian protokol. Kami sekarang memiliki protokol transfer data yang andal!

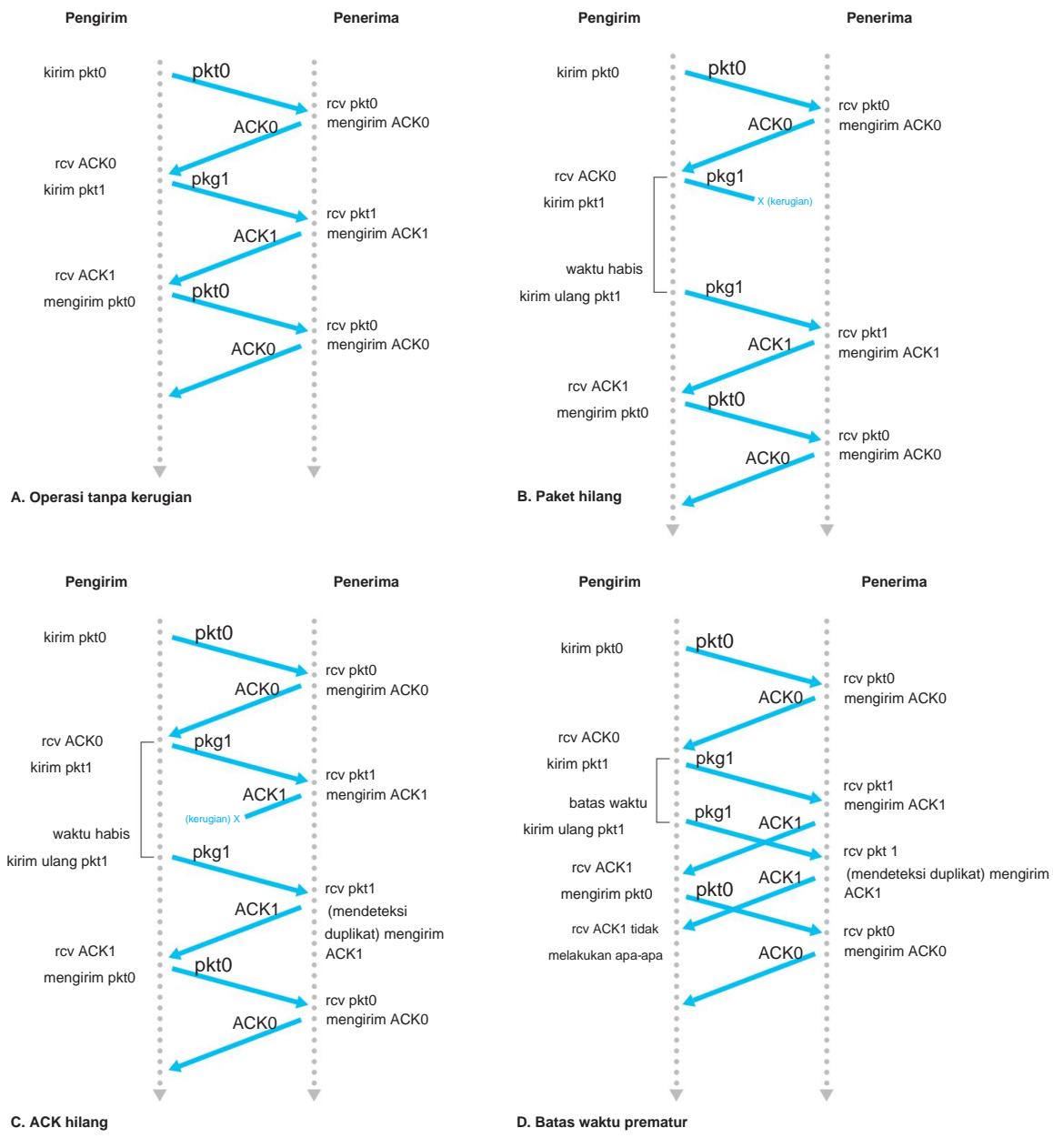


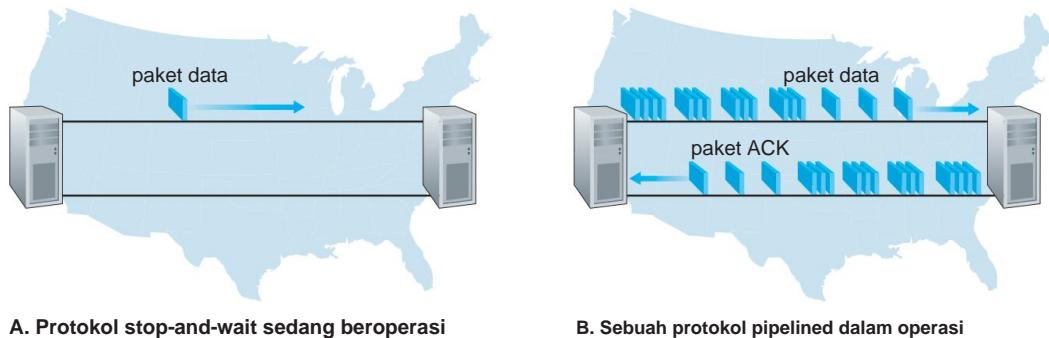
3.4.2 Protokol Transfer Data Terpipa yang Andal

Protokol rdt3.0 adalah protokol yang benar secara fungsional, tetapi tidak mungkin ada orang yang senang dengan kinerjanya, terutama di jaringan berkecepatan tinggi saat ini. Inti dari masalah kinerja rdt3.0 adalah kenyataan bahwa ini adalah protokol stop-and-wait.

Untuk menghargai dampak kinerja dari perilaku stop-and-wait ini, pertimbangkan kasus ideal dari dua host, satu terletak di Pantai Barat Amerika Serikat dan yang lainnya terletak di Pantai Timur, seperti yang ditunjukkan pada Gambar 3.17. Penundaan propagasi bolak-balik kecepatan cahaya antara kedua sistem ujung ini, RTT, kira-kira 30 milidetik. Misalkan mereka dihubungkan oleh saluran dengan laju transmisi, R, sebesar 1 Gbps (109 bit per detik). Dengan ukuran paket, L, sebesar 1.000 byte

216 BAB 3 • LAPISAN TRANSPORTASI

**Gambar 3.16** Pengoperasian rdt3.0, protokol alternating-bit



Gambar 3.17 Stop-and-wait versus protokol pipelined

(8.000 bit) per paket, termasuk bidang header dan data, waktu yang diperlukan untuk benar-benar mengirimkan paket ke tautan 1 Gbps adalah

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bit/paket}}{\text{detik mikrodetik } 109 \text{ bit/}} =$$

Gambar 3.18(a) menunjukkan bahwa dengan protokol stop-and-wait kami, jika pengirim mulai mengirimkan paket pada $t = 0$, kemudian pada $t = L/R = 8$ mikrodetik, bit terakhir memasuki saluran di sisi pengirim. Paket kemudian melakukan perjalanan lintas negara 15 msec, dengan bit terakhir paket muncul di penerima pada $t = RTT/2 + L/R = 15.008$ msec. Asumsikan untuk kesederhanaan bahwa paket ACK sangat kecil (sehingga kita dapat mengabaikan waktu transmisinya) dan bahwa penerima dapat mengirim ACK segera setelah bit terakhir dari paket data diterima, ACK muncul kembali ke pengirim pada $t = RTT + L/R = 30.008$ mdet. Pada titik ini, pengirim sekarang dapat mengirim pesan berikutnya. Jadi, dalam 30,008 msec, pengirim mengirim hanya 0,008 msec. Jika kita mendefinisikan **pemanfaatan** pengirim (atau saluran) sebagai bagian dari waktu pengirim benar-benar sibuk mengirim bit ke saluran, analisis pada Gambar 3.18(a) menunjukkan bahwa protokol stop-and-wait memiliki tampilan yang agak suram. pemanfaatan pengirim, Pengguna, dari

$$\text{Pemanfaat} = \frac{L>R}{RTT + L>R} = \frac{.008}{30.008} = 0,00027$$

Artinya, pengirim hanya sibuk 2,7 seperseratus dari satu persen waktu!

Dilihat dengan cara lain, pengirim hanya dapat mengirim 1.000 byte dalam 30,008 mil lidetik, throughput efektif hanya 267 kbps—meskipun tautan 1 Gbps tersedia! Bayangkan manajer jaringan yang tidak senang yang baru saja membayar mahal untuk link berkapasitas giga bit tetapi berhasil mendapatkan throughput hanya 267 kilobit per detik!

Ini adalah contoh grafis tentang bagaimana protokol jaringan dapat membatasi kemampuannya

218 BAB 3 • LAPISAN TRANSPORTASI

disediakan oleh perangkat keras jaringan yang mendasarinya. Selain itu, kami telah mengabaikan waktu pemrosesan protokol lapisan bawah di pengirim dan penerima, serta penundaan pemrosesan dan antrean yang akan terjadi di setiap router perantara antara pengirim dan penerima. Menyertakan efek ini hanya akan meningkatkan penundaan lebih lanjut dan semakin menonjolkan kinerja yang buruk.

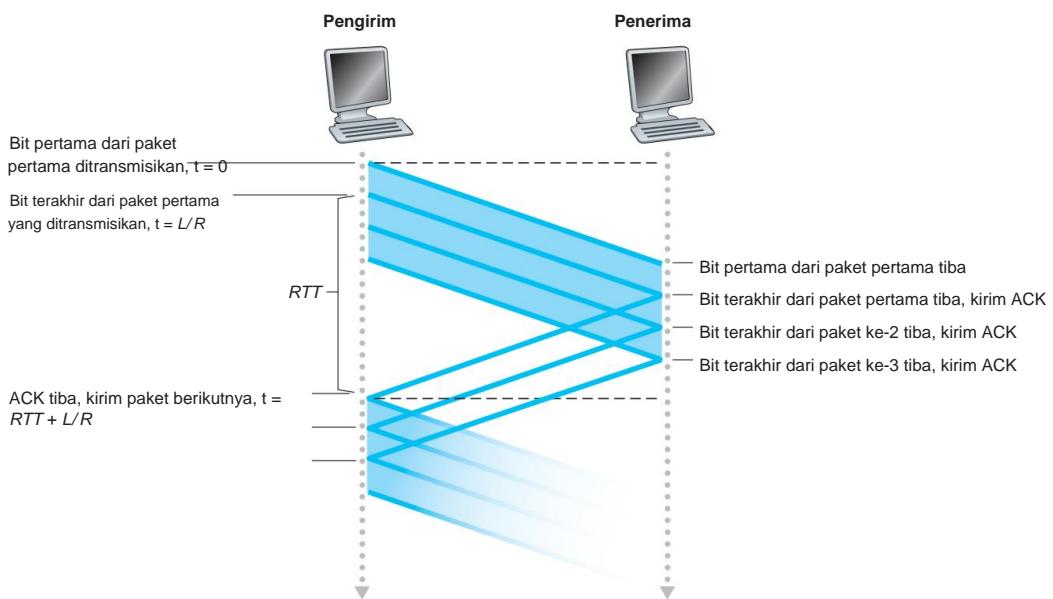
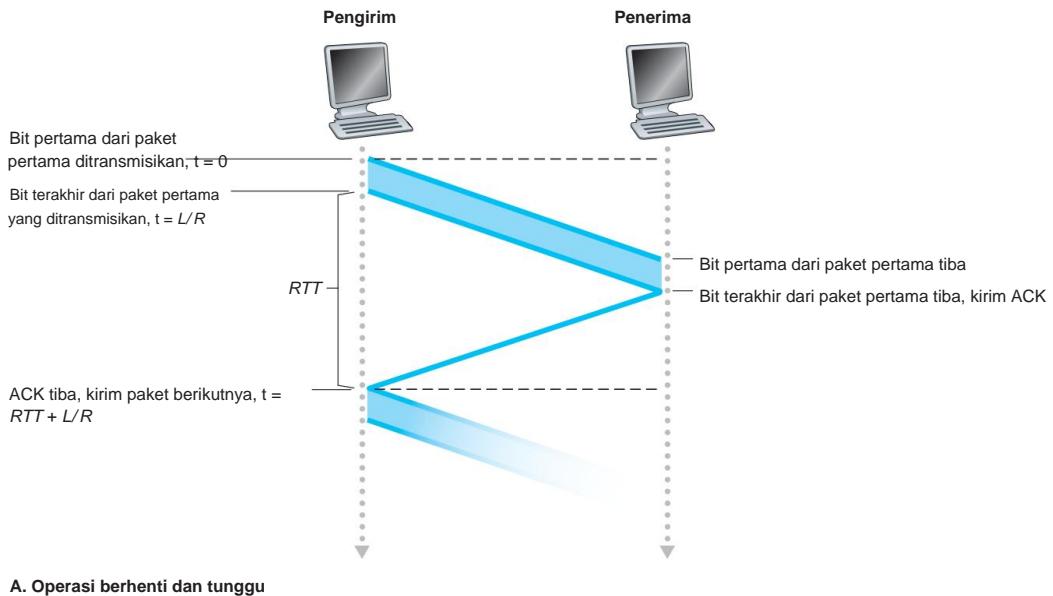
Solusi untuk masalah kinerja khusus ini sederhana: Daripada beroperasi dengan cara stop-and-wait, pengirim diperbolehkan mengirim banyak paket tanpa menunggu pemberitahuan, seperti yang diilustrasikan pada Gambar 3.17(b). Gambar 3.18(b) menunjukkan bahwa jika pengirim diperbolehkan mengirimkan tiga paket sebelum harus menunggu acknowledgment, pemanfaatan pengirim pada dasarnya tiga kali lipat. Karena banyak paket pengirim-ke-penerima dalam transit dapat divisualisasikan sebagai pengisian pipa, teknik ini dikenal sebagai **perpipaan**. Pipelining memiliki konsekuensi berikut untuk protokol transfer data yang andal:

- Kisaran nomor urut harus ditingkatkan, karena setiap paket transit (tidak termasuk pengiriman ulang) harus memiliki nomor urut yang unik dan mungkin ada banyak paket yang tidak diketahui saat transit.
- Sisi pengirim dan penerima dari protokol mungkin harus buffer lebih dari satu paket. Minimal, pengirim harus buffer paket yang telah ditransmisikan tetapi belum diakui. Penyangga paket yang diterima dengan benar mungkin juga diperlukan di penerima, seperti yang dibahas di bawah ini.
- Kisaran nomor urut yang diperlukan dan persyaratan buffering akan bergantung pada cara protokol transfer data merespons paket yang hilang, rusak, dan terlalu tertunda. Dua pendekatan dasar menuju pemulihan kesalahan pipelined dapat diidentifikasi: **Go-Back-N** dan **pengulangan selektif**.

3.4.3 Kembali-N (GBN) .

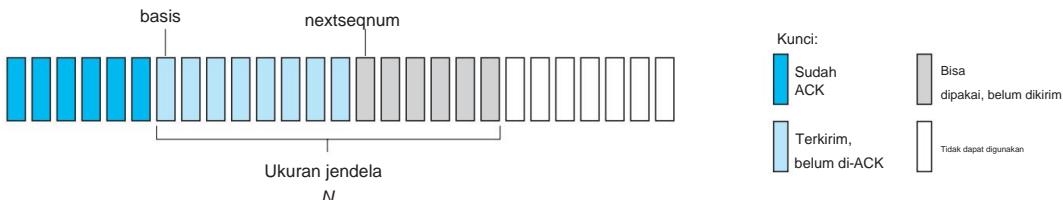
Dalam **protokol Go-Back-N (GBN)**, pengirim diizinkan untuk mengirimkan beberapa paket (bila tersedia) tanpa menunggu pengakuan, tetapi dibatasi untuk memiliki tidak lebih dari beberapa jumlah maksimum yang diperbolehkan, N , dari paket yang tidak diakui di saluran pipa. Kami menjelaskan protokol GBN secara rinci di bagian ini. Namun sebelum membaca, Anda dianjurkan untuk bermain dengan applet GBN (applet yang luar biasa!) di situs Web pendamping.

Gambar 3.19 memperlihatkan pandangan pengirim terhadap range sequence number dalam protokol GBN. Jika kita mendefinisikan basis sebagai nomor urut dari paket terlama yang tidak diakui dan nextseqnum sebagai nomor urut terkecil yang tidak terpakai (yaitu, nomor urut dari paket berikutnya yang akan dikirim), maka empat interval dalam kisaran nomor urut dapat menjadi diidentifikasi. Nomor urut dalam interval $[0, \text{basis}-1]$ sesuai dengan paket yang telah dikirim dan diakui. Inter val $[\text{base},\text{nextseqnum}-1]$ sesuai dengan paket yang telah dikirim tetapi belum diakui. Nomor urut dalam interval $[\text{nextseqnum},\text{base}+N-1]$ bisa



Gambar 3.18 Stop-and-wait dan pengiriman pipelined

220 BAB 3 • LAPISAN TRANSPORTASI



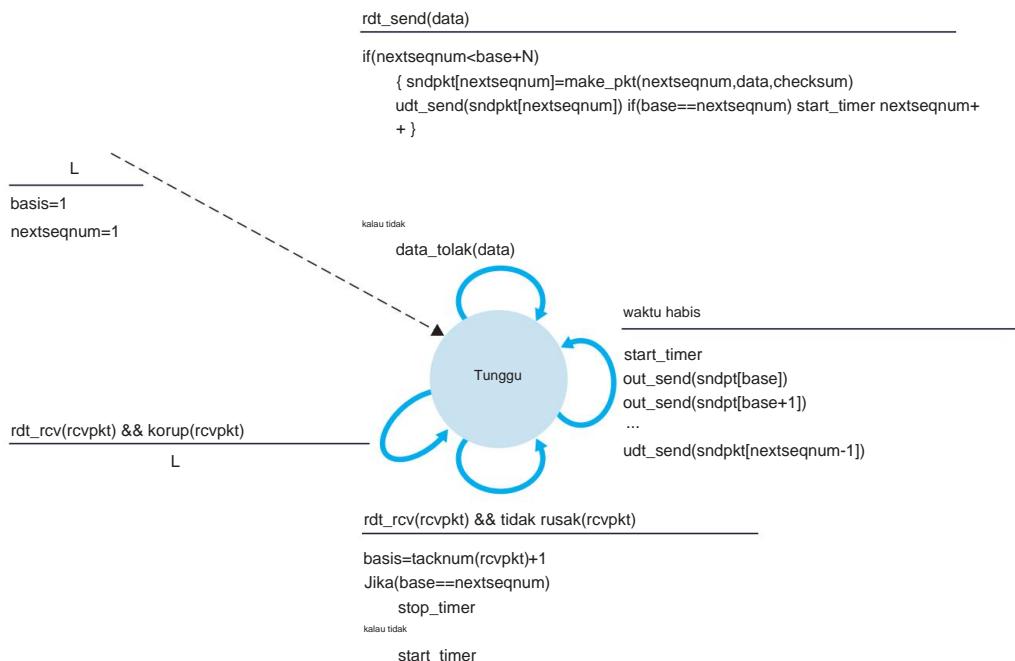
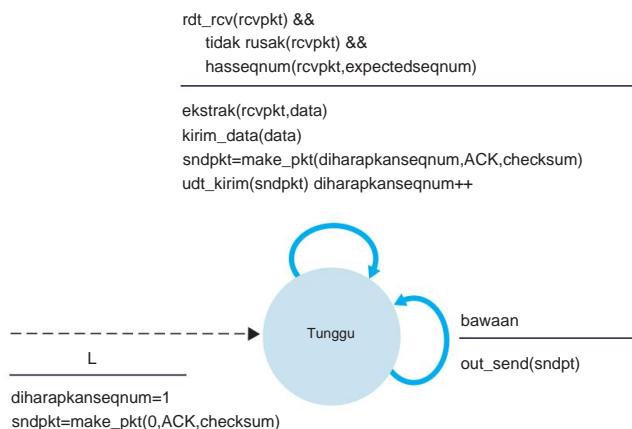
Gambar 3.19 Tampilan nomor urut pengirim di Go-Back-N

digunakan untuk paket yang dapat dikirim segera, jika data datang dari lapisan atas. Terakhir, nomor urut yang lebih besar atau sama dengan basis+N tidak dapat digunakan sampai paket yang tidak diakui yang saat ini ada dalam pipa (khususnya, paket dengan basis nomor urut) telah diakui.

Seperti yang disarankan oleh Gambar 3.19, kisaran nomor urut yang diizinkan untuk paket yang ditransmisikan tetapi belum diakui dapat dilihat sebagai jendela ukuran N di atas kisaran nomor urut. Saat protokol beroperasi, jendela ini meluncur ke depan di atas ruang nomor urut. Untuk alasan ini, N sering disebut sebagai **ukuran jendela** dan protokol GBN itu sendiri sebagai **protokol sliding-window**. Anda mungkin bertanya-tanya mengapa kami bahkan membatasi jumlah paket pengetahuan yang beredar dan tidak diketahui menjadi nilai N di tempat pertama. Mengapa tidak mengizinkan paket seperti itu dalam jumlah yang tidak terbatas? Kita akan melihat di Bagian 3.5 bahwa flow control adalah salah satu alasan untuk membatasi pengirim. Kami akan memeriksa alasan lain untuk melakukannya di Bagian 3.7, saat mempelajari kontrol kemacetan TCP.

Dalam praktiknya, nomor urut paket dibawa dalam bidang dengan panjang tetap di header paket. Jika k adalah jumlah bit dalam bidang nomor urut paket, kisaran nomor urut adalah $[0,2^k - 1]$. Dengan rentang bilangan urut yang terbatas, semua aritmatika yang melibatkan bilangan urut kemudian harus dilakukan dengan menggunakan aritmatika modulo 2^k . (Yaitu, ruang nomor urut dapat dianggap sebagai cincin berukuran 2^k di mana nomor urut $2^k - 1$ segera diikuti oleh nomor urut 0.) Ingatlah bahwa rdt3.0 memiliki nomor urut 1-bit dan rentang urutan angka $[0,1]$. Beberapa masalah di akhir bab ini mengeksplorasi konsekuensi dari rangkaian bilangan berhingga. Kita akan melihat di Bagian 3.5 bahwa TCP memiliki bidang nomor urut 32-bit, di mana nomor urut TCP menghitung byte dalam aliran byte daripada paket.

Gambar 3.20 dan 3.21 memberikan deskripsi FSM yang diperluas dari sisi pengirim dan penerima dari protokol GBN berbasis ACK, bebas NAK. Kami menyebut deskripsi FSM ini sebagai *FSM yang diperluas* karena kami telah menambahkan variabel (mirip dengan variabel bahasa pemrograman) untuk base dan nextseqnum, dan menambahkan operasi pada variabel ini dan tindakan bersyarat yang melibatkan variabel ini. Perhatikan bahwa spesifikasi FSM yang diperluas sekarang mulai terlihat seperti spesifikasi bahasa pemrograman. [Bochman 1984] memberikan survei yang sangat baik tentang ekstensi tambahan untuk teknik FSM serta teknik berbasis bahasa pemrograman lainnya untuk menentukan protokol.

**Gambar 3.20** Deskripsi Extended FSM pengirim GBN**Gambar 3.21** Deskripsi FSM yang diperluas dari penerima GBN

222 BAB 3 • LAPISAN TRANSPORTASI

Pengirim GBN harus menanggapi tiga jenis peristiwa:

- *Doa dari atas.* Ketika `rdt_send()` dipanggil dari atas, pengirim pertama-tama memeriksa untuk melihat apakah jendela sudah penuh, yaitu, apakah ada N paket yang belum diketahui. Jika jendela tidak penuh, paket dibuat dan dikirim, dan variabel diperbarui dengan tepat. Jika jendela penuh, pengirim hanya mengembalikan data ke lapisan atas, sebuah indikasi implisit bahwa jendela penuh. Lapisan atas mungkin harus mencoba lagi nanti. Dalam implementasi nyata, pengirim kemungkinan besar akan melakukan buffer (tetapi tidak segera mengirim) data ini, atau akan memiliki mekanisme sinkronisasi (misalnya, semaphore atau flag) yang akan memungkinkan lapisan atas untuk memanggil `rdt_send()` saja ketika jendela tidak penuh.
- *Penerimaan ACK.* Dalam protokol GBN kami, pengakuan untuk paket dengan nomor urut n akan dianggap sebagai **pengakuan kumulatif**, yang menunjukkan bahwa semua paket dengan nomor urut hingga dan termasuk n telah diterima dengan benar di penerima. Kami akan segera kembali ke masalah ini ketika kami memeriksa sisi penerima GBN.
- *Acara batas waktu.* Nama protokol, "Go-Back-N," berasal dari perilaku pengirim di hadapan paket yang hilang atau terlalu tertunda. Seperti dalam protokol stop-and-wait, pengatur waktu akan digunakan lagi untuk memulihkan dari data yang hilang atau paket pengakuan. Jika timeout terjadi, pengirim mengirim ulang *semua* paket yang telah dikirim sebelumnya tetapi belum diakui. Pengirim kami pada Gambar 3.20 hanya menggunakan satu pengatur waktu, yang dapat dianggap sebagai pengatur waktu untuk paket terlama yang ditransmisikan tetapi belum diakui. Jika ACK diterima tetapi masih ada paket tambahan yang ditransmisikan tetapi belum diakui, timer akan di-restart. Jika tidak ada paket yang beredar dan tidak diakui, pengatur waktu akan dihentikan.

Tindakan penerima di GBN juga sederhana. Jika paket dengan nomor urut n diterima dengan benar dan teratur (yaitu, data yang terakhir dikirim ke lapisan atas berasal dari paket dengan nomor urut $n - 1$), penerima mengirimkan ACK untuk paket n dan mengirimkan bagian data dari paket ke lapisan atas. Dalam semua kasus lain, penerima membuang paket dan mengirim ulang ACK untuk paket pesanan yang paling baru diterima. Perhatikan bahwa karena paket dikirimkan satu per satu ke lapisan atas, jika paket k telah diterima dan dikirim, maka semua paket dengan nomor urut lebih rendah dari k juga telah dikirimkan. Dengan demikian, penggunaan pengakuan kumulatif adalah pilihan alami untuk GBN.

Dalam protokol GBN kami, penerima membuang paket yang tidak sesuai pesanan. Meskipun kelihatannya konyol dan sia-sia untuk membuang paket yang diterima dengan benar (tetapi rusak), ada beberapa pemberian untuk melakukannya. Ingat bahwa penerima harus mengirimkan data untuk lapisan atas. Misalkan sekarang paket n diharapkan, tetapi paket $n + 1$ tiba. Karena data harus dikirimkan secara berurutan, penerima *dapat melakukan* buffer (menyimpan) paket $n + 1$ dan kemudian mengirimkan paket ini ke lapisan atas setelahnya nanti.

menerima dan mengirimkan paket n . Namun, jika paket n hilang, keduanya dan paket $n+1$ pada akhirnya akan ditransmisikan ulang sebagai akibat dari aturan transmisi ulang GBN di pengirim. Dengan demikian, penerima dapat dengan mudah membuang paket $n+1$.

Keuntungan dari pendekatan ini adalah kesederhanaan penyangga penerima—penerima tidak perlu menyangga paket yang *tidak* sesuai pesanan. Jadi, sementara pengirim harus mempertahankan batas atas dan bawah jendelanya dan posisi nextseqnum dalam jendela ini, satu-satunya informasi yang perlu dipertahankan penerima adalah nomor urut dari paket pesanan berikutnya. Nilai ini disimpan dalam variabel expectseqnum, yang ditunjukkan pada FSM penerima pada Gambar 3.21. Tentu saja, kerugian membuang paket yang diterima dengan benar adalah bahwa pengiriman ulang selanjutnya dari paket itu mungkin hilang atau kacau dan dengan demikian lebih banyak transmisi ulang akan diperlukan.

Gambar 3.22 menunjukkan pengoperasian protokol GBN untuk kasus ukuran jendela empat paket. Karena batasan ukuran jendela ini, pengirim mengirimkan paket ets 0 sampai 3 tetapi kemudian harus menunggu satu atau lebih dari paket ini untuk diakui sebelum melanjutkan. Karena setiap ACK berturut-turut (misalnya, ACK0 dan ACK1) diterima, jendela bergeser ke depan dan pengirim dapat mengirimkan satu paket baru (masing-masing pkt4 dan pkt5). Di sisi penerima, paket 2 hilang dan dengan demikian paket 3, 4, dan 5 ditemukan rusak dan dibuang.

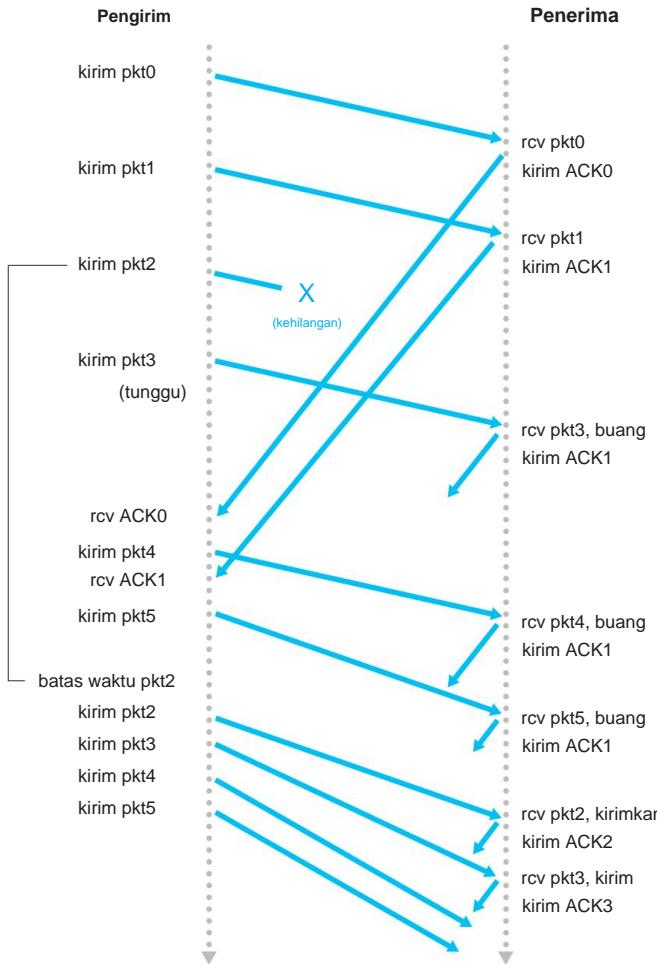
Sebelum menutup diskusi kita tentang GBN, perlu dicatat bahwa penerapan protokol ini dalam tumpukan protokol kemungkinan akan memiliki struktur yang mirip dengan FSM yang diperluas pada Gambar 3.20. Implementasinya juga kemungkinan besar berupa berbagai prosedur yang mengimplementasikan tindakan-tindakan yang akan diambil sebagai tanggapan atas berbagai peristiwa yang dapat terjadi. Dalam **pemrograman berbasis peristiwa seperti itu**, berbagai prosedur dipanggil (dipanggil) baik oleh prosedur lain di tumpukan protokol, atau sebagai hasil dari interupsi. Di pengirim, peristiwa ini akan menjadi (1) panggilan dari entitas lapisan atas untuk memanggil `rdt_send()`, (2) interupsi pengatur waktu, dan (3) panggilan dari lapisan bawah untuk memanggil `rdt_rcv()` ketika sebuah paket tiba. Latihan pemrograman di akhir bab ini akan memberi Anda kesempatan untuk benar-benar mengimplementasikan rutinitas ini dalam pengaturan jaringan yang disimulasikan namun realistik.

Kami perhatikan di sini bahwa protokol GBN menggabungkan hampir semua teknik yang akan kami temui ketika kami mempelajari komponen transfer data TCP yang andal di Bagian 3.5. Teknik ini termasuk penggunaan nomor urut, pengakuan kumulatif, checksum, dan operasi timeout/retransmit.

3.4.4 Pengulangan Selektif (SR)

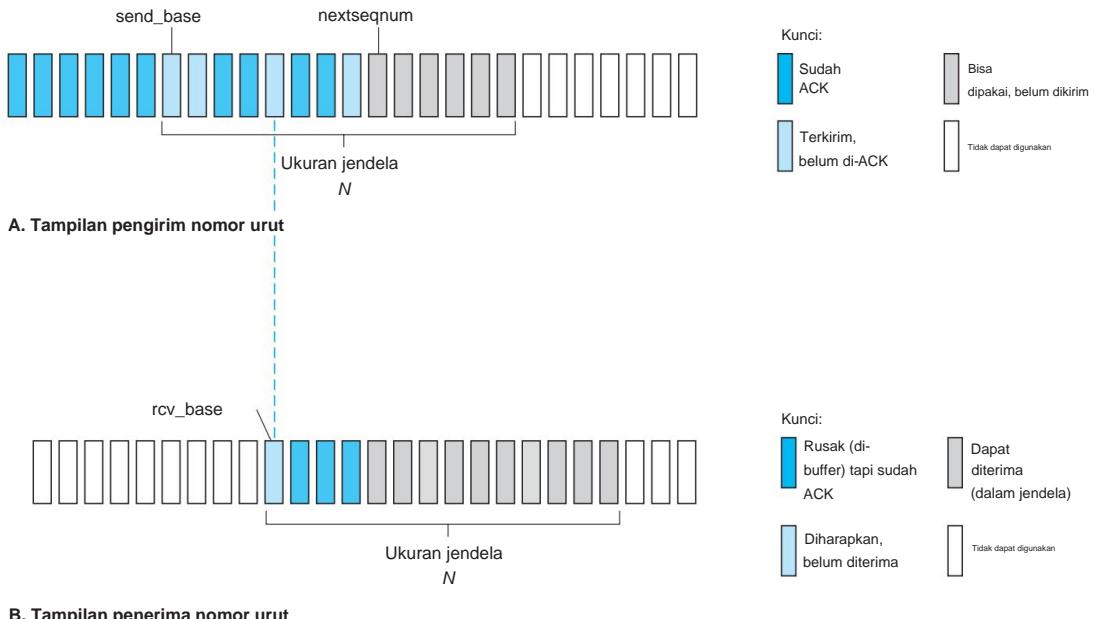
Protokol GBN memungkinkan pengirim untuk berpotensi "mengisi pipa" pada Gambar 3.17 dengan paket, sehingga menghindari masalah pemanfaatan saluran yang kami catat dengan protokol stop and-wait. Namun, ada skenario di mana GBN sendiri mengalami masalah kinerja. Secara khusus, ketika ukuran jendela dan produk bandwidth-delay sama-sama besar, banyak paket dapat berada di dalam pipa. Dengan demikian, kesalahan paket tunggal dapat menyebabkan GBN mengirim ulang sejumlah besar paket, banyak yang tidak perlu. Karena kemungkinan kesalahan saluran meningkat, saluran pipa dapat terisi

224 BAB 3 • LAPISAN TRANSPORTASI

**Gambar 3.22** Go-Back-N beroperasi

transmisi ulang yang tidak perlu ini. Bayangkan, dalam skenario pendiktean pesan kami, bahwa jika setiap kali sebuah kata dikacaukan, sekitar 1.000 kata (misalnya, ukuran jendela 1.000 kata) harus diulang. Dikte akan diperlambat oleh semua kata yang diulang.

Sebagai namanya, protokol pengulangan selektif menghindari transmisi ulang yang tidak perlu dengan meminta pengirim hanya mengirim ulang paket yang dicurigai diterima karena kesalahan (yaitu, hilang atau rusak) di penerima. Individu ini, sesuai kebutuhan, pengiriman ulang akan mengharuskan penerima secara *individual* mengakui paket yang diterima dengan benar. Ukuran jendela N akan digunakan lagi untuk membatasi jumlah



Gambar 3.23 Tampilan pengirim dan penerima Selective-repeat (SR) dari ruang nomor urut

beredar, paket yang tidak diakui dalam pipa. Namun, tidak seperti GBN, pengirim sudah menerima ACK untuk beberapa paket di jendela.

Gambar 3.23 memperlihatkan pandangan pengirim SR terhadap ruang nomor urut. Gambar 3.24 merinci berbagai tindakan yang dilakukan oleh pengirim SR.

Penerima SR akan mengakui paket yang diterima dengan benar apakah itu sesuai atau tidak. Paket out-of-order di-buffer sampai ada paket yang hilang (yaitu, paket dengan nomor urut yang lebih rendah) diterima, di mana sekumpulan paket dapat dikirim ke lapisan atas. Gambar 3.25 merinci berbagai tindakan yang diambil oleh penerima SR. Gambar 3.26 menunjukkan contoh operasi SR dengan adanya paket yang hilang. Perhatikan bahwa pada Gambar 3.26, penerima awalnya mem-buffer paket 3, 4, dan 5, dan mengirimkannya bersama dengan paket 2 ke lapisan atas ketika paket 2 akhirnya diterima.

Penting untuk dicatat bahwa pada Langkah 2 pada Gambar 3.25, penerima mengakui kembali (daripada mengabaikan) paket yang telah diterima dengan nomor urut tertentu *di bawah* basis jendela saat ini. Anda harus meyakinkan diri sendiri bahwa pengakuan ulang ini memang diperlukan. Mengingat ruang nomor urut pengirim dan penerima pada Gambar 3.23, misalkan, jika tidak ada ACK untuk paket send_base yang menyebar dari penerima ke pengirim, pengirim pada akhirnya akan mengirimkan kembali paket send_base, meskipun sudah jelas (bagi kami, bukan pengirim!) yang telah diterima oleh penerima

226 BAB 3 • LAPISAN TRANSPORTASI

1. *Data diterima dari atas.* Ketika data diterima dari atas, pengirim SR memeriksa nomor urut berikutnya yang tersedia untuk paket tersebut. Jika nomor urut berada di dalam jendela pengirim, data dikemas dan dikirim; jika tidak, itu akan disangga atau dikembalikan ke lapisan atas untuk transmisi selanjutnya, seperti di GBN.
2. *Waktu habis.* Pengatur waktu kembali digunakan untuk melindungi dari paket yang hilang. Namun, setiap paket sekarang harus memiliki pengatur waktu logisnya sendiri, karena hanya satu paket yang akan dikirim pada waktu habis. Timer perangkat keras tunggal dapat digunakan untuk meniru pengoperasian beberapa timer logis [Varghese 1997].
3. *ACK diterima.* Jika ACK diterima, pengirim SR menandai paket itu sebagai telah diterima, asalkan ada di jendela. Jika nomor urut paket sama dengan `send_base`, basis jendela dipindahkan ke depan ke paket yang tidak diakui dengan nomor urut terkecil. Jika jendela bergerak dan ada paket yang tidak terkirim dengan nomor urut yang sekarang berada di dalam jendela, paket ini ditransmisikan.

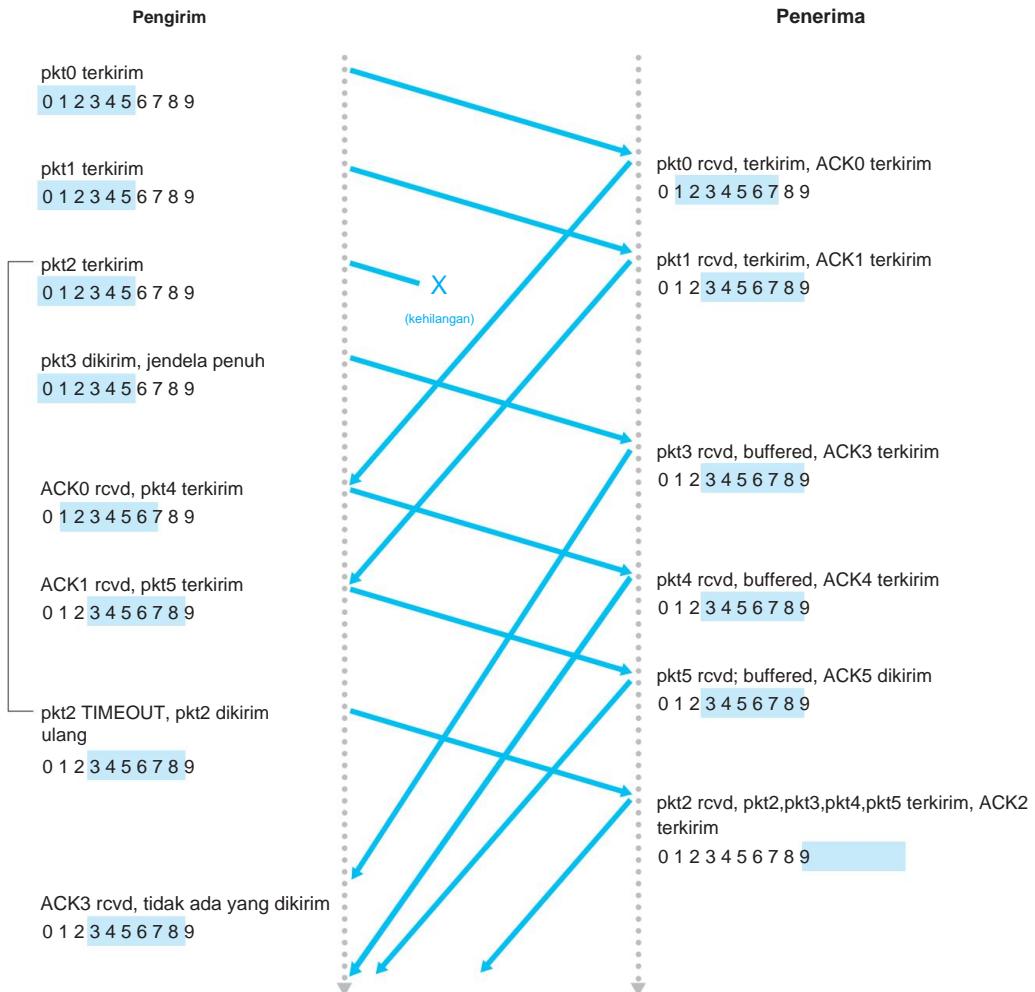
Gambar 3.24 peristiwa dan tindakan pengirim SR

1. *Paket dengan nomor urut di $[rcv_base, rcv_base+N-1]$ diterima dengan benar.* Dalam hal ini, paket yang diterima termasuk dalam window penerima dan paket ACK selektif dikembalikan ke pengirim. Jika paket tersebut sebelumnya tidak diterima, itu di-buffer. Jika paket ini memiliki nomor urut yang sama dengan dasar dari jendela penerima (`rcv_base` pada Gambar 3.22), maka paket ini, dan paket yang sebelumnya di-buffer dan diberi nomor secara berurutan (dimulai dengan `rcv_base`) dikirimkan ke lapisan atas. Jendela terima kemudian dipindahkan ke depan dengan jumlah paket yang dikirim ke lapisan atas. Sebagai contoh, perhatikan Gambar 3.26. Ketika paket dengan nomor urut $rcv_base=2$ diterima, paket tersebut dan paket 3, 4, dan 5 dapat dikirimkan ke lapisan atas.
2. *Paket dengan nomor urut di $[rcv_base-N, rcv_base-1]$ diterima dengan benar.* Dalam hal ini, ACK harus dibuat, meskipun ini adalah paket yang sebelumnya telah diakui oleh penerima.
3. *Jika tidak.* Abaikan paketnya.

Gambar 3.25 peristiwa dan tindakan penerima SR

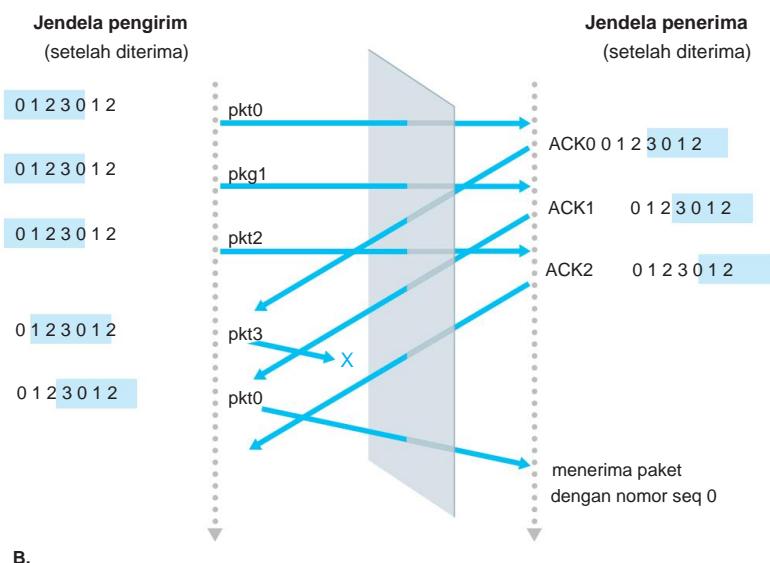
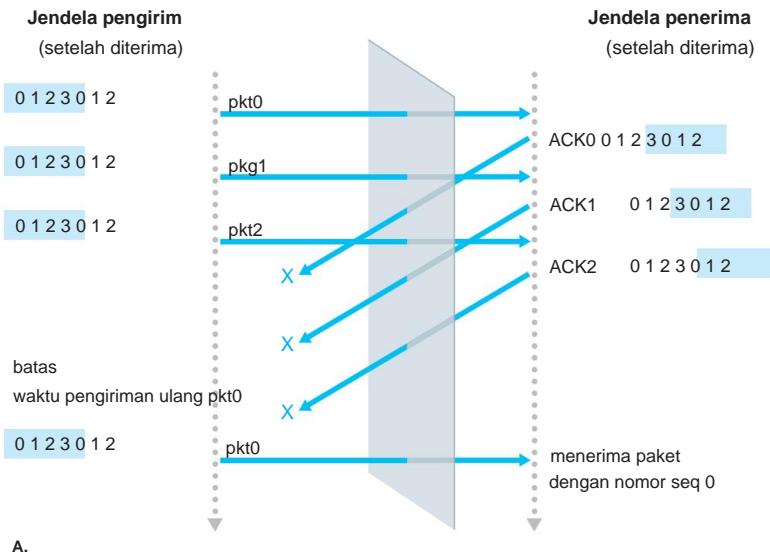
paket itu. Jika penerima tidak mengakui paket ini, kemenangan pengirim tidak akan pernah bergerak maju! Contoh ini mengilustrasikan aspek penting dari protokol SR (dan banyak protokol lainnya juga). Pengirim dan penerima tidak akan selalu memiliki pandangan yang identik tentang apa yang diterima dengan benar dan apa yang tidak.

Untuk protokol SR, ini berarti jendela pengirim dan penerima tidak selalu bersamaan.

**Gambar 3.26** Operasi SR

Kurangnya sinkronisasi antara jendela pengirim dan penerima memiliki konsekuensi penting ketika kita dihadapkan pada realitas kisaran nomor urut yang terbatas. Pertimbangkan apa yang bisa terjadi, misalnya, dengan kisaran terbatas dari empat nomor urut paket, 0, 1, 2, 3, dan ukuran jendela tiga. Misalkan paket 0 sampai 2 ditransmisikan dan diterima dengan benar dan diakui penerima. Pada titik ini, jendela penerima berada di atas paket keempat, kelima, dan keenam, yang masing-masing memiliki nomor urut 3, 0, dan 1. Sekarang pertimbangkan dua skenario. Dalam skenario pertama, ditunjukkan pada Gambar 3.27(a), ACK untuk tiga paket pertama hilang dan

228 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.27 Dilema penerima SR dengan jendela terlalu besar: Paket baru atau pengiriman ulang?

pengirim mentransmisikan ulang paket-paket ini. Penerima selanjutnya menerima paket dengan nomor urut 0—salinan dari paket pertama yang dikirim.

Dalam skenario kedua, ditunjukkan pada Gambar 3.27(b), ACK untuk tiga paket pertama semuanya terkirim dengan benar. Pengirim dengan demikian memindahkan jendelanya ke depan dan mengirimkan paket keempat, kelima, dan keenam, dengan nomor urut 3, 0, dan 1, masing-masing. Paket dengan nomor urut 3 hilang, tetapi paket dengan nomor urut 0 tiba—paket berisi data *baru*.

Sekarang pertimbangkan sudut pandang penerima pada Gambar 3.27, yang memiliki tirai kiasan antara pengirim dan penerima, karena penerima tidak dapat "melihat" tindakan yang diambil oleh pengirim. Semua yang diamati penerima adalah urutan pesan yang diterimanya dari saluran dan dikirim ke saluran. Sejauh ini, dua skenario pada Gambar 3.27 *identik*. Tidak ada cara untuk membedakan transmisi ulang paket pertama dari transmisi asli paket kelima.

Jelas, ukuran jendela yang kurang dari 1 ukuran ruang nomor urut tidak akan berfungsi. Tapi seberapa kecil ukuran jendelanya? Masalah di akhir bab meminta Anda untuk menunjukkan bahwa ukuran jendela harus kurang dari atau sama dengan setengah ukuran ruang nomor urut untuk protokol SR.

Di situs Web pendamping, Anda akan menemukan applet yang menjawab pengoperasian protokol SR. Coba lakukan eksperimen yang sama seperti yang Anda lakukan dengan applet GBN. Apakah hasilnya sesuai dengan yang Anda harapkan?

Ini melengkapi diskusi kami tentang protokol transfer data yang andal. Kami telah membahas *banyak* hal dan memperkenalkan banyak mekanisme yang bersama-sama menyediakan transfer data yang andal. Tabel 3.1 merangkum mekanisme ini. Sekarang kita telah melihat semua mekanisme ini beroperasi dan dapat melihat "gambaran besarnya", kami mendorong Anda untuk meninjau kembali bagian ini untuk melihat bagaimana mekanisme ini ditambahkan secara bertahap untuk mencakup model saluran yang semakin kompleks (dan realistik) yang menghubungkan pengirim dan penerima, atau untuk meningkatkan kinerja protokol.

Mari kita simpulkan diskusi kita tentang protokol transfer data yang andal dengan mempertimbangkan satu asumsi tersisa dalam model saluran dasar kita. Ingatlah bahwa kita telah mengasumsikan bahwa paket tidak dapat diatur ulang dalam saluran antara pengirim dan penerima. Ini umumnya merupakan asumsi yang masuk akal ketika pengirim dan penerima dihubungkan oleh satu kabel fisik. Namun, ketika "saluran" yang menghubungkan keduanya adalah jaringan, penataan ulang paket dapat terjadi. Salah satu manifestasi dari penyusunan ulang paket adalah bahwa salinan lama dari sebuah paket dengan urutan atau nomor pengakuan x dapat muncul, meskipun jendela pengirim maupun penerima tidak berisi x . Dengan penyusunan ulang paket, saluran dapat dianggap sebagai buffering paket dan secara spontan memancarkan paket-paket ini kapan saja di masa depan. Karena nomor urut dapat digunakan kembali, beberapa kehati-hatian harus dilakukan untuk mencegah paket duplikat tersebut. Pendekatan yang diambil dalam praktiknya adalah untuk memastikan bahwa nomor urut tidak digunakan kembali sampai pengirim "yakin" bahwa paket yang dikirim sebelumnya dengan nomor urut x tidak lagi berada di jaringan. Ini dilakukan dengan mengasumsikan bahwa sebuah paket tidak dapat "hidup" di jaringan lebih lama dari jumlah waktu maksimum yang tetap. Masa pakai paket maksimum sekitar tiga menit diasumsikan dalam ekstensi TCP

230 BAB 3 • LAPISAN TRANSPORTASI

Mekanisme	Gunakan, Komentar
Checksum	Digunakan untuk mendeteksi kesalahan bit dalam paket yang ditransmisikan.
Timer	Digunakan untuk timeout/transmisi ulang paket, mungkin karena paket (atau ACK-nya) hilang di dalam saluran. Karena batas waktu dapat terjadi ketika sebuah paket tertunda tetapi tidak hilang (batas waktu prematur), atau ketika sebuah paket telah diterima oleh penerima tetapi ACK penerima-ke-pengirim telah hilang, salinan duplikat dari sebuah paket dapat diterima oleh penerima .
Nomor urut	Digunakan untuk penomoran berurutan paket data yang mengalir dari pengirim ke penerima. Kesenjangan dalam nomor urut paket yang diterima memungkinkan penerima untuk mendeteksi paket yang hilang. Paket dengan nomor urut duplikat memungkinkan penerima untuk mendeteksi salinan duplikat dari sebuah paket.
Pengakuan	Digunakan oleh penerima untuk memberi tahu pengirim bahwa paket atau kumpulan paket telah diterima dengan benar. Pengakuan biasanya akan membawa nomor urut dari paket atau paket yang diakui. Pengakuan dapat individual atau kumulatif, tergantung pada protokol.
Pengakuan negatif	Digunakan oleh penerima untuk memberi tahu pengirim bahwa paket belum diterima dengan benar. Acknowledgment negatif biasanya akan membawa nomor urut paket et yang tidak diterima dengan benar.
Jendela, pipa	Pengirim mungkin dibatasi hanya mengirim paket dengan nomor urut yang berada dalam kisaran tertentu. Dengan mengizinkan beberapa paket untuk ditransmisikan tetapi belum diakui, pemanfaatan pengirim dapat ditingkatkan melalui mode operasi stop-and-wait. Kita akan segera melihat bahwa ukuran jendela dapat diatur berdasarkan kemampuan penerima untuk menerima dan menyangga pesan, atau tingkat kermacetan di jaringan, atau keduanya.

Tabel 3.1 Ringkasan mekanisme transfer data yang andal dan penggunaannya

untuk jaringan berkecepatan tinggi [RFC 1323]. [Sunshine 1978] menjelaskan metode untuk menggunakan nomor urut sehingga masalah penataan ulang dapat dihindari sepenuhnya.

3.5 Transportasi Berorientasi Koneksi: TCP

Sekarang setelah kita membahas prinsip-prinsip dasar transfer data yang andal, mari kita beralih ke TCP—protokol transport-layer Internet, berorientasi-koneksi, dan andal. Di bagian ini, kita akan melihat bahwa untuk menyediakan transfer data yang andal, TCP bergantung pada banyak prinsip dasar yang dibahas di bagian sebelumnya, termasuk deteksi kesalahan, pengiriman ulang, pengakuan kumulatif, pengatur waktu,

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 231

dan bidang tajuk untuk urutan dan nomor pengakuan. TCP didefinisikan dalam RFC 793, RFC 1122, RFC 1323, RFC 2018, dan RFC 2581.

3.5.1 Koneksi TCP

TCP dikatakan berorientasi **koneksi** karena sebelum satu proses aplikasi dapat mulai mengirim data ke yang lain, kedua proses pertama-tama harus "berjabat tangan" satu sama lain — yaitu, mereka harus mengirim beberapa segmen awal satu sama lain untuk menetapkan parameter transfer data berikutnya. Sebagai bagian dari pembentukan koneksi TCP, kedua sisi koneksi akan menginisialisasi banyak variabel status TCP (banyak di antaranya akan dibahas di bagian ini dan di Bagian 3.7) yang terkait dengan koneksi TCP.

"Koneksi" TCP bukanlah sirkuit TDM atau FDM end-to-end seperti pada jaringan circuit switched. Juga bukan sirkuit virtual (lihat Bab 1), karena status koneksi berada sepenuhnya di dua sistem ujung. Karena protokol TCP hanya berjalan di sistem akhir dan bukan di elemen jaringan perantara (router dan sakelar lapisan tautan), elemen jaringan perantara tidak mempertahankan status koneksi TCP.



VINTON CERF, ROBERT KAHN, DAN TCP/IP

Pada awal 1970-an, jaringan packet-switched mulai berkembang biak, dengan ARPAnet—pendahulu Internet—hanya salah satu dari banyak jaringan. Masing-masing jaringan ini memiliki protokolnya sendiri. Dua peneliti, Vinton Cerf dan Robert Kahn, menyadari pentingnya interkoneksi jaringan ini dan menemukan protokol lintas jaringan yang disebut TCP/IP, yang merupakan singkatan dari Transmission Control Protocol/Internet Protocol. Meskipun Cerf dan Kahn mulai dengan melihat protokol sebagai satu kesatuan, kemudian dibagi menjadi dua bagian, TCP dan IP, yang beroperasi secara terpisah. Cerf dan Kahn menerbitkan makalah tentang TCP/IP pada Mei 1974 di [Cerf 1974]. IEEE
Transaksi Komunikasi Teknologi

Protokol TCP/IP, yang merupakan roti dan mentega dari Internet saat ini, dirancang sebelum PC, workstation, smartphone, dan tablet, sebelum普及化 Ethernet, kabel, dan DSL, WiFi, dan teknologi jaringan akses lainnya, dan sebelum Web, media sosial, dan streaming video. Cerf dan Kahn melihat perlunya protokol jaringan yang, di satu sisi, memberikan dukungan luas untuk aplikasi yang belum ditentukan dan, di sisi lain, memungkinkan host sewenang-wenang dan protokol lapisan tautan untuk saling beroperasi.

Pada tahun 2004, Cerf dan Kahn menerima ACM's Turing Award, dianggap sebagai "Hadiah Nobel Komputasi" untuk "pekerjaan perintis pada internetworking, termasuk desain dan implementasi protokol komunikasi dasar Internet, TCP/IP, dan kepemimpinan yang menginspirasi dalam jaringan."

232 BAB 3 • LAPISAN TRANSPORTASI

Faktanya, router perantara sama sekali tidak menyadari koneksi TCP; mereka melihat datagram, bukan koneksi.

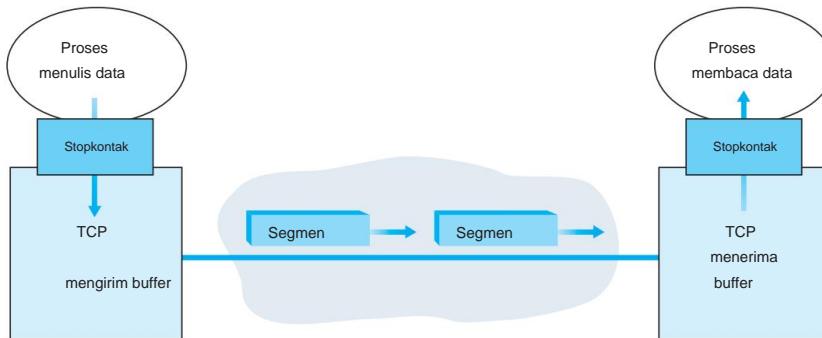
Koneksi TCP menyediakan **layanan dupleks penuh**: Jika ada koneksi TCP antara Proses A di satu host dan Proses B di host lain, maka data lapisan aplikasi dapat mengalir dari Proses A ke Proses B pada saat yang sama dengan aliran data lapisan aplikasi dari Proses B ke Proses A. Koneksi TCP juga selalu **point-to-point**, yaitu antara satu pengirim dan satu penerima. Apa yang disebut "multicasting" (lihat Bagian 4.7)—transfer data dari satu pengirim ke banyak penerima dalam satu operasi pengiriman—tidak dimungkinkan dengan TCP. Dengan TCP, dua host adalah perusahaan dan tiga adalah kerumunan!

Sekarang mari kita lihat bagaimana koneksi TCP dibuat. Misalkan proses yang berjalan di satu host ingin memulai koneksi dengan proses lain di host lain. Ingatlah bahwa proses yang memulai koneksi disebut *proses klien*, sedangkan proses lainnya disebut *proses server*. Proses aplikasi klien pertama-tama memberi tahu lapisan transport klien bahwa ia ingin membuat koneksi ke proses di server. Ingat dari Bagian 2.7.2, program klien Python melakukan ini dengan mengeluarkan perintah

```
clientSocket.connect((serverName,serverPort))
```

di mana `serverName` adalah nama server dan `serverPort` mengidentifikasi proses di server. TCP di klien kemudian mulai membuat koneksi TCP dengan TCP di server. Di akhir bagian ini kami membahas secara rinci prosedur pembuatan koneksi. Untuk saat ini cukup diketahui bahwa klien terlebih dahulu mengirimkan segmen TCP khusus; server merespons dengan segmen TCP khusus kedua; dan akhirnya klien merespon lagi dengan segmen khusus ketiga. Dua segmen pertama tidak membawa muatan, yaitu tidak ada data lapisan aplikasi; ketiga dari segmen ini dapat membawa muatan. Karena tiga segmen dikirim antara dua host, prosedur pembentukan koneksi ini sering disebut sebagai **jabat tangan tiga arah**.

Setelah koneksi TCP dibuat, kedua proses aplikasi dapat saling mengirim data. Mari pertimbangkan pengiriman data dari proses klien ke proses server. Proses klien melewati aliran data melalui soket (pintu proses), seperti yang dijelaskan dalam Bagian 2.7. Setelah data melewati pintu, data berada di tangan TCP yang berjalan di klien. Seperti ditunjukkan pada Gambar 3.28, TCP mengarahkan data ini ke **send buffer** koneksi, yang merupakan salah satu buffer yang disisihkan selama three-way handshake awal. Dari waktu ke waktu, TCP akan mengambil potongan data dari send buffer dan meneruskan data ke lapisan jaringan. Menariknya, spesifikasi TCP [RFC 793] sangat santai tentang menentukan kapan TCP harus benar-benar mengirim data buffer, yang menyatakan bahwa TCP harus "mengirimkan data itu dalam segmen-segmen dengan kenyamanannya sendiri." Jumlah maksimum data yang dapat diambil dan ditempatkan dalam suatu segmen dibatasi oleh **ukuran segmen maksimum (MSS)**. MSS biasanya diatur dengan terlebih dahulu menentukan panjang frame link-layer terbesar yang dapat dikirim oleh host pengirim



Gambar 3.28 TCP mengirim dan menerima buffer

unit transmisi, MTU), dan kemudian mengatur MSS untuk memastikan bahwa segmen TCP (ketika dienkapsulasi dalam datagram IP) ditambah panjang header TCP/IP (biasanya 40 byte) akan masuk ke dalam bingkai lapisan tautan tunggal. Baik protokol link-layer Ethernet maupun PPP memiliki MSS sebesar 1.500 byte. Pendekatan juga telah diusulkan untuk menemukan jalur MTU—bingkai lapisan tautan terbesar yang dapat dikirim pada semua tautan dari sumber ke tujuan [RFC 1191]—dan menyetel MSS berdasarkan nilai jalur MTU. Perhatikan bahwa MSS adalah jumlah maksimum data lapisan aplikasi dalam segmen, bukan ukuran maksimum segmen TCP termasuk header. (Istilah ini membingungkan, tetapi kita harus menerimanya, karena sudah tertanam dengan baik.)

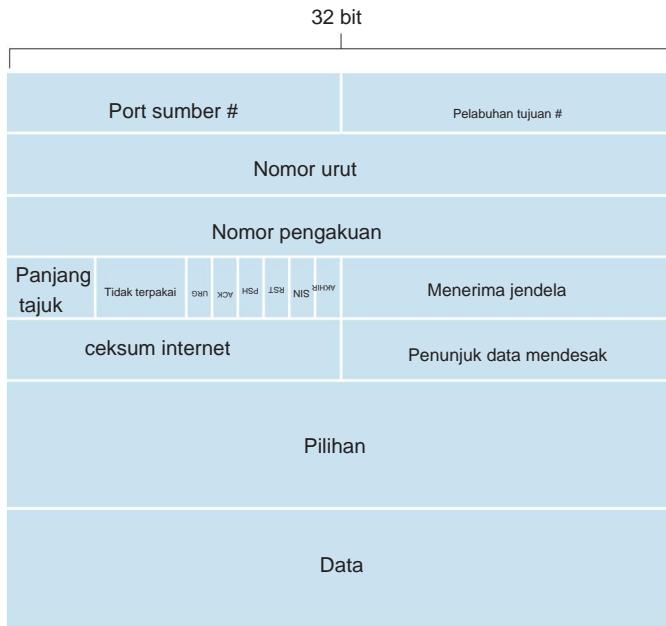
TCP memasangkan setiap potongan data klien dengan header TCP, sehingga membentuk **segmen TCP**. Segmen diturunkan ke lapisan jaringan, di mana mereka secara terpisah dienkapsulasi dalam datagram IP lapisan jaringan. Datagram IP kemudian dikirim ke jaringan. Ketika TCP menerima segmen di ujung yang lain, data segmen ditempatkan di buffer penerima koneksi TCP, seperti yang ditunjukkan pada Gambar 3.28. Aplikasi membaca aliran data dari buffer ini. Setiap sisi koneksi memiliki buffer kirim sendiri dan buffer penerima sendiri. (Anda dapat melihat applet kontrol aliran online di <http://www.awl.com/kurose-ross>, yang menyediakan animasi buffer kirim dan terima.)

Kita melihat dari diskusi ini bahwa koneksi TCP terdiri dari buffer, variabel, dan koneksi soket ke proses di satu host, dan satu set buffer, variabel, dan koneksi soket ke proses di host lain. Seperti disebutkan sebelumnya, tidak ada buffer atau variabel yang dialokasikan untuk koneksi di elemen jaringan (router, switch, dan repeater) antara host.

3.5.2 Struktur Segmen TCP

Setelah melihat sekilas koneksi TCP, mari kita periksa struktur segmen TCP. Segmen TCP terdiri dari bidang header dan bidang data. Bidang data berisi potongan data aplikasi. Seperti disebutkan di atas, MSS membatasi

234 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.29 Struktur segmen TCP

ukuran maksimum bidang data segmen. Ketika TCP mengirim file besar, seperti gambar sebagai bagian dari halaman Web, biasanya TCP memecah file menjadi potongan ukuran MSS (kecuali untuk potongan terakhir, yang seringkali lebih kecil dari MSS). Aplikasi interaktif, bagaimanapun, sering mengirimkan potongan data yang lebih kecil dari MSS; misalnya, dengan aplikasi login jarak jauh seperti Telnet, bidang data di segmen TCP seringkali hanya satu byte. Karena header TCP biasanya 20 byte (12 byte lebih banyak dari header UDP), segmen yang dikirim oleh Telnet mungkin hanya sepanjang 21 byte.

Gambar 3.29 menunjukkan struktur segmen TCP. Seperti UDP, header menyertakan **nomor port sumber dan tujuan**, yang digunakan untuk data multiplexing/demultiplexing dari/ke aplikasi lapisan atas. Juga, seperti UDP, header menyertakan **bidang checksum**. Header segmen TCP juga berisi bidang-bidang berikut:

- **Bidang nomor urut** 32-bit dan **bidang nomor pengakuan** 32-bit digunakan oleh pengirim dan penerima TCP dalam mengimplementasikan layanan transfer data yang andal, seperti dibahas di bawah ini.
- **Bidang jendela penerima** 16-bit digunakan untuk kontrol aliran. Kita akan segera melihat bahwa ini digunakan untuk menunjukkan jumlah byte yang bersedia diterima oleh penerima.
- **Kolom panjang header** 4-bit menentukan panjang header TCP dalam word 32-bit. Header TCP dapat memiliki panjang variabel karena bidang opsi TCP.

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 235

(Biasanya, bidang pilihan kosong, sehingga panjang header TCP tipikal adalah 20 byte.)

- **Bidang opsi** opsional dan panjang variabel digunakan saat pengirim dan penerima menegosiasikan ukuran segmen maksimum (MSS) atau sebagai faktor penskalaan jendela untuk digunakan dalam jaringan berkecepatan tinggi. Opsi stempel waktu juga ditentukan. Lihat RFC 854 dan RFC 1323 untuk detail tambahan.
- Bidang **bendera** berisi 6 bit. Bit **ACK** digunakan untuk menunjukkan bahwa nilai yang dibawa dalam field acknowledgment adalah valid; yaitu, segmen berisi pengakuan atas segmen yang telah berhasil diterima. Bit **RST**, **SYN**, dan **FIN** digunakan untuk pengaturan koneksi dan teardown, seperti yang akan kita bahas di akhir bagian ini. Menetapkan bit **PSH** menunjukkan bahwa penerima harus segera meneruskan data ke lapisan atas. Terakhir, bit **URG** digunakan untuk menunjukkan bahwa ada data di segmen ini yang telah ditandai oleh entitas lapisan atas sisi pengirim sebagai "mendesak". Lokasi byte terakhir dari data mendesak ini ditunjukkan oleh **bidang penunjuk data mendesak 16-bit**. TCP harus memberi tahu entitas lapisan atas sisi penerima ketika data mendesak ada dan meneruskannya sebagai penunjuk ke akhir data mendesak. (Dalam praktiknya, PSH, URG, dan penunjuk data mendesak tidak digunakan. Namun, kami menyebutkan bidang ini untuk kelengkapan.)

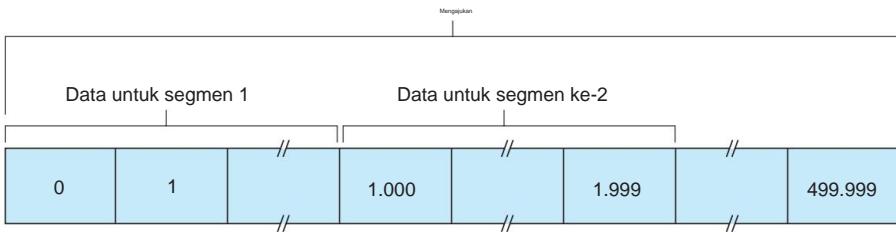
Nomor Urut dan Nomor Pengakuan

Dua bidang terpenting dalam header segmen TCP adalah bidang nomor urut dan bidang nomor pengakuan. Bidang-bidang ini adalah bagian penting dari layanan transfer data TCP yang andal. Namun sebelum membahas bagaimana bidang ini digunakan untuk menyediakan transfer data yang andal, pertama-tama mari kita jelaskan apa sebenarnya yang diletakkan TCP di bidang ini.

TCP memandang data sebagai aliran byte yang tidak terstruktur, tetapi teratur. Penggunaan nomor urut oleh TCP mencerminkan pandangan ini bahwa nomor urut berada di atas aliran byte yang ditransmisikan dan *bukan* di atas rangkaian segmen yang ditransmisikan. Oleh karena itu, nomor **urut untuk suatu segmen** adalah nomor aliran byte dari byte pertama dalam segmen tersebut. Mari kita lihat sebuah contoh. Misalkan sebuah proses di Host A ingin mengirim aliran data ke proses di Host B melalui koneksi TCP. TCP di Host A secara implisit akan menomori setiap byte dalam aliran data. Misalkan aliran data terdiri dari file yang terdiri dari 500.000 byte, MSS adalah 1.000 byte, dan byte pertama dari aliran data diberi nomor 0. Seperti yang ditunjukkan pada Gambar 3.30, TCP membangun 500 segmen dari aliran data. Segmen pertama diberi nomor urut 0, segmen kedua diberi nomor urut 1.000, segmen ketiga diberi nomor urut 2.000, dan seterusnya. Setiap nomor urut dimasukkan ke bidang nomor urut di header segmen TCP yang sesuai.

Sekarang mari kita pertimbangkan nomor pengakuan. Ini sedikit lebih rumit daripada nomor urut. Ingatlah bahwa TCP adalah full-duplex, sehingga Host A mungkin menerima data dari Host B saat mengirimkan data ke Host B (sebagai bagian dari koneksi TCP yang sama). Setiap segmen yang datang dari Host B memiliki nomor urut untuk datanya

236 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.30 Membagi data file menjadi segmen-segmen TCP

mengalir dari B ke A. *Nomor pengakuan yang ditempatkan Host A di segmennya adalah nomor urut byte berikutnya yang diharapkan Host A dari Host B.* Ada baiknya melihat beberapa contoh untuk memahami apa yang terjadi di sini. Misalkan Host A telah menerima semua byte bernomor 0 hingga 535 dari B dan misalkan akan mengirim segmen ke Host B. Host A sedang menunggu byte 536 dan semua byte berikutnya dalam aliran data Host B. Jadi Host A menempatkan 536 di bidang nomor pengakuan dari segmen yang dikirimkannya ke B.

Sebagai contoh lain, misalkan Host A telah menerima satu segmen dari Host B yang berisi byte 0 hingga 535 dan segmen lain yang berisi byte 900 hingga 1.000. Untuk beberapa alasan Host A belum menerima byte 536 hingga 899. Dalam contoh ini, Host A masih menunggu byte 536 (dan seterusnya) untuk membuat ulang aliran data B. Dengan demikian, segmen A berikutnya ke B akan berisi 536 di bidang nomor pengakuan. Karena TCP hanya mengakui byte hingga byte pertama yang hilang dalam aliran, TCP dikatakan memberikan **pengakuan kumulatif**.

Contoh terakhir ini juga memunculkan masalah yang penting namun tidak kentara. Host A menerima segmen ketiga (byte 900 hingga 1.000) sebelum menerima segmen kedua (byte 536 hingga 899). Dengan demikian, segmen ketiga tiba rusak. Masalah halusnya adalah: Apa yang dilakukan host ketika menerima segmen yang rusak dalam koneksi TCP? Menariknya, TCP RFC tidak memaksakan aturan apa pun di sini dan menyerahkan keputusan kepada orang yang memprogram implementasi TCP. Pada dasarnya ada dua pilihan: (1) penerima segera membuang segmen yang tidak sesuai pesanan (yang, seperti yang telah kita bahas sebelumnya, dapat menyederhanakan desain penerima), atau (2) penerima menyimpan byte yang tidak sesuai pesanan dan menunggu untuk byte yang hilang untuk mengisi kekosongan. Jelas, pilihan terakhir lebih efisien dalam hal bandwidth jaringan, dan merupakan pendekatan yang diambil dalam praktiknya.

Pada Gambar 3.30, kami mengasumsikan bahwa nomor urut awal adalah nol. Sebenarnya, kedua sisi koneksi TCP secara acak memilih nomor urut awal. Hal ini dilakukan untuk meminimalkan kemungkinan bahwa segmen yang masih ada di jaringan dari koneksi sebelumnya yang sudah dihentikan antara dua host disalahartikan sebagai segmen yang valid di koneksi selanjutnya antara dua host yang sama ini (yang juga kebetulan terjadi). menggunakan nomor port yang sama dengan koneksi lama) [Sunshine 1978].

Telnet: Studi Kasus untuk Urutan dan Nomor Pengakuan

Telnet, didefinisikan dalam RFC 854, adalah protokol lapisan aplikasi populer yang digunakan untuk login jarak jauh. Ini berjalan di atas TCP dan dirancang untuk bekerja di antara setiap pasangan host. Berbeda dengan aplikasi transfer data massal yang dibahas di Bab 2, Telnet adalah aplikasi interaktif. Kami membahas contoh Telnet di sini, karena dengan baik mengilustrasikan urutan TCP dan nomor pengakuan. Kami mencatat bahwa banyak pengguna sekarang lebih suka menggunakan protokol SSH daripada Telnet, karena data yang dikirim dalam koneksi Telnet (termasuk kata sandi!) tidak dienkripsi, membuat Telnet rentan terhadap serangan penyadapan (seperti dibahas di Bagian 8.7).

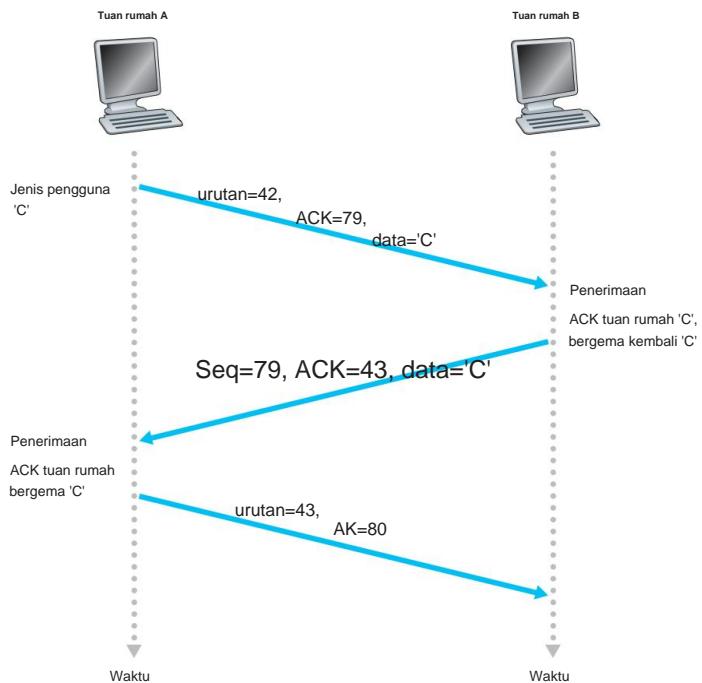
Misalkan Host A memulai sesi Telnet dengan Host B. Karena Host A memulai sesi, itu diberi label klien, dan Host B diberi label server. Setiap karakter yang diketik oleh pengguna (di klien) akan dikirim ke host jarak jauh; host jarak jauh akan mengirimkan kembali salinan dari setiap karakter, yang akan ditampilkan di layar pengguna Telnet. "Gema kembali" ini digunakan untuk memastikan bahwa karakter yang dilihat oleh pengguna Telnet telah diterima dan diproses di situs jarak jauh. Setiap karakter melintasi jaringan dua kali antara saat pengguna menekan tombol dan saat karakter ditampilkan di monitor pengguna.

Sekarang misalkan pengguna mengetik satu huruf, 'C,' dan kemudian mengambil kopi. Mari kita periksa segmen TCP yang dikirim antara klien dan server. Seperti yang ditunjukkan pada Gambar 3.31, kami menganggap nomor urut awal masing-masing adalah 42 dan 79 untuk klien dan server. Ingat bahwa nomor urut segmen adalah nomor urut byte pertama dalam bidang data. Dengan demikian, segmen pertama yang dikirim dari klien akan memiliki nomor urut 42; segmen pertama yang dikirim dari server akan memiliki nomor urut 79. Ingatlah bahwa nomor pengakuan adalah nomor urut dari byte data berikutnya yang ditunggu oleh host. Setelah koneksi TCP dibuat tetapi sebelum data dikirim, klien menunggu byte 79 dan server menunggu byte 42.

Seperti yang ditunjukkan pada Gambar 3.31, tiga segmen dikirim. Segmen pertama dikirim dari klien ke server, berisi representasi ASCII 1-byte dari huruf 'C' di bidang datanya. Segmen pertama ini juga memiliki 42 di bidang nomor urutnya, seperti yang baru saja kami jelaskan. Juga, karena klien belum menerima data apa pun dari server, segmen pertama ini akan memiliki 79 di bidang nomor pengakuannya.

Segmen kedua dikirim dari server ke klien. Ini melayani pose tujuan ganda. Pertama, ini memberikan pengakuan atas data yang telah diterima server. Dengan menempatkan 43 di bidang pengakuan, server memberi tahu klien bahwa ia telah berhasil menerima semuanya hingga byte 42 dan sekarang menunggu byte 43 dan seterusnya. Tujuan kedua dari segmen ini adalah untuk mengirimkan kembali huruf 'C.' Dengan demikian, segmen kedua memiliki representasi ASCII 'C' di bidang datanya. Segmen kedua ini memiliki nomor urut 79, nomor urut awal aliran data server-klien dari koneksi TCP ini, karena ini adalah byte data pertama yang dikirimkan server. Perhatikan bahwa pengakuan untuk data klien-ke-server dilakukan dalam segmen yang membawa data server-ke-klien; pengakuan ini dikatakan **membonceng** pada segmen data server-ke-klien.

238 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.31 Urutan dan nomor pengakuan untuk aplikasi Telnet sederhana melalui TCP

Segmen ketiga dikirim dari klien ke server. Tujuan utamanya adalah untuk mengakui data yang telah diterimanya dari server. (Ingat bahwa segmen kedua berisi data—huruf 'C'—dari server ke klien.) Segmen ini memiliki bidang data kosong (artinya, pengakuan tidak didukung dengan data klien-ke-server apa pun). Segmen tersebut memiliki 80 di bidang nomor pengakuan karena klien telah menerima aliran byte ke atas melalui nomor urutan byte 79 dan sekarang menunggu byte 80 dan seterusnya. Anda mungkin merasa aneh bahwa segmen ini juga memiliki nomor urut karena segmen tersebut tidak berisi data. Tetapi karena TCP memiliki bidang nomor urut, segmen tersebut perlu memiliki beberapa nomor urut.

3.5.3 Estimasi Waktu Pulang Pergi dan Timeout

TCP, seperti protokol rdt kami di Bagian 3.4, menggunakan mekanisme timeout/retransmit untuk memulihkan dari segmen yang hilang. Meskipun ini secara konseptual sederhana, banyak masalah halus muncul ketika kami menerapkan mekanisme batas waktu/transmisi ulang dalam protokol aktual seperti TCP. Mungkin pertanyaan yang paling jelas adalah lamanya waktu tunggu

interval. Jelas, timeout harus lebih besar dari round-trip time (RTT) koneksi, yaitu waktu dari saat segmen dikirim hingga diakui. Jika tidak, transmisi ulang yang tidak perlu akan dikirim. Tapi seberapa besar? Bagaimana seharusnya RTT diperkirakan? Haruskah pengatur waktu dikaitkan dengan setiap segmen yang tidak diakui? Begitu banyak pertanyaan! Pembahasan kita di bagian ini didasarkan pada pekerjaan TCP di [Jacobson 1988] dan rekomendasi IETF saat ini untuk mengatur waktu TCP [RFC 6298].

Memperkirakan Waktu Round-Trip

Mari kita mulai studi kita tentang manajemen pengatur waktu TCP dengan mempertimbangkan bagaimana TCP memperkirakan waktu bolak-balik antara pengirim dan penerima. Ini dicapai sebagai berikut. Sampel RTT, dilambangkan SampleRTT, untuk segmen adalah jumlah waktu antara saat segmen dikirim (yaitu diteruskan ke IP) dan saat pengakuan untuk segmen diterima. Alih-alih mengukur SampleRTT untuk setiap segmen yang ditransmisikan, sebagian besar implementasi TCP hanya mengambil satu pengukuran SampleRTT dalam satu waktu. Yaitu, kapan saja, SampleRTT diperkirakan hanya untuk satu segmen yang ditransmisikan tetapi saat ini tidak diakui, yang mengarah ke nilai baru SampleRTT kira-kira sekali setiap RTT. Juga, TCP tidak pernah menghitung SampleRTT untuk segmen yang telah ditransmisikan ulang; itu hanya mengukur SampleRTT untuk segmen yang telah ditransmisikan sekali [Karn 1987]. (Masalah di akhir bab meminta Anda mempertimbangkan alasannya.)

Jelas, nilai SampleRTT akan berfluktuasi dari segmen ke segmen karena kemacetan di router dan beban yang bervariasi pada sistem akhir. Karena fluktuasi ini, nilai SampleRTT apa pun yang diberikan mungkin tidak biasa. Untuk memperkirakan RTT tipikal, oleh karena itu wajar untuk mengambil semacam rata-rata nilai SampleRTT. TCP mempertahankan rata-rata, yang disebut EstimatedRTT, dari nilai SampleRTT. Setelah mendapatkan SampleRTT baru, TCP memperbarui EstimatedRTT menurut rumus berikut:

$$\text{EstimasiRTT} = (1 - \alpha) \cdot \text{EstimasiRTT} + \alpha \cdot \text{SampleRTT}$$

Rumus di atas ditulis dalam bentuk pernyataan bahasa pemrograman— nilai baru EstimatedRTT adalah kombinasi bobot dari nilai EstimatedRTT sebelumnya dan nilai baru untuk SampleRTT. Nilai yang disarankan dari adalah = 0,125 (yaitu, 1/8) [RFC 6298], dalam hal ini rumus di atas menjadi:

$$\text{EstimasiRTT} = 0,875 \cdot \text{EstimasiRTT} + 0,125 \cdot \text{SampleRTT}$$

Perhatikan bahwa EstimatedRTT adalah rata-rata tertimbang dari nilai SampleRTT. Seperti yang dibahas dalam soal pekerjaan rumah di akhir bab ini, bobot rata-rata usia ini memberi bobot lebih pada sampel baru daripada sampel lama. Ini wajar, sebagai

240 BAB 3 • LAPISAN TRANSPORTASI



PRINSIP DALAM PRAKTEK

TCP menyediakan transfer data yang andal dengan menggunakan acknowledgment positif dan timer dengan cara yang sama seperti yang kita pelajari di Bagian 3.4. TCP mengakui data yang telah diterima dengan benar, dan kemudian mentransmisi ulang segmen ketika segmen atau pengakuan yang sesuai dianggap hilang atau rusak. Versi tertentu dari TCP juga memiliki mekanisme NAK implisit—dengan mekanisme pengiriman ulang cepat TCP, penerimaan tiga ACK duplikat untuk segmen tertentu berfungsi sebagai NAK implisit untuk segmen berikutnya, yang memicu transmisi ulang segmen tersebut sebelum waktu habis. TCP menggunakan urutan angka untuk memungkinkan penerima mengidentifikasi segmen yang hilang atau duplikat. Sama seperti dalam kasus protokol transfer data kami yang andal, rdt3.0, TCP tidak dapat mengetahui dengan pasti apakah suatu segmen, atau ACK-nya, hilang, rusak, atau terlalu tertunda. Di pengirim, respons TCP akan sama: mengirimkan ulang segmen yang dimaksud.

TCP juga menggunakan perpipaan, yang memungkinkan pengirim memiliki banyak pengiriman tetapi belum segmen yang diakui beredar pada waktu tertentu. Kita telah melihat sebelumnya bahwa pipelining dapat sangat meningkatkan throughput sesi ketika rasio ukuran segmen terhadap penundaan pulang pergi kecil. Jumlah tertentu dari segmen luar biasa yang tidak diakui yang dapat dimiliki oleh pengirim ditentukan oleh mekanisme kontrol aliran dan kontrol kongesti TCP.

Kontrol aliran TCP dibahas pada akhir bagian ini; Kontrol kongesti TCP dibahas di Bagian 3.7. Untuk saat ini, kita harus menyadari bahwa pengirim TCP menggunakan perpipaan.

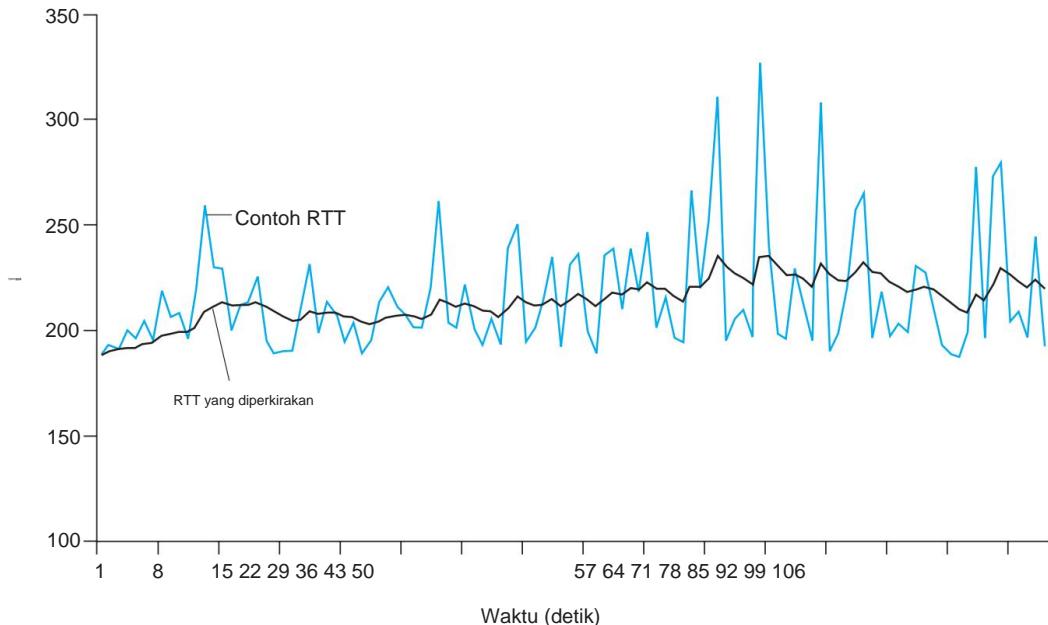
sampel yang lebih baru lebih mencerminkan kemacetan saat ini di jaringan. Dalam statistik, rata-rata seperti itu disebut **rata-rata bergerak tertimbang eksponensial (EWMA)**. Kata "eksponensial" muncul di EWMA karena bobot Sam pleRTT yang diberikan berkurang secara eksponensial dengan cepat saat pembaruan dilanjutkan. Dalam soal PR, Anda akan diminta untuk menurunkan suku eksponensial dalam EstimasiRTT.

Gambar 3.32 menunjukkan nilai SampleRTT dan EstimatedRTT untuk nilai $\gamma = 1/8$ untuk koneksi TCP antara gaia.cs.umass.edu (di Amherst, Massachu setts) ke fantasia.eurecom.fr (di selatan Perancis). Jelas, variasi dalam SampleRTT dihaluskan dalam perhitungan EstimasiRTT.

Selain memiliki perkiraan RTT, ada baiknya juga memiliki ukuran variabilitas RTT. [RFC 6298] mendefinisikan variasi RTT, DevRTT, sebagai perkiraan seberapa banyak SampleRTT biasanya menyimpang dari EstimatedRTT:

$$\text{DevRTT} = (1 - \gamma) \cdot \text{DevRTT} + \gamma | \text{SampleRTT} - \text{EstimasiRTT} |$$

Perhatikan bahwa DevRTT adalah EWMA dari perbedaan antara SampleRTT dan EstimatedRTT. Jika nilai SampleRTT sedikit berfluktuasi, maka DevRTT akan kecil; di sisi lain, jika banyak fluktuasi, DevRTT akan besar. Nilai γ yang disarankan adalah 0,25.



Gambar 3.32 Sampel RTT dan Estimasi RTT

Mengatur dan Mengelola Interval Timeout Pengiriman Ulang

Nilai yang diberikan dari EstimatedRTT dan DevRTT, nilai apa yang harus digunakan untuk interval waktu habis TCP? Jelas, interval harus lebih besar dari atau sama dengan EstimatedRTT, atau pengiriman ulang yang tidak perlu akan dikirim. Tetapi interval waktu tunggu tidak boleh terlalu besar dari EstimatedRTT; jika tidak, ketika sebuah segmen hilang, TCP tidak akan dengan cepat mentransmisi ulang segmen tersebut, menyebabkan penundaan transfer data yang besar. Oleh karena itu diinginkan untuk mengatur batas waktu sama dengan EstimasiRTT ditambah beberapa margin. Margin harus besar bila ada banyak fluktuasi nilai SampleRTT; itu harus kecil ketika ada sedikit fluktuasi. Nilai DevRTT seharusnya berperan di sini. Semua pertimbangan ini diperhitungkan dalam metode TCP untuk menentukan interval batas waktu transmisi ulang:

$$\text{TimeoutInterval} = \text{EstimasiRTT} + 4 \cdot \text{DevRTT}$$

Nilai TimeoutInterval awal 1 detik disarankan [RFC 6298]. Selain itu, saat timeout terjadi, nilai TimeoutInterval digandakan untuk menghindari timeout prematur yang terjadi untuk segmen berikutnya yang akan segera dikenali. Namun, segera setelah segmen diterima dan EstimatedRTT diperbarui, TimeoutInterval kembali dihitung menggunakan rumus di atas.

242 BAB 3 • LAPISAN TRANSPORTASI

3.5.4 Transfer Data yang Andal

Ingatlah bahwa layanan lapisan jaringan Internet (layanan IP) tidak dapat diandalkan. IP tidak menjamin pengiriman datagram, tidak menjamin pengiriman gram data secara berurutan, dan tidak menjamin integritas data dalam datagram. Dengan layanan IP, datagram dapat meluap dari buffer router dan tidak pernah mencapai tujuannya, datagram dapat tiba dengan rusak, dan bit dalam datagram dapat rusak (dibalik dari 0 ke 1 dan sebaliknya). Karena segmen transport-layer dibawa melintasi jaringan oleh datagram IP, segmen transport-layer juga dapat mengalami masalah ini.

TCP menciptakan **layanan transfer data yang andal** di atas layanan upaya terbaik IP yang tidak dapat diandalkan. Layanan transfer data TCP yang andal memastikan bahwa aliran data yang dibaca oleh proses dari buffer penerima TCP-nya tidak rusak, tanpa celah, tanpa duplikasi, dan berurutan; yaitu, aliran byte persis sama dengan aliran byte yang dikirim oleh sistem akhir di sisi lain koneksi. Bagaimana TCP menyediakan transfer data yang andal melibatkan banyak prinsip yang telah kita pelajari di Bagian 3.4.

Dalam pengembangan teknik transfer data andal kami sebelumnya, secara konseptual paling mudah untuk mengasumsikan bahwa pengatur waktu individu dikaitkan dengan setiap segmen yang ditransmisikan tetapi belum diakui. Meskipun ini bagus secara teori, manajemen pengatur waktu dapat memerlukan biaya tambahan yang cukup besar. Dengan demikian, prosedur manajemen pengatur waktu TCP yang direkomendasikan [RFC 6298] hanya menggunakan satu *pengatur waktu transmisi ulang*, bahkan jika ada beberapa segmen yang ditransmisikan tetapi belum diakui. Protokol TCP yang dijelaskan di bagian ini mengikuti rekomendasi pengatur waktu ini.

Kami akan membahas bagaimana TCP menyediakan transfer data yang andal dalam dua langkah bertahap. Kami pertama kali menyajikan deskripsi yang sangat disederhanakan dari pengirim TCP yang hanya menggunakan waktu tunggu untuk memulihkan dari segmen yang hilang; kami kemudian menyajikan deskripsi yang lebih lengkap yang menggunakan ucapan terima kasih duplikat selain batas waktu. Dalam diskusi berikutnya, kami menganggap bahwa data dikirim hanya dalam satu arah, dari Host A ke Host B, dan Host A mengirimkan file besar.

Gambar 3.33 menyajikan deskripsi yang sangat disederhanakan dari pengirim TCP. Kami melihat bahwa ada tiga kejadian utama yang terkait dengan transmisi data dan pengiriman ulang di pengirim TCP: data diterima dari aplikasi di atas; waktu habis waktu; dan tanda terima ACK. Setelah peristiwa besar pertama terjadi, TCP menerima data dari aplikasi, mengenkapsulasi data dalam sebuah segmen, dan meneruskan segmen tersebut ke IP. Perhatikan bahwa setiap segmen menyertakan nomor urut yang merupakan nomor aliran byte dari byte data pertama dalam segmen, seperti yang dijelaskan di Bagian 3.5.2. Perhatikan juga bahwa jika pengatur waktu sudah tidak berjalan untuk beberapa segmen lain, TCP memulai pengatur waktu saat segmen diteruskan ke IP. (Akan sangat membantu jika menganggap pengatur waktu terkait dengan segmen tertua yang belum diakui.) Interval kedaluwarsa untuk pengatur waktu ini adalah TimeoutInterval, yang dihitung dari EstimatedRTT dan DevRTT, seperti yang dijelaskan di Bagian 3.5.3.

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 243

```

/* Asumsikan pengirim tidak dibatasi oleh aliran TCP atau kontrol kongesti, bahwa data dari atas berukuran lebih kecil dari MSS, dan bahwa transfer data hanya dalam satu arah. */

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (selamanya)
{ switch(event)

    event: data yang diterima dari aplikasi di atas
        buat segmen TCP dengan nomor urut NextSeqNum jika (timer saat ini tidak
        berjalan)
        mulai pengatur waktu
        meneruskan segmen ke IP
        NextSeqNum=NextSeqNum+panjang(data) istirahat;

    acara: waktu habis waktu
        mentransmisikan ulang segmen yang belum diakui dengan nomor urut
        terkecil
        mulai pengatur waktu
        merusak;

    acara: ACK diterima, dengan nilai bidang ACK y if (y > SendBase) {

        SendBase=y if
        (saat ini ada segmen yang belum diakui)
        mulai pengatur waktu
    }
    merusak;

} /* akhir perulangan selamanya */

```

Gambar 3.33 Pengirim TCP yang disederhanakan

Peristiwa besar kedua adalah batas waktu. TCP merespons acara batas waktu dengan mentransmisi ulang segmen yang menyebabkan timeout. TCP kemudian me-restart timer.

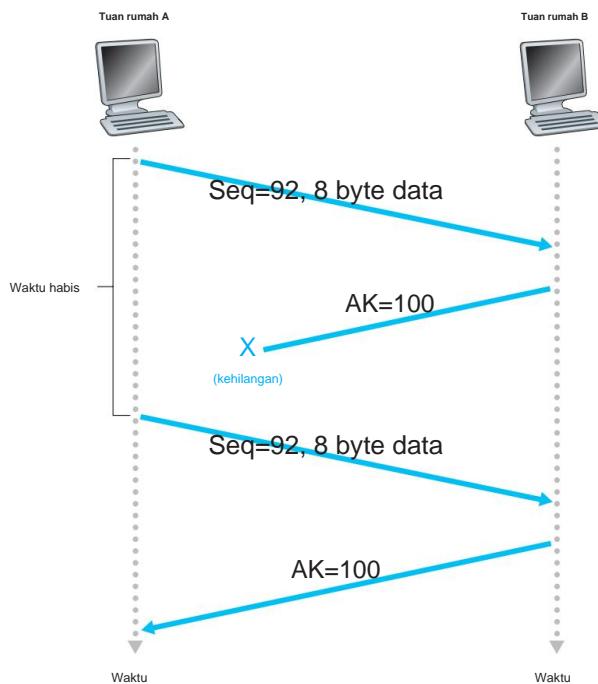
Peristiwa besar ketiga yang harus ditangani oleh pengirim TCP adalah kedatangan segmen pengakuan (ACK) dari penerima (lebih khusus lagi, segmen yang berisi nilai bidang ACK yang valid). Pada kejadian ini, TCP membandingkan nilai ACK y dengan variabel SendBase. Variabel status TCP SendBase adalah nomor urut byte terlama yang belum diakui. (Dengan demikian SendBase-1 adalah nomor urut byte terakhir yang diketahui telah diterima dengan benar dan berurutan di penerima.) Seperti yang ditunjukkan sebelumnya, TCP menggunakan acknowledgment kumulatif, sehingga y mengakui penerimaan semua byte sebelum nomor byte y. Jika y > SendBase,

244 BAB 3 • LAPISAN TRANSPORTASI

maka ACK mengakui satu atau lebih segmen yang sebelumnya tidak diakui. Jadi pengirim memperbarui variabel SendBase-nya; itu juga me-restart timer jika saat ini ada segmen yang belum diakui.

Beberapa Skenario Menarik

Kami baru saja menjelaskan versi yang sangat disederhanakan tentang bagaimana TCP menyediakan transfer data yang andal. Tetapi bahkan versi yang sangat disederhanakan ini memiliki banyak kehalusan. Untuk memahami cara kerja protokol ini, mari kita telusuri beberapa skenario sederhana. Gambar 3.34 menggambarkan skenario pertama, di mana Host A mengirim satu segmen ke Host B. Misalkan segmen ini memiliki nomor urut 92 dan berisi 8 byte data. Setelah mengirimkan segmen ini, Host A menunggu segmen dari B dengan nomor pengakuan 100. Meskipun segmen dari A diterima di B, pengakuan dari B ke A hilang. Dalam hal ini, peristiwa batas waktu terjadi, dan Host A mentransmisikan ulang segmen yang sama. Tentu saja, ketika Host B menerima transmisi ulang, ia mengamati dari nomor urut bahwa segmen tersebut berisi data yang telah diterima. Jadi, TCP di Host B akan membuang byte di segmen yang ditransmisikan ulang.

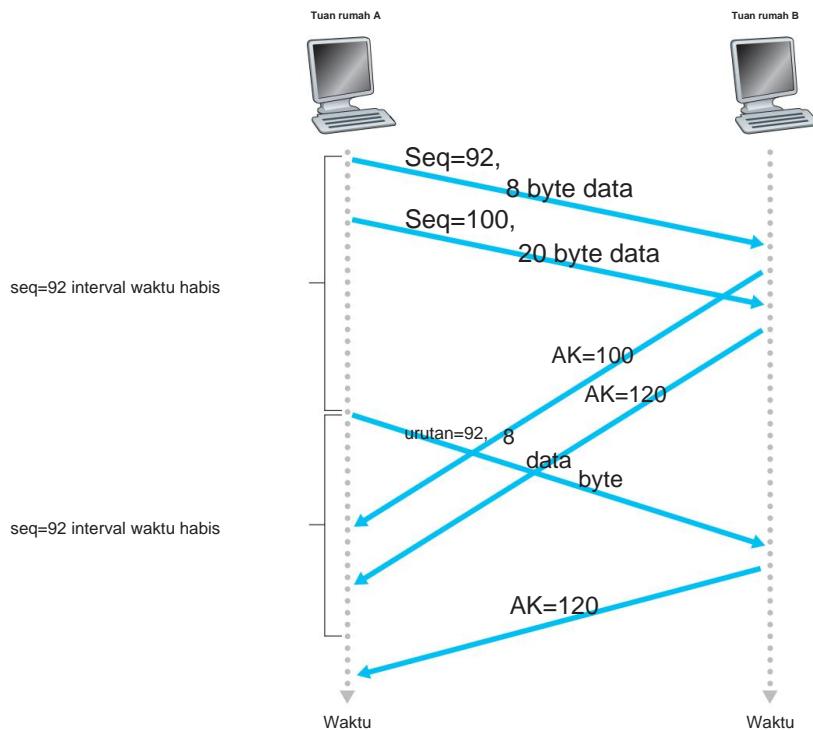


Gambar 3.34 Pengiriman ulang karena pengakuan yang hilang

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 245

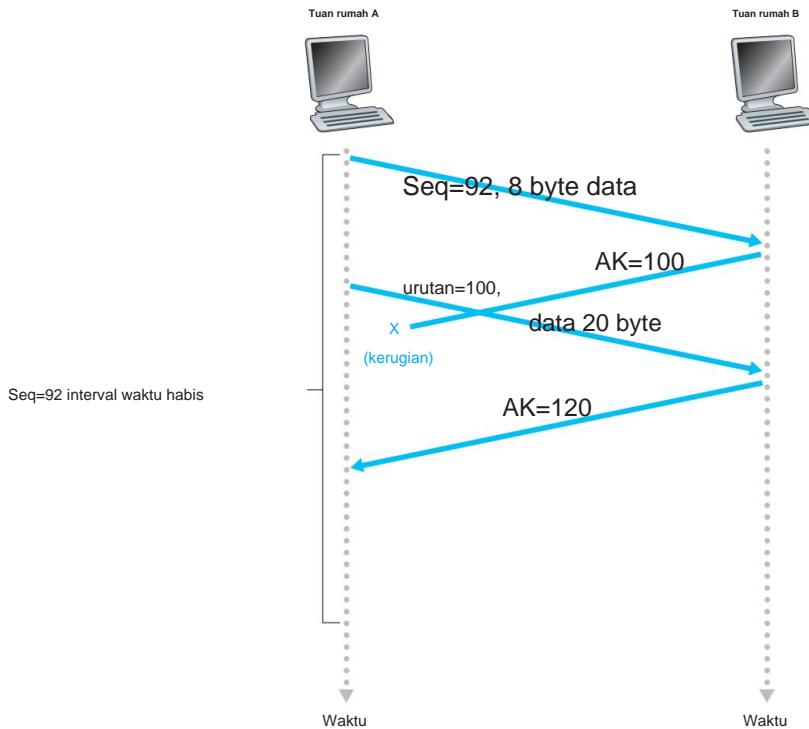
Dalam skenario kedua, ditunjukkan pada Gambar 3.35, Host A mengirimkan dua segmen secara berurutan. Segmen pertama memiliki nomor urut 92 dan 8 byte data, dan segmen kedua memiliki nomor urut 100 dan 20 byte data. Misalkan kedua segmen tiba utuh di B, dan B mengirimkan dua pengakuan terpisah untuk masing-masing segmen ini. Yang pertama dari ucapan terima kasih ini memiliki nomor pengakuan 100; yang kedua memiliki nomor pengakuan 120. Misalkan sekarang tak satu pun dari tepi pengakuan tiba di Host A sebelum batas waktu. Saat peristiwa batas waktu terjadi, Host A mengirim ulang segmen pertama dengan nomor urut 92 dan memulai ulang pengatur waktu. Selama ACK untuk segmen kedua tiba sebelum batas waktu yang baru, segmen kedua tidak akan dipancarkan ulang.

Dalam skenario ketiga dan terakhir, misalkan Host A mengirimkan dua segmen, persis seperti pada contoh kedua. Acknowledgment dari segmen pertama hilang dalam jaringan, tetapi tepat sebelum timeout event, Host A menerima acknowledgment dengan nomor acknowledgment 120. Oleh karena itu, Host A mengetahui bahwa Host B telah menerima *semuanya* hingga byte 119; jadi Host A tidak mengirim ulang salah satu dari kedua segmen tersebut. Skenario ini dilustrasikan pada Gambar 3.36.



Gambar 3.35 Segmen 100 tidak dipancarkan ulang

246 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.36 Pengakuan kumulatif menghindari pengiriman ulang segmen pertama

Menggandakan Interval Timeout

Kami sekarang membahas beberapa modifikasi yang diterapkan oleh sebagian besar implementasi TCP. Yang pertama menyangkut panjang interval waktu habis setelah waktu habis. Dalam modifikasi ini, setiap kali peristiwa timeout terjadi, TCP mentransmisikan ulang segmen yang belum diakui dengan nomor urut terkecil, seperti dijelaskan di atas. Tetapi setiap kali TCP mentransmisi ulang, ia menyetel interval waktu habis berikutnya menjadi dua kali nilai sebelumnya, daripada menurunkannya dari EstimatedRTT dan DevRTT terakhir (seperti yang dijelaskan dalam Bagian 3.5.3). Misalnya, TimeoutInterval yang terkait dengan segmen terlama yang belum diakui adalah 0,75 detik saat penghitung waktu pertama kali kedaluwarsa.

TCP kemudian akan mentransmisi ulang segmen ini dan menyetel waktu kedaluwarsa baru menjadi 1,5 detik. Jika pengatur waktu berakhir lagi 1,5 detik kemudian, TCP akan mengirim ulang segmen ini lagi, sekarang menyetel waktu kedaluwarsa menjadi 3,0 detik. Dengan demikian interval tumbuh secara eksponensial setelah setiap transmisi ulang. Namun, kapan pun pengatur waktu dimulai setelah salah satu dari dua peristiwa lainnya (yaitu, data diterima dari aplikasi di atas, dan ACK diterima),

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 247

TimeoutInterval diturunkan dari nilai terbaru EstimatedRTT dan DevRTT.

Modifikasi ini memberikan bentuk kontrol kemacetan yang terbatas. (Bentuk kontrol kongesti TCP yang lebih komprehensif akan dipelajari di Bagian 3.7.) Kedaluwarsa timer kemungkinan besar disebabkan oleh kongesti di jaringan, yaitu terlalu banyak paket yang tiba di satu (atau lebih) antrian router di jalur antara sumber dan tujuan, menyebabkan paket dijatuhkan dan/atau penundaan antrian yang lama. Pada saat kemacetan, jika sumber terus mentransmisikan ulang paket secara terus-menerus, kemacetan bisa menjadi lebih buruk. Sebaliknya, TCP bertindak lebih sopan, dengan setiap pengirim melakukan transmisi ulang setelah interval yang semakin lama. Kita akan melihat bahwa ide serupa digunakan oleh Ethernet ketika kita mempelajari CSMA/CD di Bab 5.

Pengiriman Ulang Cepat

Salah satu masalah dengan transmisi ulang yang dipicu timeout adalah bahwa periode timeout bisa relatif lama. Ketika sebuah segmen hilang, periode timeout yang lama ini memaksa pengirim untuk menunda pengiriman ulang paket yang hilang, sehingga meningkatkan penundaan end-to-end. Untungnya, pengirim seringkali dapat mendeteksi kehilangan paket dengan baik sebelum peristiwa time out terjadi dengan mencatat apa yang disebut ACK duplikat. ACK **duplicat** adalah ACK yang mengenali kembali segmen yang pengirimnya telah menerima pengakuan sebelumnya. Untuk memahami respon pengirim terhadap duplikat ACK, kita harus melihat mengapa penerima mengirimkan duplikat ACK. Tabel 3.2 merangkum kebijakan pembangkitan ACK penerima TCP [RFC 5681]. Ketika penerima TCP menerima segmen dengan nomor urut yang lebih besar dari nomor urut berikutnya yang diharapkan, ia mendeteksi celah dalam aliran data—yaitu, segmen yang hilang. Kesenjangan ini bisa jadi merupakan hasil dari segmen yang hilang atau disusun ulang dalam jaringan.

Peristiwa	Tindakan Penerima TCP
Kedatangan segmen in-order dengan nomor urut yang diharapkan. Semua data hingga nomor urut yang diharapkan sudah diakui.	ACK tertunda. Tunggu hingga 500 msec untuk kedatangan segmen in-order lainnya . Jika segmen pesanan berikutnya tidak sampai dalam interval ini, kirimkan ACK.
Kedatangan segmen in-order dengan nomor urut yang diharapkan. Satu segmen in-order lainnya menunggu transmisi ACK.	Segera kirim ACK kumulatif tunggal, ACK kedua segmen dalam urutan.
Kedatangan segmen out-of-order dengan nomor urut yang lebih tinggi dari perkiraan. Celah terdeteksi.	Segera kirim duplikat ACK, yang menunjukkan nomor urut byte yang diharapkan berikutnya (yang merupakan ujung bawah celah).
Kedatangan segmen yang sebagian atau seluruhnya mengisi celah pada data yang diterima.	Segera kirim ACK, asalkan segmen dimulai di ujung bawah gap.

Tabel 3.2 Rekomendasi Pembuatan TCP ACK [RFC 5681]

248 BAB 3 • LAPISAN TRANSPORTASI

Karena TCP tidak menggunakan acknowledgment negatif, penerima tidak dapat mengirimkan acknowledgment negatif eksplisit kembali ke pengirim. Alih-alih, itu hanya mengakui kembali tepi (yaitu, menghasilkan duplikat ACK untuk) byte data terakhir yang telah diterimanya. (Perhatikan bahwa Tabel 3.2 memungkinkan penerima tidak membuang segmen yang rusak.)

Karena pengirim sering mengirimkan sejumlah besar segmen secara berurutan, jika satu segmen hilang, kemungkinan besar akan ada banyak ACK duplikat yang berurutan. Jika pengirim TCP menerima tiga ACK duplikat untuk data yang sama, hal ini dianggap sebagai indikasi bahwa segmen yang mengikuti segmen yang telah ACK tiga kali telah hilang. (Dalam masalah pekerjaan rumah, kami mempertimbangkan pertanyaan mengapa pengirim menunggu tiga ACK duplikat, bukan hanya satu duplikat ACK.) Dalam kasus tiga ACK duplikat diterima, pengirim TCP melakukan pengiriman ulang cepat [RFC 5681], mentransmisikan ulang segmen yang hilang *sebelum* pengatur waktu segmen tersebut berakhir. Hal ini ditunjukkan pada Gambar 3.37, di mana segmen kedua hilang, kemudian ditransmisikan ulang sebelum waktunya habis. Untuk TCP dengan pengiriman ulang cepat, cuplikan kode berikut menggantikan kejadian yang diterima ACK pada Gambar 3.33:

```

acara: ACK diterima, dengan nilai bidang ACK y
if (y > SendBase) {
    SendBase=y if
    (saat ini ada segmen yang belum diakui)

    mulai pengatur waktu
}
else { /* duplikat ACK untuk segmen yang sudah di-ACK */ pertambahan
    jumlah ACK duplikat

    diterima untuk y if
    (jumlah duplikat ACKS diterima
     untuk y==3)
    /* TCP mengirim ulang dengan cepat */
    kirim ulang segmen dengan nomor urut y
}
merusak;

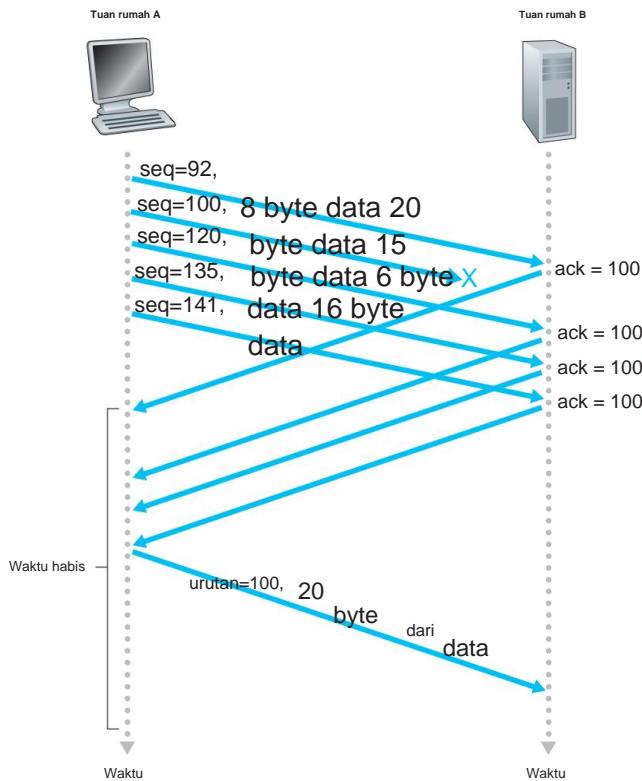
```

Kami mencatat sebelumnya bahwa banyak masalah halus muncul ketika mekanisme timeout/retransmit diimplementasikan dalam protokol aktual seperti TCP. Prosedur di atas, yang telah berevolusi sebagai hasil dari pengalaman lebih dari 20 tahun dengan pengatur waktu TCP, harus meyakinkan Anda bahwa memang demikian!

Go-Back-N atau Selective Repeat?

Mari kita tutup studi kita tentang mekanisme pemulihan kesalahan TCP dengan mempertimbangkan pertanyaan berikut: Apakah TCP adalah protokol GBN atau SR? Ingatlah bahwa pengakuan TCP bersifat kumulatif dan diterima dengan benar tetapi segmen yang rusak tidak secara individual

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 249



Gambar 3.37 Pengiriman ulang cepat: pengiriman ulang segmen yang hilang sebelum pengatur waktu segmen berakhir

ACK oleh penerima. Akibatnya, seperti yang ditunjukkan pada Gambar 3.33 (lihat juga Gambar 3.19), pengirim TCP hanya perlu mempertahankan nomor urut terkecil dari byte yang ditransmisikan tetapi tidak diakui (SendBase) dan nomor urut byte berikutnya yang akan dikirim (NextSeqNum). Dalam pengertian ini, TCP sangat mirip dengan protokol gaya GBN. Namun ada beberapa perbedaan mencolok antara TCP dan Go-Back-N. Banyak implementasi TCP akan menyangga segmen yang diterima dengan benar tetapi rusak [Stevens 1994]. Pertimbangkan juga apa yang terjadi ketika pengirim mengirimkan rangkaian segmen 1, 2, ..., N, dan semua segmen tiba secara berurutan tanpa kesalahan pada penerima. Lebih lanjut misalkan bahwa pengakuan untuk paket $n < N$ hilang, tetapi sisa pengakuan $N - 1$ tiba di pengirim sebelum waktu habis masing-masing. Dalam contoh ini, GBN akan mentransmisi ulang tidak hanya paket n , tetapi juga semua paket berikutnya $n + 1, n + 2, \dots, N$. TCP, sebaliknya, akan mentransmisi ulang paling banyak satu segmen, yaitu, segmen n . Selain itu, TCP bahkan tidak akan mentransmisi ulang segmen n jika pengakuan untuk segmen $n + 1$ tiba sebelum waktu habis untuk segmen n .

250 BAB 3 • LAPISAN TRANSPORTASI

Modifikasi yang diusulkan untuk TCP, yang disebut **pengakuan selektif** [RFC 2018], memungkinkan penerima TCP untuk mengenali segmen yang tidak sesuai pesanan secara selektif daripada hanya mengakui secara kumulatif penerimaan terakhir yang diterima dengan benar, dalam segmen pesanan. Ketika digabungkan dengan transmisi ulang selektif—melewati transmisi ulang segmen yang telah diakui secara selektif oleh penerima—TCP sangat mirip dengan protokol SR umum kami. Dengan demikian, mekanisme pemulihan kesalahan TCP mungkin paling baik dikategorikan sebagai gabungan dari protokol GBN dan SR.

3.5.5 Kontrol Aliran

Ingatlah bahwa host di setiap sisi koneksi TCP menyediakan buffer penerima untuk koneksi tersebut. Ketika koneksi TCP menerima byte yang benar dan berurutan, ia menempatkan data dalam buffer penerima. Proses aplikasi terkait akan membaca data dari buffer ini, tetapi belum tentu pada saat data tiba.

Memang, aplikasi penerima mungkin sibuk dengan beberapa tugas lain dan bahkan mungkin tidak mencoba membaca data sampai lama setelah tiba. Jika aplikasi relatif lambat dalam membaca data, pengirim dapat dengan mudah meluapkan buffer penerima koneksi dengan mengirim terlalu banyak data terlalu cepat.

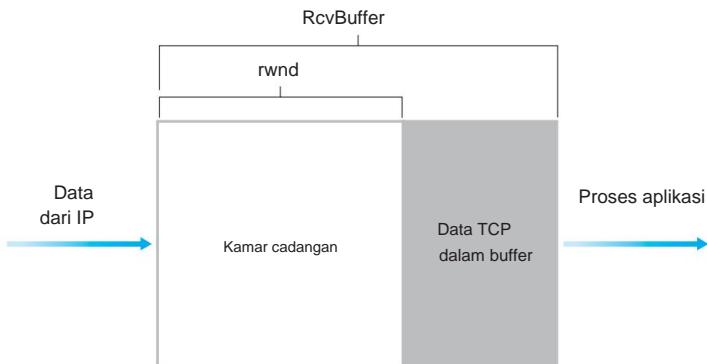
TCP menyediakan **layanan flow-control** untuk aplikasinya untuk menghilangkan kemungkinan pengirim meluap dari buffer penerima. Dengan demikian, kontrol aliran adalah layanan pencocokan kecepatan — mencocokkan kecepatan pengiriman pengirim dengan kecepatan pembacaan aplikasi penerima. Seperti disebutkan sebelumnya, pengirim TCP juga dapat dicekik karena kemacetan di dalam jaringan IP; bentuk kontrol pengirim ini disebut sebagai **kontrol kongesti**, sebuah topik yang akan kita bahas secara rinci di Bagian 3.6 dan 3.7. Meskipun tindakan yang diambil oleh kontrol aliran dan kemacetan serupa (pembatasan pengirim), mereka jelas diambil untuk alasan yang sangat berbeda. Sayangnya, banyak penulis menggunakan istilah tersebut secara bergantian, dan pembaca yang cerdas akan bijaksana untuk membedakannya. Sekarang mari kita bahas bagaimana TCP menyediakan layanan kontrol alirannya. Untuk melihat hutan pohon, kami mengandaikan seluruh bagian ini bahwa implementasi TCP sedemikian rupa sehingga penerima TCP membuang segmen yang rusak.

TCP menyediakan kontrol aliran dengan meminta *pengirim* memelihara variabel yang disebut **jendela penerima**. Secara informal, jendela penerima digunakan untuk memberikan gambaran kepada pengirim tentang berapa banyak ruang buffer kosong yang tersedia di penerima. Karena TCP adalah dupleks penuh, pengirim di setiap sisi koneksi mempertahankan jendela penerimaan yang berbeda. Mari selidiki jendela terima dalam konteks transfer file. Misalkan Host A mengirim file besar ke Host B melalui koneksi TCP. Host B mengalokasikan buffer penerima ke koneksi ini; menunjukkan ukurannya dengan RcvBuffer. Dari waktu ke waktu, proses aplikasi di Host B membaca dari buffer. Tentukan variabel berikut:

- LastByteRead: jumlah byte terakhir dalam aliran data yang dibaca dari buffer oleh proses aplikasi di B
- LastByteRcvd: jumlah byte terakhir dalam aliran data yang telah tiba

dari jaringan dan telah ditempatkan di buffer penerima di B

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 251



Gambar 3.38 Jendela terima (rwnd) dan buffer terima (RcvBuffer)

Karena TCP tidak diizinkan meluap buffer yang dialokasikan, kita harus memiliki ruang cadangan

LastByteRcvd – LastByteRead RcvBuffer

Jendela terima, dilambangkan dengan rwnd diatur ke jumlah ruang cadangan di buffer:

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

Karena kamar cadangan berubah seiring waktu, rwnd bersifat dinamis. Variabel rwnd diilustrasikan pada Gambar 3.38.

Bagaimana koneksi menggunakan variabel RWND untuk menyediakan layanan kontrol aliran? Host B memberi tahu Host A berapa banyak ruang kosong yang dimilikinya di buffer koneksi dengan menempatkan nilai rwnd saat ini di bidang jendela terima dari setiap segmen yang dikirimnya ke A. Awalnya, Host B menetapkan $\text{rwnd} = \text{RcvBuffer}$. Perhatikan bahwa untuk melakukan ini, Host B harus melacak beberapa variabel khusus koneksi.

Host A pada gilirannya melacak dua variabel, LastByteSent dan Last ByteAcked, yang memiliki arti yang jelas. Perhatikan bahwa perbedaan antara kedua variabel ini, $\text{LastByteSent} - \text{LastByteAcked}$, adalah jumlah data tidak diketahui yang telah dikirimkan A ke dalam koneksi. Dengan menjaga jumlah data yang tidak diakui kurang dari nilai rwnd, Host A yakin bahwa itu tidak meluap buffer penerima di Host B. Dengan demikian, Host A memastikan sepanjang hidup koneksi itu

$\text{LastByteSent} - \text{LastByteAcked}$ rwnd

252 BAB 3 • LAPISAN TRANSPORTASI

Ada satu masalah teknis kecil dengan skema ini. Untuk melihat ini, misalkan buffer penerima Host B menjadi penuh sehingga $rwnd = 0$. Setelah mengiklankan $rwnd = 0$ ke Host A, misalkan juga B tidak mengirim apa pun ke A. Sekarang pertimbangkan apa yang terjadi. Karena proses aplikasi di B mengosongkan buffer, TCP tidak mengirimkan segmen baru dengan nilai $rwnd$ baru ke Host A; memang, TCP mengirimkan segmen ke Host A hanya jika memiliki data untuk dikirim atau jika memiliki pengakuan untuk dikirim. Oleh karena itu, Host A tidak pernah diberi tahu bahwa beberapa ruang telah dibuka di buffer penerima Host B—Host A diblokir dan tidak dapat mengirimkan data lagi! Untuk mengatasi masalah ini, spesifikasi TCP mengharuskan Host A untuk terus mengirim segmen dengan satu byte data ketika jendela penerimaan B adalah nol. Segmen ini akan diakui oleh penerima. Akhirnya buffer akan mulai kosong dan ucapan terima kasih akan berisi nilai bukan nol.

Situs online di <http://www.awl.com/kurose-ross> untuk buku ini menyediakan sebuah applet Java interaktif yang mengilustrasikan pengoperasian jendela penerimaan TCP.

Setelah menggambarkan layanan kontrol aliran TCP, kami secara singkat menyebutkan di sini bahwa UDP tidak menyediakan kontrol aliran. Untuk memahami masalah ini, pertimbangkan untuk mengirimkan serangkaian segmen UDP dari proses di Host A ke proses di Host B. Untuk implementasi UDP tipikal, UDP akan menambahkan segmen dalam buffer berukuran terbatas yang “mendahului” soket yang sesuai (yaitu, pintu menuju proses). Proses membaca seluruh segmen sekaligus dari buffer. Jika proses tidak membaca segmen dengan cukup cepat dari buffer, buffer akan meluap dan segmen akan dibuang.

3.5.6 Manajemen Koneksi TCP

Dalam subbagian ini kita melihat lebih dekat bagaimana koneksi TCP dibuat dan dihancurkan. Meskipun topik ini mungkin tidak terlalu menarik, ini penting karena pembentukan koneksi TCP dapat secara signifikan menambah penundaan yang dirasakan (misalnya, saat menjelajahi Web). Selain itu, banyak dari serangan jaringan yang paling umum—termasuk serangan banjir SYN yang sangat populer—mengeksloitasi kerentanan dalam manajemen koneksi TCP. Pertama mari kita lihat bagaimana koneksi TCP dibuat. Misalkan sebuah proses yang berjalan di satu host (klien) ingin memulai koneksi dengan proses lain di host lain (server). Proses aplikasi klien pertama-tama memberi tahu TCP klien bahwa ia ingin membuat koneksi ke proses di server. TCP di klien kemudian mulai membuat koneksi TCP dengan TCP di server dengan cara berikut:

- *Langkah 1.* TCP sisi-klien pertama-tama mengirimkan segmen TCP khusus ke TCP sisi-server. Segmen khusus ini tidak berisi data lapisan aplikasi. Tapi salah satu bit flag di header segmen (lihat Gambar 3.29), bit SYN, diset ke 1. Untuk alasan ini, segmen khusus ini disebut sebagai segmen SYN. Selain itu, klien secara acak memilih nomor urut awal (`client_isn`) dan menempatkan nomor ini di bidang nomor urut segmen TCP SYN awal. Segmen ini dikemas dalam datagram IP dan dikirim ke server. Disitu ada

3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 253

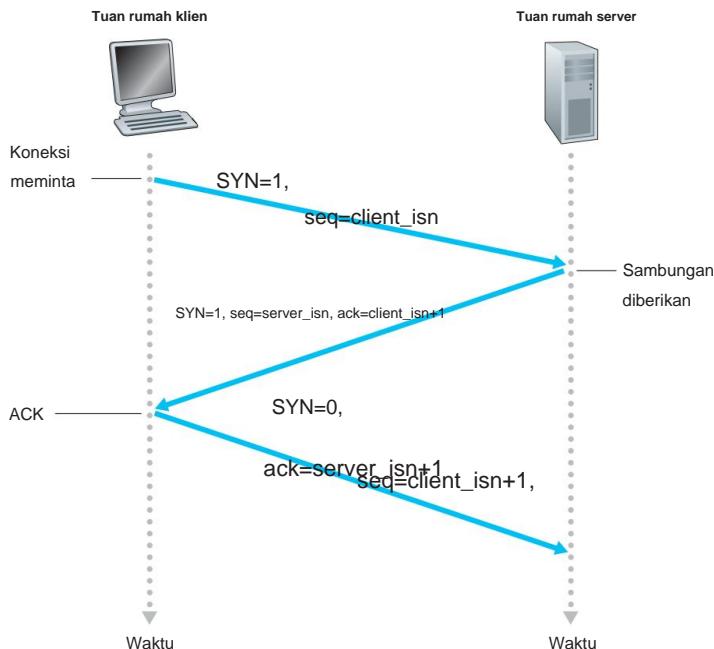
sangat tertarik untuk mengacak pilihan client_isn dengan benar untuk menghindari serangan keamanan tertentu [CERT 2001–09]. • *Langkah 2.* Setelah datagram IP yang berisi segmen TCP SYN tiba di host server (dengan asumsi memang tiba!), server mengekstrak segmen TCP SYN dari datagram, mengalokasikan buffer dan variabel TCP ke koneksi, dan mengirimkan segmen yang diberikan koneksi ke klien TCP. (Kita akan melihat di Bab 8 bahwa alokasi buffer dan variabel ini sebelum menyelesaikan langkah ketiga dari jabat tangan tiga arah membuat TCP rentan terhadap serangan denial-of-service yang dikenal sebagai banjir SYN.) Segmen yang diberikan koneksi ini juga tidak berisi data lapisan aplikasi. Namun, itu berisi tiga informasi penting di header segmen. Pertama, bit SYN diatur ke 1. Kedua, bidang pengakuan dari header segmen TCP diatur ke client_isn+1. Terakhir, server memilih nomor urut awalnya sendiri (server_isn) dan menempatkan nilai ini di bidang nomor urut header segmen TCP. Segmen yang diberikan koneksi ini mengatakan, pada dasarnya, “Saya menerima paket SYN Anda untuk memulai koneksi dengan nomor urut awal Anda, client_isn. Saya setuju untuk menjalin hubungan ini. Nomor urut awal saya sendiri adalah server_isn.” Segmen koneksi yang diberikan disebut sebagai **segmen SYNACK**.

- *Langkah 3.* Setelah menerima segmen SYNACK, klien juga mengalokasikan buffer dan variabel ke koneksi. Host klien kemudian mengirimkan segmen lain ke server; segmen terakhir ini mengakui segmen yang diberikan koneksi server (klien) melakukannya dengan meletakkan nilai server_isn+1 di bidang tepi pengakuan dari header segmen TCP. Bit SYN diatur ke nol, karena koneksi dibuat. Tahap ketiga dari jabat tangan tiga arah ini dapat membawa data klien-ke-server dalam muatan segmen.

Setelah ketiga langkah ini diselesaikan, host klien dan server dapat saling mengirim segmen yang berisi data. Di setiap segmen masa depan ini, bit SYN akan disetel ke nol. Perhatikan bahwa untuk membangun koneksi, tiga paket dikirim antara dua host, seperti yang diilustrasikan pada Gambar 3.39. Untuk alasan ini, prosedur pembuatan koneksi ini sering disebut sebagai **jabat tangan tiga arah**. Beberapa aspek dari jabat tangan tiga arah TCP dieksplorasi dalam masalah pekerjaan rumah (Mengapa nomor urut awal diperlukan? Mengapa jabat tangan tiga arah, bukan jabat tangan dua arah, diperlukan?). Sangat menarik untuk dicatat bahwa pemanjat tebing dan belayer (yang ditempatkan di bawah pemanjat tebing dan yang tugasnya menangani tali pengaman pemanjat) menggunakan protokol komunikasi jabat tangan tiga arah yang identik dengan TCP untuk memastikan bahwa kedua belah pihak sudah siap sebelum pendaki memulai pendakian.

Semua hal baik harus berakhir, dan hal yang sama berlaku dengan koneksi TCP. Salah satu dari dua proses yang berpartisipasi dalam koneksi TCP dapat mengakhiri koneksi. Saat koneksi berakhir, "sumber daya" (yaitu, buffer dan variabel) di host tidak dialokasikan. Sebagai contoh, misalkan klien memutuskan untuk menutup koneksi, seperti yang ditunjukkan pada Gambar 3.40. Masalah proses aplikasi klien ditutup

254 BAB 3 • LAPISAN TRANSPORTASI

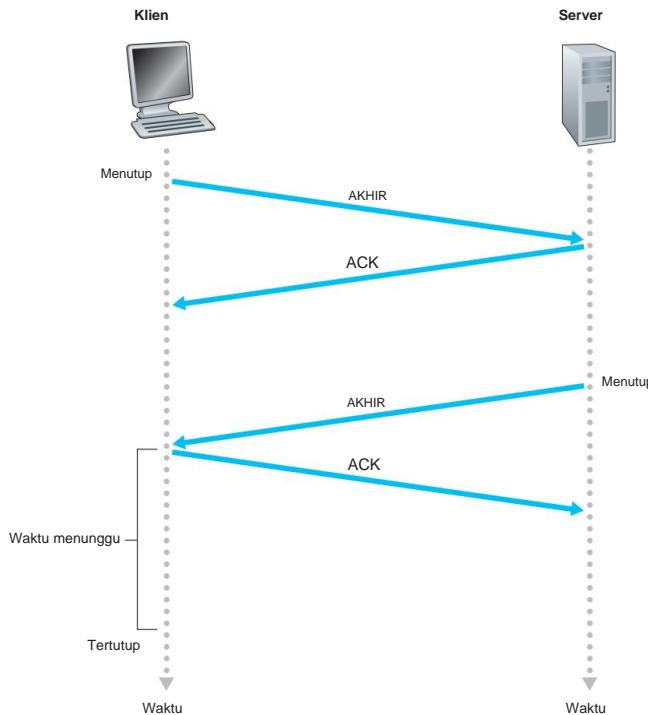


Gambar 3.39 Jabat tangan tiga arah TCP: pertukaran segmen

memerintah. Ini menyebabkan klien TCP mengirim segmen TCP khusus ke proses server. Segmen khusus ini memiliki bit bendera di header segmen, bit FIN (lihat Gambar 3.29), disetel ke 1. Saat server menerima segmen ini, ia mengirimkan segmen pengakuan kepada klien sebagai balasannya. Server kemudian mengirimkan segmen penonaktifannya sendiri, yang bit FIN-nya disetel ke 1. Akhirnya, klien mengakui segmen penonaktifan server. Pada titik ini, semua sumber daya di kedua host sekarang dibatalkan alokasinya.

Selama masa koneksi TCP, protokol TCP yang berjalan di setiap host melakukan transisi melalui berbagai **status TCP**. Gambar 3.41 mengilustrasikan urutan tipikal status TCP yang dikunjungi oleh *klien* TCP. TCP klien dimulai dalam keadaan TERTUTUP. Aplikasi di sisi klien memulai koneksi TCP baru (dengan membuat objek Socket dalam contoh Java kita seperti dalam contoh Python dari Bab 2). Ini menyebabkan TCP di klien mengirim segmen SYN ke TCP di server. Setelah mengirim segmen SYN, klien TCP memasuki status SYN_SENT. Saat dalam status SYN_SENT, TCP klien menunggu segmen dari server TCP yang mencakup pengakuan untuk segmen klien sebelumnya dan mengatur bit SYN ke 1. Setelah menerima segmen seperti itu, TCP klien memasuki status ESTABLISHED. Saat berada dalam status ESTABLISHED, klien TCP dapat mengirim dan menerima segmen TCP yang berisi data muatan (yaitu, data yang dihasilkan aplikasi).

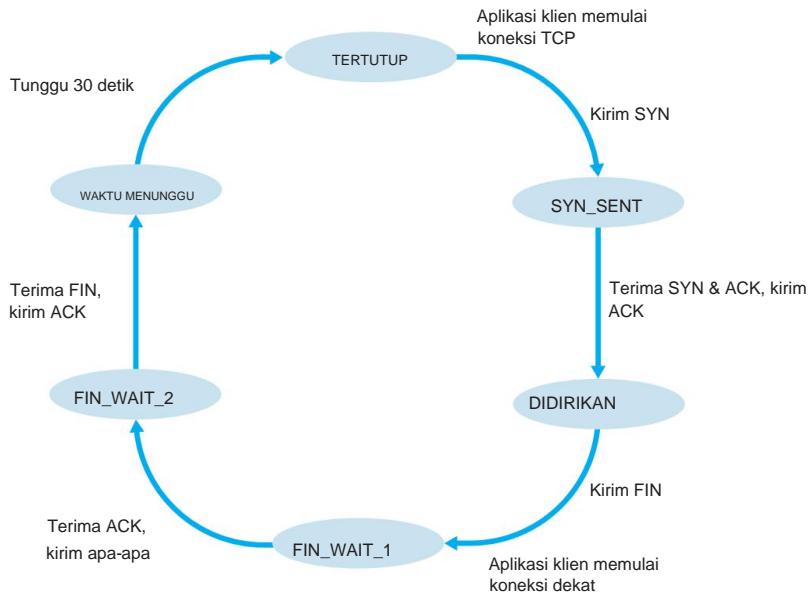
3.5 • TRANSPORTASI BERORIENTASI KONEKSI: TCP 255

**Gambar 3.40** Menutup koneksi TCP

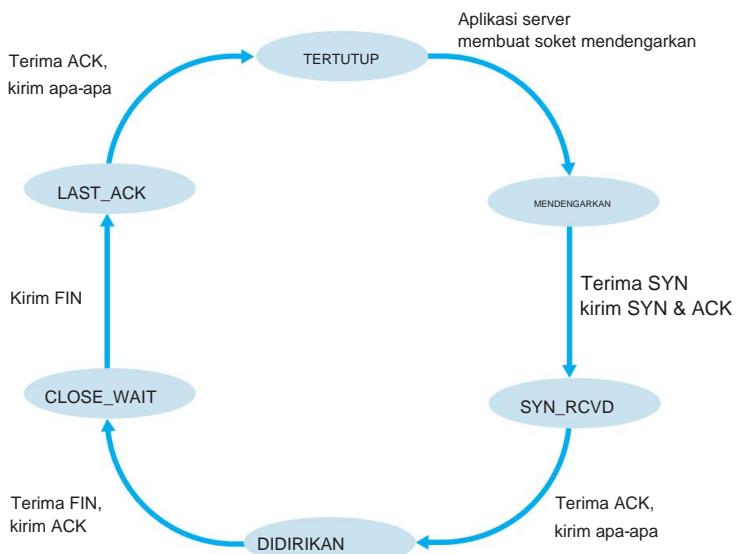
Misalkan aplikasi klien memutuskan ingin menutup koneksi. (Perhatikan bahwa server juga dapat memilih untuk menutup koneksi.) Hal ini menyebabkan klien TCP mengirim segmen TCP dengan bit FIN diatur ke 1 dan memasuki status FIN_WAIT_1. Saat berada dalam status FIN_WAIT_1, klien TCP menunggu segmen TCP dari server dengan pemberitahuan. Saat menerima segmen ini, TCP klien memasuki status FIN_WAIT_2. Saat dalam status FIN_WAIT_2, klien menunggu segmen lain dari server dengan bit FIN disetel ke 1; setelah menerima segmen ini, TCP klien mengakui segmen server dan memasuki status TIME_WAIT. Status TIME_WAIT memungkinkan klien TCP mengirim ulang pengakuan akhir jika ACK hilang. Waktu yang dihabiskan dalam status TIME_WAIT bergantung pada implementasi, tetapi nilai umumnya adalah 30 detik, 1 menit, dan 2 menit. Setelah menunggu, koneksi ditutup secara resmi dan semua sumber daya di sisi klien (termasuk nomor port) dilepaskan.

Gambar 3.42 mengilustrasikan rangkaian status yang biasanya dikunjungi oleh TCP sisi server, dengan asumsi klien memulai pembongkaran koneksi. Transisi cukup jelas. Dalam dua diagram transisi negara ini, kami hanya menunjukkan bagaimana koneksi TCP biasanya dibuat dan dimatikan. Kami belum menjelaskan apa

256 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.41 Urutan khas dari status TCP yang dikunjungi oleh klien TCP



Gambar 3.42 Urutan tipikal dari status TCP yang dikunjungi oleh TCP sisi server



FOKUS PADA KEAMANAN

SERANGAN SYN BANJIR

Kami telah melihat dalam diskusi kami tentang jabat tangan tiga arah TCP bahwa server mengalokasikan dan menginisialisasi variabel koneksi dan buffer sebagai respons terhadap SYN yang diterima. Server kemudian mengirimkan SYNACK sebagai tanggapan, dan menunggu segmen ACK dari klien. Jika klien tidak mengirim ACK untuk menyelesaikan langkah ketiga dari jabat tangan 3 arah ini, pada akhirnya (seringkali setelah satu menit atau lebih) server akan mengakhiri koneksi setengah terbuka dan merebut kembali sumber daya yang dialokasikan.

Protokol manajemen koneksi TCP ini menetapkan panggung untuk serangan Denial of Service (DoS) klasik yang dikenal sebagai **serangan banjir SYN**. Dalam serangan ini, penyerang mengirimkan sejumlah besar segmen TCP SYN, tanpa menyelesaikan langkah jabat tangan ketiga. Dengan banyaknya segmen SYN ini, sumber daya koneksi server menjadi habis karena dialokasikan (tetapi tidak pernah digunakan!) untuk koneksi setengah terbuka; klien yang sah kemudian ditolak layanannya. Serangan banjir SYN tersebut termasuk di antara serangan DoS pertama yang didokumentasikan [CERT SYN 1996]. Untungnya, pertahanan efektif yang dikenal sebagai **cookie SYN** [RFC 4987] sekarang digunakan di sebagian besar sistem operasi utama.

Cookie SYN berfungsi sebagai berikut:

- o Ketika server menerima segmen SYN, server tidak mengetahui apakah segmen tersebut berasal dari pengguna yang sah atau merupakan bagian dari serangan banjir SYN. Jadi, alih-alih membuat koneksi TCP setengah terbuka untuk SYN ini, server membuat nomor urutan TCP awal yang merupakan fungsi rumit (fungsi hash) dari alamat IP sumber dan tujuan dan nomor port dari segmen SYN, serta nomor rahasia yang hanya diketahui oleh server. Nomor urut awal yang dibuat dengan hati-hati ini disebut "cookie". Server kemudian mengirimkan paket SYNACK kepada klien dengan nomor urut awal khusus ini.

Yang penting, server tidak ingat tidak

Kue kering atau apapun regara lain informasi yang sesuai ke itu SIN .

- o Klien yang sah akan mengembalikan segmen ACK. Ketika server menerima ini ACK, harus memverifikasi bahwa ACK sesuai dengan beberapa SYN yang dikirim sebelumnya. Tetapi bagaimana ini dilakukan jika server tidak menyimpan memori tentang segmen SYN? Seperti yang mungkin sudah Anda duga, ini dilakukan dengan cookie. Ingatlah bahwa untuk ACK yang sah, nilai dalam bidang pengakuan sama dengan nomor urut awal dalam SYNACK (nilai cookie dalam kasus ini) ditambah satu (lihat Gambar 3.39). Server kemudian dapat menjalankan fungsi hash yang sama menggunakan alamat IP sumber dan tujuan serta nomor port di SYNACK (yang sama dengan di SYN asli) dan nomor rahasia. Jika hasil dari fungsi plus satu sama dengan nilai pengakuan (cookie) di SYNACK klien, server menyimpulkan bahwa ACK cor merespons segmen SYN sebelumnya dan karenanya valid. Server kemudian membuat koneksi terbuka penuh bersama dengan soket.

- o Di sisi lain, jika klien tidak mengembalikan segmen ACK, maka SYN asli tidak membahayakan server, karena server belum mengalokasikan sumber daya apa pun sebagai respons terhadap SYN palsu asli.

258 BAB 3 • LAPISAN TRANSPORTASI

terjadi dalam skenario patologis tertentu, misalnya, ketika kedua sisi koneksi ingin memulai atau mematikan pada waktu yang sama. Jika Anda tertarik untuk mempelajari hal ini dan isu lanjutannya mengenai TCP, Anda dianjurkan untuk melihat buku lengkap Stevens [Stevens 1994].

Pembahasan kita di atas mengasumsikan bahwa baik klien maupun server siap untuk berkomunikasi, yaitu, bahwa server mendengarkan pada port tempat klien mengirimkan segmen SYN-nya. Mari pertimbangkan apa yang terjadi ketika host menerima segmen TCP yang nomor port atau alamat IP sumbernya tidak cocok dengan soket yang ada di host. Misalnya, sebuah host menerima paket TCP SYN dengan port tujuan 80, tetapi host tersebut tidak menerima koneksi pada port 80 (yaitu, tidak menjalankan server Web pada port 80). Kemudian tuan rumah akan mengirimkan segmen reset khusus ke sumbernya. Segmen TCP ini memiliki bit flag RST (lihat Bagian 3.5.2) yang disetel ke 1. Jadi, ketika host mengirimkan segmen reset, ia memberi tahu sumber "Saya tidak memiliki soket untuk segmen itu. Mohon jangan mengirim ulang segmen tersebut." Ketika host menerima paket UDP yang nomor port tujuannya tidak cocok dengan soket UDP yang sedang berjalan, host mengirimkan datagram ICMP khusus, seperti yang dibahas di Bab 4.

Sekarang setelah kita memiliki pemahaman yang baik tentang manajemen koneksi TCP, mari kita tinjau kembali alat pemindaian port nmap dan periksa lebih dekat cara kerjanya. Untuk menjelajahi port TCP tertentu, katakanlah port 6789, pada host target, nmap akan mengirimkan segmen TCP SYN dengan port tujuan 6789 ke host tersebut. Ada tiga kemungkinan hasil:

- *Host sumber menerima segmen TCP SYNACK dari host target.* Karena ini berarti aplikasi sedang berjalan dengan port TCP 6789 pada pos target, nmap mengembalikan "terbuka".
- *Host sumber menerima segmen TCP RST dari host target.* Ini berarti bahwa segmen SYN mencapai host target, tetapi host target tidak menjalankan aplikasi dengan port TCP 6789. Tetapi penyerang setidaknya tahu bahwa segmen yang ditujukan ke host di port 6789 tidak diblokir oleh firewall apa pun di jalur antara host sumber dan target. (Firewall dibahas di Bab 8.)
- *Sumber tidak menerima apa pun.* Ini kemungkinan berarti bahwa segmen SYN diblokir

oleh firewall intervensi dan tidak pernah mencapai host target.

Nmap adalah alat yang ampuh, yang dapat "menyatukan sambungan" tidak hanya untuk port TCP terbuka, tetapi juga untuk port UDP terbuka, untuk firewall dan konfigurasinya, dan bahkan untuk versi aplikasi dan sistem operasi. Sebagian besar dilakukan dengan memanipulasi segmen manajemen sambungan TCP [Skoudis 2006]. Anda dapat mengunduh nmap dari www.nmap.org.

Ini melengkapi pengantar kami tentang kontrol kesalahan dan kontrol aliran di TCP. Pada Bagian 3.7 kita akan kembali ke TCP dan melihat kontrol kongesti TCP secara mendalam. Namun, sebelum melakukannya, pertama-tama kami mundur dan memeriksa masalah pengendalian kemacetan dalam konteks yang lebih luas.

3.6 Prinsip Pengendalian Kemacetan

Pada bagian sebelumnya, kami memeriksa prinsip umum dan mekanisme TCP khusus yang digunakan untuk menyediakan layanan transfer data yang andal dalam menghadapi kehilangan paket. Kami sebutkan sebelumnya bahwa, dalam praktiknya, kerugian tersebut biasanya hasil dari melimpahnya buffer router karena jaringan menjadi padat. Pengiriman ulang paket dengan demikian memperlakukan gejala kemacetan jaringan (hilangnya segmen lapisan transport tertentu) tetapi tidak mengobati penyebab kemacetan jaringan — terlalu banyak sumber yang mencoba mengirim data dengan kecepatan yang terlalu tinggi. Untuk mengatasi penyebab kemacetan jaringan, diperlukan mekanisme untuk membatasi pengirim dalam menghadapi kemacetan jaringan.

Pada bagian ini, kami mempertimbangkan masalah kontrol kemacetan dalam konteks umum, berusaha memahami mengapa kemacetan adalah hal yang buruk, bagaimana kemacetan jaringan dimanifestasikan dalam kinerja yang diterima oleh aplikasi lapisan atas, dan berbagai pendekatan yang dapat diambil untuk menghindari, atau bereaksi terhadap, kemacetan jaringan. Studi yang lebih umum tentang kontrol kongesti ini sesuai karena, seperti halnya transfer data yang andal, studi ini menempati urutan teratas dalam daftar "sepuluh besar" masalah fundamental penting kami dalam jaringan. Kami mengakhiri bagian ini dengan diskusi tentang kontrol kongesti dalam layanan **laju bit (ABR)** yang tersedia dalam jaringan **mode transfer asinkron (ATM)**. Bagian berikut berisi studi rinci tentang algoritma kontrol kemacetan TCP.

3.6.1 Penyebab dan Biaya Kemacetan

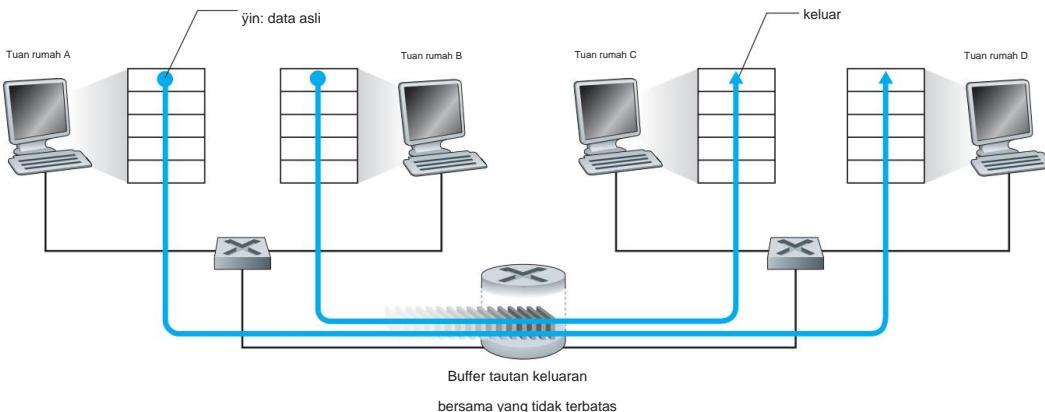
Mari kita mulai studi umum tentang pengendalian kemacetan dengan memeriksa tiga skenario yang semakin kompleks di mana kemacetan terjadi. Dalam setiap kasus, kita akan melihat mengapa kemacetan terjadi sejak awal dan pada biaya kemacetan (dalam hal sumber daya yang tidak digunakan sepenuhnya dan kinerja buruk yang diterima oleh sistem akhir). Kami tidak akan (belum) fokus pada bagaimana bereaksi terhadap, atau menghindari, kemacetan, melainkan fokus pada masalah yang lebih sederhana untuk memahami apa yang terjadi saat host meningkatkan laju transmisinya dan kerja jaringan menjadi padat.

Skenario 1: Dua Pengirim, sebuah Router dengan Buffer Tak Terbatas

Kita mulai dengan mempertimbangkan kemungkinan skenario kemacetan yang paling sederhana: Dua host (A dan B) masing-masing memiliki koneksi yang berbagi satu hop antara sumber dan tujuan, seperti yang ditunjukkan pada Gambar 3.43.

Mari kita asumsikan bahwa aplikasi di Host A mengirim data ke koneksi (misalnya, meneruskan data ke protokol tingkat transportasi melalui soket) dengan kecepatan rata-rata byte/detik. Data ini asli dalam artian ~~sempela~~ data adalah ke soket hanya sekali. Protokol tingkat transportasi yang

260 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.43 Skenario kemacetan 1: Dua koneksi berbagi satu lompatan dengan buffer tak terbatas

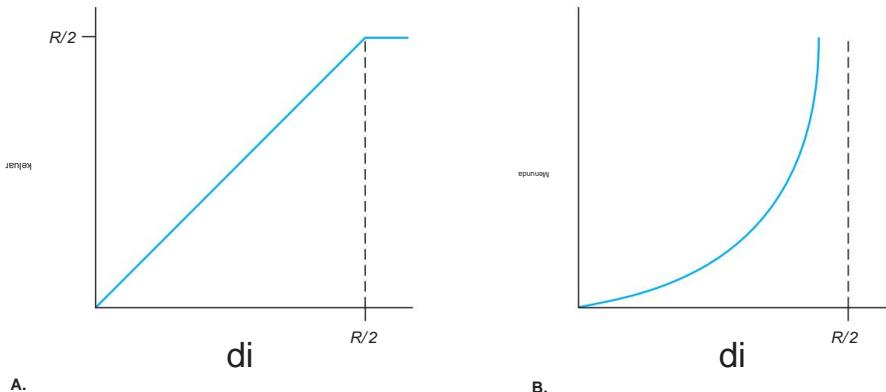
yang sederhana. Data dikemas dan dikirim; tidak ada pemulihkan kesalahan (misalnya, misi retrans), kontrol aliran, atau kontrol kemacetan dilakukan. Mengabaikan overhead tambahan karena menambahkan informasi header transport dan lapisan bawah, kecepatan di mana Host A menawarkan lalu lintas ke router dalam skenario pertama ini adalah byte/detik.

Host B beroperasi dengan cara yang sama, dan kami mengasumsikan untuk kesederhanaan bahwa itu juga dikirim dengan ~~kelebihan kapasitas link~~ ~~Raketr dari host A ke host B kapasitas link~~ ~~Raketr~~ memiliki buffer yang memungkinkannya untuk menyimpan paket yang masuk ketika tingkat kedatangan paket melebihi kapasitas link keluar. Dalam skenario pertama ini, kami berasumsi bahwa router memiliki ruang buffer dalam jumlah tak terbatas.

Gambar 3.44 memplot kinerja koneksi Host A di bawah skenario pertama ini. Grafik kiri memplot **throughput per sambungan** (jumlah byte per detik pada penerima) sebagai fungsi dari laju pengiriman sambungan. Untuk kecepatan pengiriman antara 0 dan R/2, throughput pada penerima sama dengan kecepatan pengiriman pengirim—semua yang dikirim oleh pengirim diterima di penerima dengan penundaan yang terbatas.

Namun, ketika kecepatan pengiriman di atas R/2, throughputnya hanya R/2. Batas atas throughput ini merupakan konsekuensi dari pembagian kapasitas link antara dua koneksi. Tautan tersebut tidak dapat mengirimkan paket ke penerima dengan kecepatan tetap yang melebihi R/2. Tidak peduli seberapa tinggi Host A dan B menetapkan kecepatan pengirimannya, mereka masing-masing tidak akan pernah melihat throughput yang lebih tinggi dari R/2.

Mencapai throughput per-koneksi dari R/2 mungkin benar-benar tampak sebagai hal yang baik, karena link tersebut digunakan sepenuhnya dalam mengirimkan paket ke tujuan mereka. Grafik sebelah kanan pada Gambar 3.44, bagaimanapun, menunjukkan konsekuensi pengoperasian di dekat kapasitas link. Saat kecepatan pengiriman mendekati R/2 (dari kiri), penundaan rata-rata menjadi semakin besar. Ketika kecepatan pengiriman melebihi R/2,



Gambar 3.44 Skenario kemacetan 1: Throughput dan delay sebagai fungsi dari laju pengiriman host

jumlah rata-rata paket yang antri di router tidak terbatas, dan rata-rata penundaan antara sumber dan tujuan menjadi tidak terbatas (dengan asumsi bahwa koneksi beroperasi pada kecepatan pengiriman ini untuk periode waktu yang tidak terbatas dan tersedia buffer dalam jumlah yang tidak terbatas). Jadi, saat beroperasi pada throughput agregat mendekati R mungkin ideal dari sudut pandang throughput, ini jauh dari ideal dari sudut tunda. *Bahkan dalam skenario yang (sangat) ideal ini, kami telah menemukan satu biaya dari jaringan yang padat—penundaan antrian yang besar dialami karena tingkat kedatangan paket mendekati kapasitas tautan.*

Skenario 2: Dua Pengirim dan Router dengan Buffer Hingga

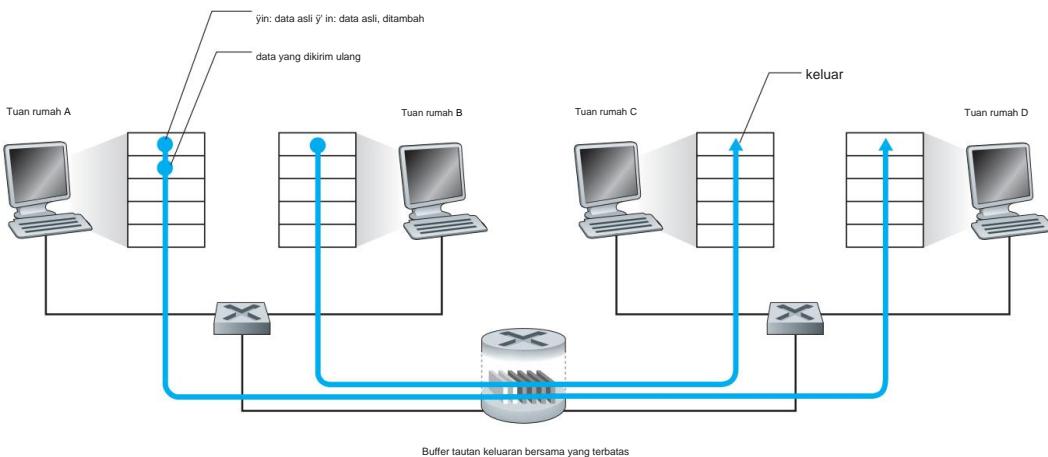
Mari kita sedikit memodifikasi skenario 1 dengan dua cara berikut (lihat Gambar 3.45).

Pertama, jumlah buffering router diasumsikan terbatas. Konsekuensi dari asumsi dunia nyata ini adalah bahwa paket akan dibuang saat tiba di buffer yang sudah penuh. Kedua, kami berasumsi bahwa setiap koneksi dapat diandalkan. Jika paket yang berisi segmen tingkat transportasi dijatuhkan di router, pengirim pada akhirnya akan mentransmisikannya kembali. Karena paket dapat ditransmisikan ulang, kita sekarang harus lebih berhati-hati dengan penggunaan istilah *kecepatan pengiriman*. Secara khusus, mari kita kembali menunjukkan tingkat di mana aplikasi mengirimkan data asli ke dalam soket dengan byte/detik. Tingkat di mana lapisan transportasi (transmisi data yang ditransmisikan ulang) ke dalam jaringan akan dilambangkan kadang-kadang disebut sebagai **beban yang ditawarkan** ke jaringan.

... byte/detik.

Performa yang direalisasikan dalam skenario 2 sekarang akan sangat bergantung pada bagaimana transmisi ulang dilakukan. Pertama, pertimbangkan kasus yang tidak realistik bahwa Host A entah bagaimana (secara ajaib!) dapat menentukan apakah buffer bebas atau tidak di router dan dengan demikian mengirimkan paket hanya ketika buffer bebas. Dalam hal ini, tidak akan terjadi kerugian,

262 BAB 3 • LAPISAN TRANSPORTASI

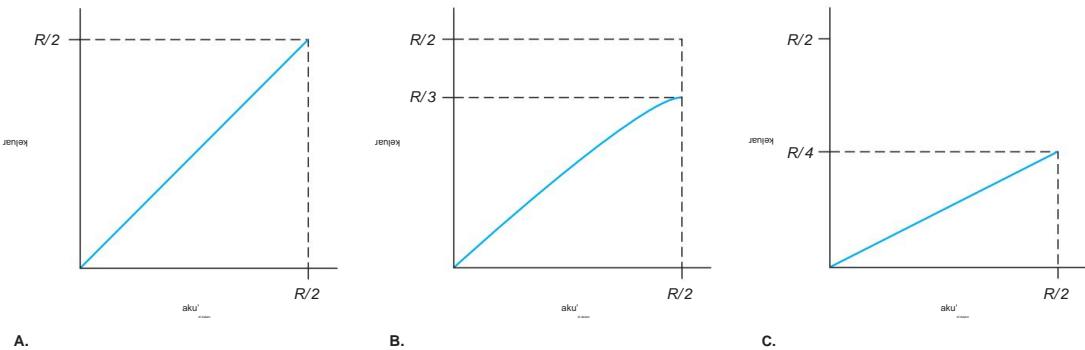


Gambar 3.45 Skenario 2: Dua host (dengan transmisi ulang) dan sebuah router dengan buffer terbatas

akan sama dengan dan throughput koneksi akan sama dengan
 Kasus ini ditunjukkan pada Gambar 3.46(a). Dari sudut pandang throughput, performa ideal—segala sesuatu yang dikirim akan diterima. Perhatikan bahwa kecepatan pengiriman host rata-rata tidak dapat melebihi $R/2$ dalam skenario ini, karena kehilangan paket diasumsikan tidak pernah terjadi.

Pertimbangkan selanjutnya kasus yang sedikit lebih realistik bahwa pengirim mengirim ulang hanya ketika sebuah paket diketahui pasti akan hilang. (Sekali lagi, asumsi ini agak berlebihan. Namun, host pengirim mungkin menyetel waktu tunggunya cukup besar untuk memastikan bahwa paket yang belum diakui telah hilang.) Dalam hal ini, kinerja mungkin terlihat seperti yang ditunjukkan pada Gambar 3.46(b). Untuk menghargai apa yang terjadi di sini, pertimbangkan kasus bahwa beban yang ditawarkan, (laju transmisi data asli ditambah transmisi ulang), sama dengan $R/2$. Menurut Gambar 3.46(b), pada nilai beban yang ditawarkan ini, ~~kelebihan pengiriman data kenyataan prima adalah R/3 atau sekitar 0,5 R bit/s~~ ~~data yang dikirim ulang~~ detik (rata-rata) adalah data yang ditransmisikan ulang. *Di sini kita melihat biaya lain dari jaringan yang padat — pengirim harus melakukan transmisi ulang untuk mengkompensasi paket yang hilang (hilang) karena buffer overflow.*

Akhirnya, mari kita pertimbangkan kasus di mana pengirim mungkin kehabisan waktu sebelum waktunya dan mentransmisikan ulang paket yang telah tertunda dalam antrian tetapi belum hilang. Dalam hal ini, paket data asli dan transmisi ulang dapat mencapai penerima. Dari



Gambar 3.46 Performa Skenario 2 dengan buffer terbatas

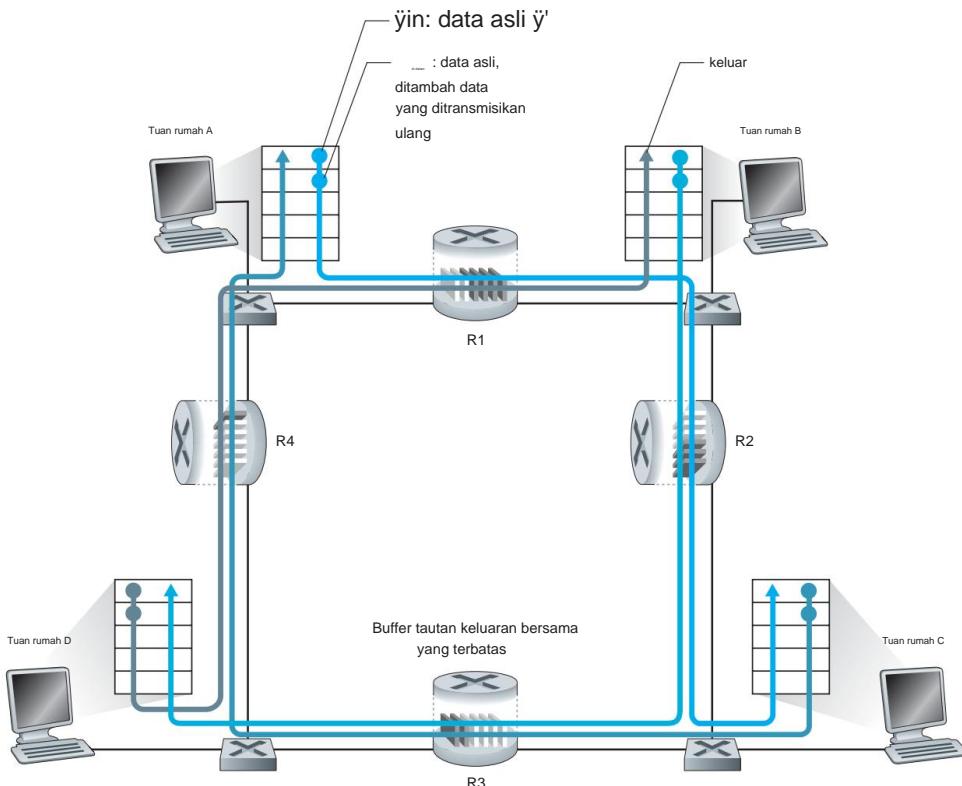
tentu saja, penerima hanya membutuhkan satu salinan dari paket ini dan akan membuang misi retrans. Dalam hal ini, pekerjaan yang dilakukan oleh router dalam meneruskan salinan yang ditransmisikan ulang dari paket asli menjadi sia-sia, karena penerima telah menerima salinan asli dari paket ini. Router sebaiknya menggunakan kapasitas link trans mission untuk mengirim paket yang berbeda. *Di sini kemudian ada biaya lain dari jaringan yang padat — transmisi ulang yang tidak dibutuhkan oleh pengirim dalam menghadapi penundaan yang besar dapat menyebabkan router menggunakan bandwidth tautannya untuk meneruskan salinan paket yang tidak dibutuhkan.* Gambar 3.46 (c) menunjukkan throughput versus beban yang ditawarkan ketika setiap paket diasumsikan diteruskan (rata-rata) dua kali oleh router. Karena setiap paket diteruskan dua kali, throughput akan memiliki nilai asimtotik $R/4$ saat muatan yang ditawarkan mendekati $R/2$.

Skenario 3: Empat Pengirim, Router dengan Buffer Hingga, dan Jalur Multihop

Dalam skenario kemacetan terakhir kami, empat host mengirimkan paket, masing-masing melewati jalur ping dua-hop yang tumpang tindih, seperti yang ditunjukkan pada Gambar 3.47. Kami sekali lagi berasumsi bahwa setiap host menggunakan mekanisme timeout/retransmisi untuk mengimplementasikan layanan transfer data yang andal, bahwa semua host memiliki nilai yang sama dan semua ~~dan~~ router memiliki kapasitas R byte/

Mari pertimbangkan koneksi dari Host A ke Host C, melewati router R1 dan R2. Koneksi A-C berbagi router R1 dengan koneksi D-B dan berbagi router R2 dengan koneksi B-D. Untuk nilai buffer overflows yang sangat kecil jarang terjadi (seperti dalam skenario kemacetan 1 dan 2), dan throughput kira-kira sama ~~dengan~~ yang ditawarkan. Untuk nilai throughput yang sedikit lebih besar juga lebih besar, karena lebih banyak data asli yang ditransmisikan ke dalam ~~dan~~ yang sesuai

264 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.47 Empat pengirim, router dengan buffer terbatas, dan jalur multihop

jaringan dan dikirim ke tujuan, dan luapan masih jarang terjadi. Jadi, untuk nilai kecil dari peningkatan in , Setelah mempertimbangkan kapasitas tautan keluar bersama yang terbatas pada R2, lalu lintas ke C yang bisa

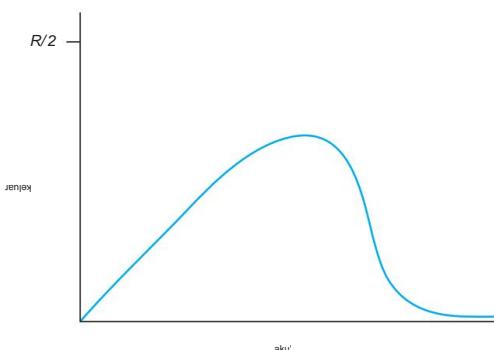
kedatangan di R2 yang paling banyak R , kapasitas tautan dari R1 ke R2, terlepas dari nilai sangat besar untuk semua koneksi (termasuk koneksi B-D), maka tingkat kedatangan lalu lintas B-D di R2 bisa jauh lebih besar daripada lalu lintas A-C. Karena lalu lintas A-C dan B-D harus bersaing di router R2 untuk jumlah ruang buffer yang terbatas, jumlah trafik A-C yang berhasil melewati R2 (yaitu, tidak hilang karena buffer over flow) menjadi lebih kecil dan lebih kecil karena beban yang ditawarkan dari B-D semakin besar dan sebaliknya. Dalam bentuk teknologi buffer kosong di R2

segera diisi oleh paket B-D, dan throughput koneksi A-C di R2 menjadi nol. Ini, pada gilirannya, menyiratkan bahwa throughput end-to-end A-C menjadi nol dalam batas lalu lintas padat. Pertimbangan ini memunculkan beban yang ditawarkan versus tradeoff throughput yang ditunjukkan pada Gambar 3.48.

Alasan untuk penurunan throughput dengan meningkatnya beban yang ditawarkan terbukti ketika seseorang mempertimbangkan jumlah pekerjaan yang terbuang yang dilakukan oleh pekerjaan jaringan. Dalam skenario lalu lintas tinggi yang diuraikan di atas, setiap kali sebuah paket dijatuhkan di router hop kedua, pekerjaan yang dilakukan oleh router hop pertama dalam meneruskan paket ke router hop kedua akhirnya "terbuang sia-sia". Jaringan akan sama baiknya (lebih tepatnya, sama buruknya) jika router pertama hanya membuang paket itu dan tetap diam. Lebih penting lagi, kapasitas transmisi yang digunakan pada router pertama untuk meneruskan paket ke router kedua bisa jadi jauh lebih menguntungkan jika digunakan untuk mengirimkan paket yang berbeda. (Misalnya, ketika memilih paket untuk transmisi, mungkin lebih baik bagi router untuk memberikan prioritas pada paket yang telah melintasi sejumlah router upstream.) Jadi di sini kita melihat biaya lain untuk menjatuhkan paket karena kemacetan—ketika sebuah paket dijatuhkan di sepanjang jalur, kapasitas transmisi yang digunakan pada masing-masing tautan hulu untuk meneruskan paket itu ke titik di mana paket itu dijatuhkan akhirnya terbuang sia-sia.

3.6.2 Pendekatan Pengendalian Kemacetan

Pada Bagian 3.7, kita akan mempelajari pendekatan spesifik TCP untuk kontrol kongesti dengan sangat rinci. Di sini, kami mengidentifikasi dua pendekatan luas untuk kontrol kemacetan yang diambil dalam praktik dan membahas arsitektur jaringan spesifik dan protokol kontrol kemacetan yang mewujudkan pendekatan ini.



Gambar 3.48 Kinerja Skenario 3 dengan buffer terbatas dan jalur multihop

266 BAB 3 • LAPISAN TRANSPORTASI

Pada tingkat yang paling luas, kita dapat membedakan antara pendekatan kontrol kemacetan dengan apakah lapisan jaringan memberikan bantuan eksplisit ke lapisan transport untuk tujuan kontrol kemacetan:

- *Kontrol kemacetan end-to-end.* Dalam pendekatan end-to-end untuk kontrol kemacetan, lapisan jaringan *tidak memberikan dukungan eksplisit* ke lapisan transport untuk tujuan kontrol kemacetan. Bahkan keberadaan kemacetan di jaringan harus disimpulkan oleh sistem akhir hanya berdasarkan perilaku jaringan yang diamati (misalnya, kehilangan paket dan penundaan). Kita akan melihat di Bagian 3.7 bahwa TCP harus mengambil pendekatan ujung ke ujung ini menuju kontrol kemacetan, karena lapisan IP tidak memberikan umpan balik ke sistem akhir mengenai kemacetan jaringan. Kehilangan segmen TCP (seperti yang ditunjukkan oleh batas waktu atau pengakuan rangkap tiga) diambil sebagai indikasi kemacetan kerja jaringan dan TCP mengurangi ukuran jendelanya. Kita juga akan melihat proposal yang lebih baru untuk kontrol kongesti TCP yang menggunakan nilai tundaan bolak-balik yang meningkat sebagai indikator peningkatan kongesti jaringan.
- *Kontrol kemacetan berbantuan jaringan.* Dengan kontrol kemacetan berbantuan jaringan, komponen lapisan jaringan (yaitu, router) memberikan umpan balik eksplisit kepada pengirim mengenai keadaan kemacetan di jaringan. Umpan balik ini mungkin sesederhana satu bit yang menunjukkan kemacetan di sebuah tautan. Pendekatan ini diambil pada awal IBM SNA [Schwartz 1982] dan DEC DECnet [Jain 1989; Ramakrishnan 1990], baru-baru ini diusulkan untuk jaringan TCP/IP [Floyd TCP 1994; RFC 3168], dan juga digunakan dalam kontrol kemacetan ATM available bit-rate (ABR), seperti dibahas di bawah ini. Umpan balik jaringan yang lebih canggih juga dimungkinkan. Misalnya, salah satu bentuk kontrol kemacetan ATM ABR yang akan kita pelajari sebentar lagi memungkinkan router untuk memberi tahu pengirim secara eksplisit tentang laju transmisi yang dapat didukungnya (router) pada tautan keluar. Protokol XCP [Katabi 2002] menyediakan umpan balik yang dihitung router untuk setiap sumber, yang dibawa dalam header paket, mengenai bagaimana sumber itu harus menambah atau mengurangi laju transmisinya.

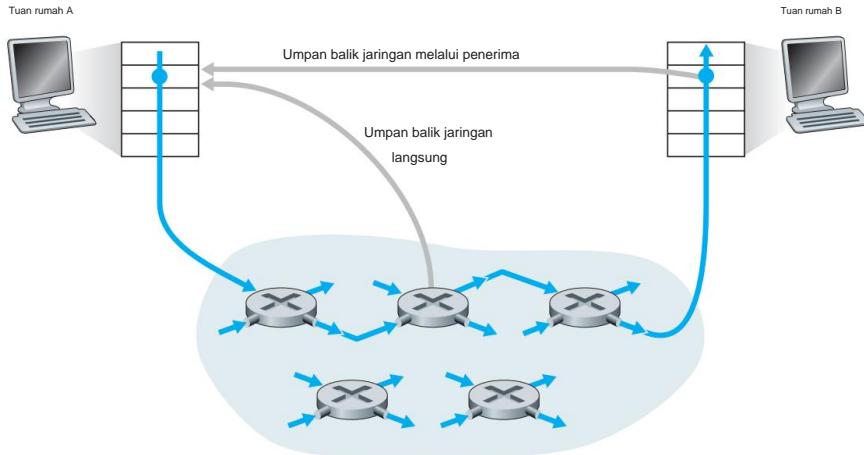
Untuk kontrol kongesti berbantuan jaringan, informasi kongesti biasanya diumpan balik dari jaringan ke pengirim dengan salah satu dari dua cara, seperti yang ditunjukkan pada Gambar 3.49. Umpan balik langsung dapat dikirim dari router jaringan ke pengirim. Bentuk pemberitahuan ini biasanya berbentuk **paket tersedak** (pada dasarnya mengatakan, "Saya sesak!"). Bentuk notifikasi kedua terjadi ketika router menandai/memperbarui bidang dalam paket yang mengalir dari pengirim ke penerima untuk menunjukkan kemacetan. Setelah menerima paket yang ditandai, penerima kemudian memberi tahu pengirim tentang indikasi kemacetan.

Perhatikan bahwa bentuk pemberitahuan yang terakhir ini membutuhkan setidaknya waktu perjalanan bolak-balik penuh.

3.6.3 Kontrol Kemacetan Berbantuan Jaringan Contoh: Kontrol Kemacetan ATM ABR

Kami menyimpulkan bagian ini dengan studi kasus singkat tentang algoritme kontrol kongesti di ATM ABR—protokol yang mengambil pendekatan berbantuan jaringan menuju kontrol kongesti. Kami menekankan bahwa tujuan kami di sini bukan untuk menjelaskan aspek arsitektur ATM

3.6 • PRINSIP PENGENDALIAN KEMAJUAN 267



Gambar 3.49 Dua jalur umpan balik untuk informasi kemacetan yang terindikasi jaringan

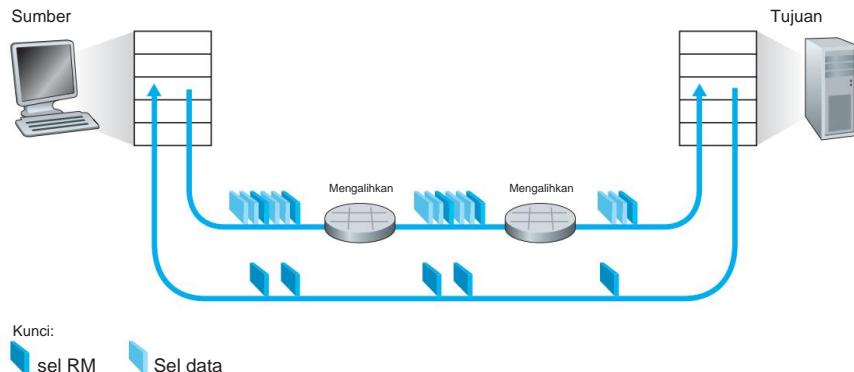
dengan sangat rinci, melainkan untuk mengilustrasikan protokol yang mengambil pendekatan yang sangat berbeda terhadap kontrol kemacetan dari protokol TCP Internet. Memang, di bawah ini kami hanya menyajikan beberapa aspek arsitektur ATM yang diperlukan untuk memahami kontrol kemacetan ABR.

Pada dasarnya ATM mengambil pendekatan berorientasi virtual-circuit (VC) terhadap pengalihan paket. Ingat dari diskusi kita di Bab 1, ini berarti bahwa setiap sakelar pada jalur sumber-ke-tujuan akan mempertahankan status tentang VC sumber-ke-tujuan. Keadaan per-VC ini memungkinkan sebuah saklar untuk melacak perilaku pengirim individu (misalnya, melacak tingkat transmisi rata-rata mereka) dan untuk mengambil tindakan kontrol kongesti sumber-spesifik (seperti secara eksplisit memberi sinyal kepada pengirim untuk mengurangi kecepatannya ketika saklar menjadi sesak). Status per-VC pada switch jaringan ini membuat ATM cocok untuk melakukan kontrol kemacetan berbantuan jaringan.

ABR telah dirancang sebagai layanan transfer data yang elastis dengan cara yang mengingatkan pada TCP. Saat jaringan kekurangan beban, layanan ABR harus dapat memanfaatkan bandwidth cadangan yang tersedia; ketika jaringan padat, layanan ABR harus membatasi laju transmisinya ke laju transmisi minimum yang telah ditentukan sebelumnya. Tutorial rinci tentang kontrol kemacetan ATM ABR dan manajemen lalu lintas tersedia di [Jain 1996].

Gambar 3.50 memperlihatkan kerangka pengendalian kongesti ATM ABR. Dalam diskusi kami, kami mengadopsi terminologi ATM (misalnya, menggunakan istilah *switch* daripada *router*, dan istilah *sel* daripada *paket*). Dengan layanan ATM ABR, sel data ditransmisikan dari sumber ke tujuan melalui serangkaian sakelar perantara. Diselingi dengan sel data adalah **sel manajemen sumber daya**

268 BAB 3 • LAPISAN TRANSPORTASI



Gambar 3.50 Kerangka kerja pengendalian kemacetan untuk layanan ATM ABR

(**sel RM**); sel-sel RM ini dapat digunakan untuk menyampaikan informasi terkait kemacetan di antara host dan sakelar. Ketika sebuah sel RM tiba di tujuan, itu akan diputar balik dan dikirim kembali ke pengirim (kemungkinan setelah tujuan mengubah isi sel RM). Switch juga dapat menghasilkan sel RM sendiri dan mengirimkan sel RM ini langsung ke sumber. Sel RM dengan demikian dapat digunakan untuk memberikan umpan balik jaringan langsung dan umpan balik jaringan melalui penerima, seperti yang ditunjukkan pada Gambar 3.50.

Kontrol kemacetan ATM ABR adalah pendekatan berbasis tarif. Artinya, pengirim secara eksplisit menghitung tingkat maksimum yang dapat dikirim dan mengatur dirinya sendiri sesuai dengan itu. ABR menyediakan tiga mekanisme untuk memberi sinyal informasi terkait kemacetan dari sakelar ke penerima:

- **bit EFCI.** Setiap **sel data** berisi bit **indikasi kemacetan maju eksplisit (EFCI)**. Sakelar jaringan yang padat dapat mengatur bit EFCI dalam sel data ke 1 untuk memberi sinyal kemacetan ke host tujuan. Tujuan harus memeriksa bit EFCI di semua sel data yang diterima. Ketika sel RM tiba di tujuan, jika sel data yang paling baru diterima memiliki bit EFCI yang disetel ke 1, maka negara tujuan menyetel bit indikasi kongesti (bit CI) dari sel RM ke 1 dan mengirim sel RM kembali kepada pengirim. Dengan menggunakan EFCI dalam sel data dan bit CI dalam sel RM, pengirim dapat diberi tahu tentang kemacetan di sakelar jaringan.
- **CI dan NI bit.** Seperti disebutkan di atas, sel RM pengirim ke penerima diselingi dengan sel data. Laju interspersi sel RM adalah parameter yang dapat disetel, dengan nilai default menjadi satu sel RM setiap 32 sel data. Sel-sel RM ini memiliki **bit indikasi kemacetan (CI)** dan **bit tanpa peningkatan (NI)** yang dapat diatur oleh a

beralih jaringan padat. Secara khusus, sebuah saklar dapat mengatur bit NI dalam sel RM yang lewat ke 1 di bawah kemacetan ringan dan dapat mengatur bit CI ke 1 di bawah kondisi kemacetan yang parah. Ketika host tujuan menerima sel RM, ia akan mengirim sel RM kembali ke pengirim dengan bit CI dan NI-nya utuh (kecuali bahwa CI dapat disetel ke 1 oleh tujuan sebagai akibat dari mekanisme EFCI yang dijelaskan di atas).

- *pengaturan UGD.* Setiap sel RM juga berisi **bidang laju eksplisit (ER) 2-byte**. Sakelar yang macet dapat menurunkan nilai yang terkandung dalam bidang ER dalam sel RM yang lewat. Dengan cara ini, bidang ER akan diatur ke tingkat minimum yang dapat didukung dari semua sakelar di jalur sumber ke tujuan.

Sumber ATM ABR menyesuaikan kecepatan pengiriman sel sebagai fungsi dari nilai CI, NI, dan ER dalam sel RM yang dikembalikan. Aturan untuk membuat penyesuaian tarif ini agak rumit dan sedikit membosankan. Pembaca yang tertarik dirujuk ke [Jain 1996] untuk detailnya.

3.7 Kontrol Kemacetan TCP

Pada bagian ini kita kembali ke studi kita tentang TCP. Seperti yang kita pelajari di Bagian 3.5, TCP menyediakan layanan transportasi yang handal antara dua proses yang berjalan pada host yang berbeda. Komponen kunci lain dari TCP adalah mekanisme kontrol kemacetannya. Seperti yang ditunjukkan pada bagian sebelumnya, TCP harus menggunakan kontrol kongesti end-to-end daripada kontrol kongesti yang dibantu oleh kerja bersih, karena lapisan IP tidak memberikan umpan balik eksplisit ke sistem akhir mengenai kongesti jaringan.

Pendekatan yang diambil oleh TCP adalah membuat setiap pengirim membatasi kecepatan pengiriman lalu lintas ke koneksinya sebagai fungsi dari kemacetan jaringan yang dirasakan. Jika pengirim TCP merasakan bahwa ada sedikit kemacetan di jalur antara dirinya dan tujuan, maka pengirim TCP meningkatkan kecepatan pengirimannya; jika pengirim merasakan bahwa ada kemacetan di sepanjang jalur, maka pengirim mengurangi kecepatan pengirimannya. Tetapi pendekatan ini menimbulkan tiga pertanyaan. Pertama, bagaimana pengirim TCP membatasi kecepatan pengiriman lalu lintas ke koneksinya? Kedua, bagaimana pengirim TCP merasakan bahwa ada kemacetan di jalur antara dirinya dan tujuan? Dan ketiga, algoritma apa yang harus digunakan pengirim untuk mengubah kecepatan pengirimannya sebagai fungsi dari kemacetan end-to-end yang dirasakan?

Pertama mari kita periksa bagaimana pengirim TCP membatasi kecepatan pengiriman lalu lintas ke koneksinya. Pada Bagian 3.5 kita melihat bahwa setiap sisi dari koneksi TCP terdiri dari buffer penerima, buffer pengiriman, dan beberapa variabel (LastByteRead, rwnd, dan sebagainya). Mekanisme kontrol kongesti TCP yang beroperasi pada pengirim melacak variabel tambahan, **jendela kongesti**. Jendela kemacetan, dilambangkan dengan cwnd, memberlakukan batasan pada kecepatan di mana pengirim TCP dapat mengirimkan lalu lintas

270 BAB 3 • LAPISAN TRANSPORTASI

ke dalam jaringan. Secara khusus, jumlah data yang tidak diakui pada pengirim tidak boleh melebihi minimum cwnd dan rwnd, yaitu:

$$\text{LastByteSent} - \text{LastByteAcked} \min\{\text{cwnd}, \text{rwnd}\}$$

Untuk fokus pada kontrol kongesti (berlawanan dengan kontrol aliran), mari kita asumsikan bahwa buffer penerima TCP sangat besar sehingga batasan jendela terima dapat diabaikan; dengan demikian, jumlah data yang tidak diakui pada pengirim hanya dibatasi oleh cwnd. Kami juga akan berasumsi bahwa pengirim selalu memiliki data untuk dikirim, yaitu semua segmen di jendela kongesti dikirim.

Kendala di atas membatasi jumlah data yang tidak diakui pada pengirim dan karenanya secara tidak langsung membatasi kecepatan pengiriman pengirim. Untuk melihat ini, pertimbangkan koneksi yang kehilangan dan penundaan pengiriman paket dapat diabaikan. Kemudian, kira-kira, pada awal setiap RTT, batasan tersebut mengizinkan pengirim untuk mengirim cwnd byte data ke dalam koneksi; di akhir RTT, pengirim menerima pemberitahuan untuk data tersebut. *Dengan demikian kecepatan pengiriman pengirim kira-kira cwnd/RTT byte/detik. Dengan menyesuaikan nilai cwnd, pengirim dapat menyesuaikan kecepatan pengiriman data ke dalam koneksinya.*

Selanjutnya, mari pertimbangkan bagaimana pengirim TCP merasakan bahwa ada kemacetan di jalur antara dirinya dan tujuan. Mari kita definisikan "kehilangan peristiwa" pada pengirim TCP sebagai terjadinya batas waktu atau penerimaan tiga ACK duplikat dari penerima. (Ingat diskusi kita di Bagian 3.5.4 dari kejadian timeout pada Gambar 3.33 dan modifikasi selanjutnya untuk menyertakan pengiriman ulang cepat saat menerima tiga ACK duplikat.) Ketika ada kemacetan yang berlebihan, maka satu (atau lebih) buffer router di sepanjang jalur meluap, menyebabkan datagram (berisi segmen TCP) dihapus. Datagram yang dijatuhkan, pada gilirannya, menghasilkan peristiwa kerugian pada pengirim — baik batas waktu atau penerimaan tiga ACK duplikat — yang dianggap oleh pengirim sebagai indikasi kemacetan di jalur pengirim ke penerima.

Setelah mempertimbangkan bagaimana kemacetan terdeteksi, selanjutnya mari kita pertimbangkan kasus yang lebih optimis ketika jaringan bebas kemacetan, yaitu ketika peristiwa kerugian tidak terjadi. Dalam hal ini, pengakuan untuk segmen yang sebelumnya tidak diakui akan diterima di pengirim TCP. Seperti yang akan kita lihat, TCP akan menganggap kedatangan ucapan terima kasih ini sebagai indikasi bahwa semuanya baik-baik saja—bahwa segmen yang ditransmisikan ke dalam jaringan berhasil dikirim ke tujuan — dan akan menggunakan ucapan terima kasih untuk meningkatkan ukuran jendela kongesti (dan karenanya tingkat penularannya). Perhatikan bahwa jika acknowledgment tiba pada tingkat yang relatif lambat (misalnya, jika jalur end-end memiliki delay yang tinggi atau berisi link bandwidth rendah), maka jendela kemacetan akan meningkat pada tingkat yang relatif lambat. Di sisi lain, jika pengakuan sampai pada tingkat yang tinggi, maka jendela kemacetan akan meningkat lebih cepat. Karena TCP menggunakan

pengakuan untuk memicu (atau jam) peningkatan ukuran jendela kemacetan, TCP dikatakan **self-clocking**.

Mengingat *mekanisme* penyesuaian nilai cwnd untuk mengontrol kecepatan pengiriman, pertanyaan kritisnya tetap: *Bagaimana* seharusnya pengirim TCP menentukan kecepatan pengirimannya? Jika pengirim TCP secara kolektif mengirim terlalu cepat, mereka dapat memadatkan jaringan, yang mengarah ke tipe kemacetan yang runtuh seperti yang kita lihat pada Gambar 3.48. Memang, versi TCP yang akan kita pelajari sebentar lagi dikembangkan sebagai tanggapan atas keruntuhan kemacetan Internet yang teramat [Jacobson 1988] di bawah versi TCP sebelumnya. Namun, jika pengirim TCP terlalu berhati-hati dan mengirim terlalu lambat, mereka dapat memanfaatkan bandwidth dalam jaringan; yaitu, pengirim TCP dapat mengirim pada kecepatan yang lebih tinggi tanpa membuat jaringan menjadi padat. Lalu bagaimana pengirim TCP menentukan kecepatan pengiriman mereka sedemikian rupa sehingga mereka tidak memacetkan jaringan tetapi pada saat yang sama memanfaatkan semua bandwidth yang tersedia? Apakah pengirim TCP terkoordinasi secara eksplisit, atau adakah pendekatan terdistribusi di mana pengirim TCP dapat mengatur kecepatan pengirimannya hanya berdasarkan informasi lokal? TCP menjawab pertanyaan-pertanyaan ini menggunakan prinsip panduan berikut:

- *Segmen yang hilang menyiratkan kemacetan, dan karenanya, kecepatan pengirim TCP harus diturunkan saat segmen hilang.* Ingat dari diskusi kita di Bagian 3.5.4, bahwa peristiwa batas waktu atau penerimaan empat pengakuan untuk segmen tertentu (satu ACK asli dan kemudian tiga ACK duplikat) ditafsirkan sebagai indikasi "peristiwa kerugian" implisit dari segmen yang mengikuti segmen ACKed empat kali lipat, memicu transmisi ulang dari segmen yang hilang. Dari sudut pandang kontrol kongesti, pertanyaannya adalah bagaimana pengirim TCP harus mengurangi ukuran jendela kongesti, dan karenanya kecepatan pengirimannya, sebagai tanggapan atas peristiwa kerugian yang disimpulkan ini.
- *Segmen yang diakui menunjukkan bahwa jaringan mengirimkan segmen pengirim ke penerima, dan karenanya, kecepatan pengirim dapat dinaikkan saat ACK tiba untuk segmen yang sebelumnya tidak diakui.* Kedatangan pengakuan tepi diambil sebagai indikasi implisit bahwa semuanya baik-baik saja — segmen berhasil dikirim dari pengirim ke penerima, dan jaringan dengan demikian tidak macet. Dengan demikian, ukuran jendela kemacetan dapat ditingkatkan.
- *Pemeriksaan bandwidth.* Mengingat ACK menunjukkan jalur sumber-ke-tujuan yang bebas kemacetan dan peristiwa kerugian yang menunjukkan jalur padat, strategi TCP untuk menyesuaikan laju transmisinya adalah dengan meningkatkan lajunya sebagai respons terhadap kedatangan ACK hingga peristiwa kerugian terjadi, pada titik mana, transmisi kadarnya menurun. Pengirim TCP dengan demikian meningkatkan laju transmisinya untuk menyelidiki laju di mana permulaan kemacetan dimulai, mundur dari laju itu, dan kemudian mulai menyelidiki lagi untuk melihat apakah laju permulaan kemacetan telah berubah. Perilaku pengirim TCP mungkin analogaus untuk anak yang meminta (dan mendapatkan) lebih banyak barang sampai akhirnya dia akhirnya diberitahu "Tidak!", Mundur sedikit, tapi kemudian mulai membuat permintaan

272 BAB 3 • LAPISAN TRANSPORTASI

lagi tidak lama kemudian. Perhatikan bahwa tidak ada pensinyalan eksplisit dari keadaan kemacetan oleh jaringan—ACK dan peristiwa kerugian berfungsi sebagai sinyal implisit—and bahwa setiap pengirim TCP bertindak berdasarkan informasi lokal secara asinkron dari pengirim TCP lainnya.

Mengingat ikhtisar kontrol kongesti TCP ini, kami sekarang berada dalam posisi untuk mempertimbangkan detail algoritma **kontrol kongesti TCP yang terkenal**, yang pertama kali dijelaskan dalam [Jacobson 1988] dan distandarisasi dalam [RFC 5681]. Algoritme memiliki tiga komponen utama: (1) start lambat, (2) penghindaran kemacetan, dan (3) pemulihan cepat. Mulai lambat dan penghindaran kemacetan adalah komponen wajib dari TCP, berbeda dalam cara mereka meningkatkan ukuran cwnd sebagai respons terhadap ACK yang diterima. Kita akan segera melihat bahwa slow start meningkatkan ukuran cwnd lebih cepat (terlepas dari namanya!) daripada penghindaran kemacetan. Pemulihan cepat disarankan, tetapi tidak wajib, untuk pengirim TCP.

Mulai Lambat

Ketika koneksi TCP dimulai, nilai cwnd biasanya diinisialisasi ke nilai kecil 1 MSS [RFC 3390], menghasilkan kecepatan pengiriman awal kira-kira MSS/RTT. Misalnya, jika MSS = 500 byte dan RTT = 200 msec, kecepatan pengiriman awal yang dihasilkan hanya sekitar 20 kbps. Karena bandwidth yang tersedia untuk pengirim TCP mungkin jauh lebih besar daripada MSS/RTT, pengirim TCP ingin mengetahui jumlah bandwidth yang tersedia dengan cepat. Jadi, dalam keadaan **slow-start**, nilai cwnd dimulai pada 1 MSS dan bertambah 1 MSS setiap kali segmen yang ditransmisikan pertama kali diakui. Pada contoh Gambar 3.51, TCP mengirimkan segmen pertama ke dalam jaringan dan menunggu pengakuan. Ketika tepi pengakuan ini tiba, pengirim TCP meningkatkan jendela kemacetan dengan satu MSS dan mengirimkan dua segmen berukuran maksimum. Segmen-segmen ini kemudian diakui, dengan pengirim meningkatkan jendela kemacetan sebesar 1 MSS untuk setiap segmen yang diakui, memberikan jendela kemacetan sebesar 4 MSS, dan seterusnya.

Proses ini menghasilkan penggandaan kecepatan pengiriman setiap RTT. Dengan demikian, kecepatan pengiriman TCP mulai lambat tetapi tumbuh secara eksponensial selama fase mulai lambat.

Tetapi kapan pertumbuhan eksponensial ini harus berakhir? Mulai lambat memberikan beberapa jawaban untuk pertanyaan ini. Pertama, jika ada peristiwa kerugian (yaitu kemacetan) yang ditunjukkan oleh batas waktu, pengirim TCP menetapkan nilai cwnd ke 1 dan memulai kembali proses mulai lambat. Ini juga menetapkan nilai variabel status kedua, ssthresh (singkatan untuk "ambang mulai lambat") ke $cwnd/2$ —setengah dari nilai jendela kemacetan saat kemacetan terdeteksi. Cara kedua di mana awal yang lambat dapat berakhir secara langsung terkait dengan nilai ssthresh. Karena ssthresh adalah setengah dari nilai cwnd saat kemacetan terakhir kali terdeteksi, mungkin agak gegabah untuk terus menggandakan cwnd saat mencapai atau melampaui nilai ssthresh. Jadi, ketika nilai cwnd sama dengan ssthresh, mulai lambat berakhir dan TCP beralih ke mode penghindaran kemacetan. Seperti yang akan kita lihat, TCP meningkat



PRINSIP DALAM PRAKTEK

TCP SPLITTING: MENGOPTIMALKAN KINERJA LAYANAN CLOUD

Untuk layanan cloud seperti pencarian, email, dan jejaring sosial, diinginkan untuk memberikan tingkat respons yang tinggi, idealnya memberikan ilusi kepada pengguna bahwa layanan berjalan dalam sistem akhir mereka sendiri (termasuk ponsel cerdas mereka). Ini bisa menjadi tantangan besar, karena pengguna seringkali berada jauh dari pusat data yang bertanggung jawab untuk menyajikan konten dinamis yang terkait dengan layanan cloud. Memang, jika sistem akhir jauh dari pusat data, maka RTT akan menjadi besar, berpotensi menyebabkan kinerja waktu respons yang buruk karena mulai lambat TCP.

Sebagai studi kasus, pertimbangkan keterlambatan dalam menerima respons untuk kueri penelusuran. Biasanya, server memerlukan tiga jendela TCP selama mulai lambat untuk mengirimkan respons [Pathak 2010]. Jadi waktu dari saat sistem akhir memulai koneksi TCP hingga saat menerima paket terakhir dari respons kira-kira 4 RTT (satu RTT untuk mengatur koneksi TCP ditambah tiga RTT untuk tiga jendela data) ditambah waktu pemrosesan di pusat data. Penundaan RTT ini dapat menyebabkan penundaan yang nyata dalam menampilkan hasil penelusuran untuk sebagian besar kueri. Selain itu, dapat terjadi kehilangan paket yang signifikan dalam jaringan akses, yang menyebabkan transmisi ulang TCP dan bahkan penundaan yang lebih besar.

Salah satu cara untuk mengurangi masalah ini dan meningkatkan kinerja yang dirasakan pengguna adalah dengan (1) menerapkan server front-end lebih dekat dengan pengguna, dan (2) memanfaatkan **pemisahan TCP** dengan memutus koneksi TCP di server front-end. Dengan pemisahan TCP, klien membuat koneksi TCP ke front-end terdekat, dan front-end mempertahankan koneksi TCP yang persisten ke pusat data dengan jendela kemacetan TCP yang sangat besar [Tariq 2008, Pathak 2010, Chen 2011]. Dengan pendekatan ini, waktu respons kira-kira menjadi 4 waktu pemrosesan RTT⁺ RTTBE , di mana RTTFE adalah waktu bolak-balik antara klien dan server front-end, dan RTTBE adalah waktu bolak-balik antara server front-end dan pusat data (server ujung belakang). Jika server front-end dekat dengan klien, maka waktu respons ini kira-kira menjadi RTT ditambah waktu pemrosesan, karena RTTFE sangat kecil dan RTTBE kira-kira RTT. Singkatnya, pemisahan TCP dapat mengurangi penundaan jaringan secara kasar dari 4 RTT ke RTT, secara signifikan meningkatkan kinerja yang dirasakan pengguna, terutama bagi pengguna yang jauh dari pusat data terdekat. Pemisahan TCP juga membantu mengurangi penundaan pengiriman ulang TCP yang disebabkan oleh hilangnya jaringan akses. Saat ini, Google dan Akamai menggunakan server CDN mereka secara ekstensif di jaringan akses (lihat Bagian 7.2) untuk melakukan pemisahan TCP untuk layanan cloud yang mereka dukung [Chen 2011].

cwnd lebih berhati-hati saat dalam mode penghindaran kemacetan. Cara terakhir di mana start lambat dapat berakhir adalah jika tiga ACK duplikat terdeteksi, dalam hal ini TCP melakukan pengiriman ulang cepat (lihat Bagian 3.5.4) dan memasuki status pemulihan cepat, seperti dibahas di bawah. Perilaku TCP pada awal yang lambat dirangkum dalam FSM