INFRASTRUKTUR INTERNET

Jilid 1

Dr. Agus Wibowo, M.Kom, M.Si, MM

YAYASAN PRIMA AGUS TEKNIK

Dr. Agus Wibowo, M.Kom, M.Si, MM

INFRASTRUKTUR INTERNET

Jilid 1

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

INFRASTRUKTUR INTERNET jilid 1

Penulis :
Dr. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 978-623-8120-51-2 (no.jil.lengkap PDF)
978-623-8120-52-9 (jil.1 PDF)

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan atas terselesaikannya buku yang berjudul **"Infrasruktur Internet – Jilid 1"**. ilmu komputer dan teks berorientasi bisnis yang berat pada teori jaringan dan penggunaan dengan sedikit penekanan pada hal-hal praktis. Mereka mencakup Protokol Kontrol Transmisi/Protokol Internet (TCP/IP), server Internet, dan dasar-dasar telekomunikasi tetapi tidak memberikan panduan tentang cara mengimplementasikan server. Buku ini membagi materi secara menjadi dua kategori: bab konsep dan bab studi kasus. Buku ini menyajikan jaringan dan Internet dari beberapa perspektif: media yang mendasari, protokol, perangkat keras, server, dan penggunaannya. Untuk banyak konsep yang tercakup, kami mengikutinya dengan bab studi kasus yang membahas cara menginstal, mengonfigurasi, dan mengamankan server yang menawarkan layanan tertentu yang dibahas.

Bab-bab ini memperkenalkan jaringan area lokal (LAN), jaringan area luas (WAN), LAN nirkabel, alat untuk menjelajahi jaringan, dan sistem nama domain. Kursus semacam itu juga dapat menyoroti topik selanjutnya seperti *Hypertext Transfer Protocol* (HTTP) dan komputasi awan. buku ini didedikasikan untuk Internet, dan TCP/IP adalah protokol yang digunakan di seluruh Internet, kita akan menghabiskan lebih banyak waktu mempelajari TCP/IP melalui teks.

Buku ini terbagi menjadi 8 bab. Bab 1 berisi tentang pengantar jaringan, akan menjelaskan tentang apa yang disebut jaringan, media jaringan dan jenis jaringan. Bab 2 akan menjelaskan tentang protokol jaringan atau aturan-aturan yang harus dipenuhi dalam suatu jaringan. Bab 3 menjelaskan tentang internet pengalamatan hingga keamanan jaringan internet. Bab 4 akan diajarkan tentang membangun jaringan lokal hingga memberikan pelatihan bagaimana membangun keamanan jaringan lokal yang sudah dibuat.

Bab 5 akan menjelaskan protokol internet, dalam bab ini akan menjelaskan tentang OSI layer. Bab 6 memberikan gambaran tentang aturan-aturan jaringan yang dibutuhkan dalam jaringan internet. Bab 7 akan membahas tentang Sistem Nama Domain? Domain Name System (DNS) mengenai infrastruktur DNS, kinerja DNS, hingga pencegahan spam berbasis DNS. Bab 8 sekaligus menutup buku jilid 1 ini akan menjelaskan tentang BIND dan DHCP, dalam bab ini menjelaskan beberapa konsep dari bab TCP/IP mengenai pengalamatan Protokol Internet (IP), dengan materi dari bab DNS. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Agustus 2023
Penulis

Dr. Agus Wibowo, M.Kom, M.Si, MM.

DAFTAR ISI

BAB 1
PENGANTAR JARINGAN

Semua orang tahu apa itu Internet, bukan? Kita semua menggunakannya, kita mengandalkannya, dan masyarakat kita hampir menjadi tergantung padanya. Namun, apakah kita benar-benar memahami apa itu Internet dan bagaimana cara kerjanya? Bagi banyak orang, Internet adalah entitas yang samar-samar. Itu ada di luar sana dan kami terhubung dengannya, dan pesan secara ajaib melewatinya. Dalam buku teks ini, kami menjelajahi Internet dan banyak komponennya.

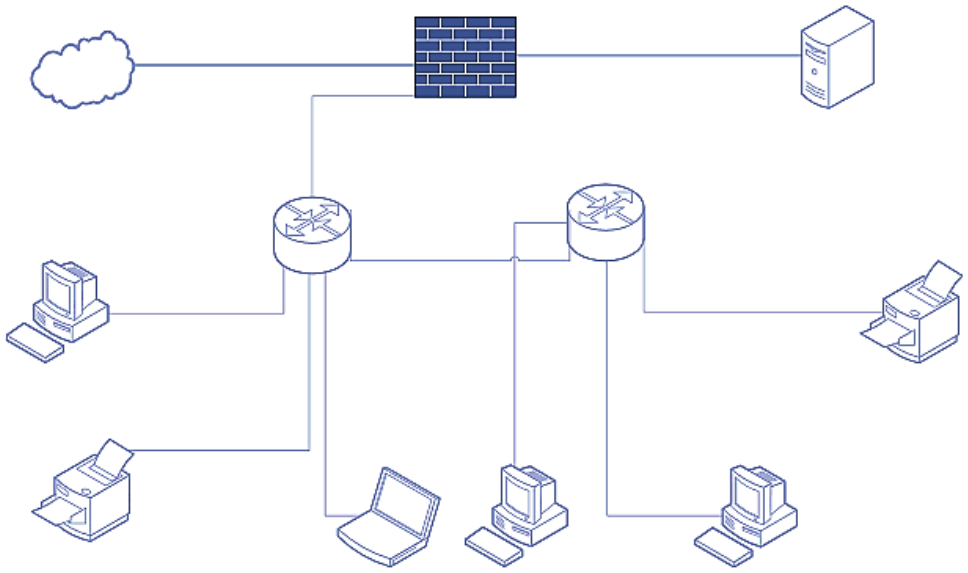
Ini bukan sekadar buku teks jaringan. Buku teks jaringan telah ada selama beberapa dekade. Banyak dari mereka menjelaskan secara rinci perangkat keras dan protokol yang membentuk jaringan. Beberapa hanya khusus untuk satu protokol, *Transmission Control Protocol/Internet Protocol* (TCP/IP). Yang lain mengeksplorasi cara menulis program yang kami gunakan di jaringan. Namun buku lain menjelaskan cara mengamankan jaringan Anda dari serangan. Buku teks ini telah mengambil pendekatan yang berbeda dalam menjelajahi Internet. Kami akan membahas dasar-dasarnya (jaringan secara umum, perangkat keras, dan TCP/IP), tetapi kemudian, kami akan menjelajahi protokol penting yang kami gunakan untuk membuat Internet berfungsi. Menggunakan beberapa studi kasus, kami akan memeriksa perangkat lunak paling populer yang membantu mendukung aspek Internet: alat TCP/IP, server Domain Name System (DNS), server Dynamic Host Configuration Protocol (DHCP), server web, server proxy, caching web, load balancing, dan perangkat lunak komputasi awan.

Dalam bab ini, kita akan mulai dengan dasar-dasarnya. Kami pertama-tama akan menjelajahi perangkat keras jaringan dan beberapa protokol jaringan yang lebih populer (tidak termasuk TCP/IP). Kami juga akan melihat beberapa topik terkait jaringan seperti deteksi dan koreksi kesalahan, enkripsi, dan cache jaringan. Sebagian besar materi ini (dan TCP/IP, tercakup dalam Bab 3) mengatur tahapan untuk sisa buku teks. Jadi, duduk, santai, dan pelajari tentang salah satu teknologi paling signifikan di planet ini.

1.1 JARINGAN KOMUNIKASI

Mari kita mulai dengan beberapa dasar. Jaringan adalah sekelompok hal yang terhubung. Jaringan komputer adalah kumpulan sumber daya komputer yang terhubung. Sumber daya ini termasuk tetapi tidak terbatas pada semua jenis komputer, perangkat jaringan, perangkat seperti printer dan menara cakram optik, MODEM (MODEM adalah singkatan dari MODulation DEModulation), kabel yang menghubungkan sumber daya ini, dan, tentu saja, manusia. Sebagian besar komputer yang terhubung ke jaringan adalah komputer pribadi dan laptop, tetapi ada juga server, komputer mainframe, dan superkomputer. Baru-baru ini, perangkat seluler seperti ponsel pintar dan tablet telah menjadi bagian dari jaringan komputer. Kami juga dapat menyertakan perangkat yang bukan komputer tujuan umum tetapi masih mengakses jaringan, seperti smart television (TV), perangkat Global Positioning System (GPS), sensor, dan konsol game. Gambar 1.1 mengilustrasikan jaringan komputer yang

dihubungkan oleh dua perangkat jaringan. Pada gambar, ada banyak komputer dan server (pojok kanan atas) serta dua printer yang terhubung ke dua router, yang menghubungkan perangkat lainnya ke Internet dengan firewall yang diatur antara jaringan dan Internet.



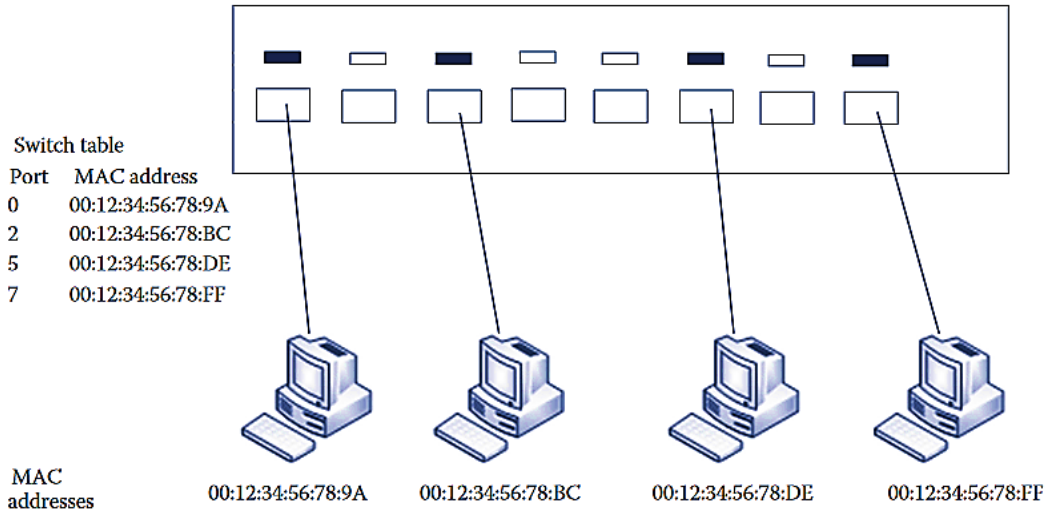
Gambar 1.1 Contoh jaringan komputer.

Perangkat Jaringan

Perangkat jaringan adalah perangkat yang menerima pesan dari satu sumber daya di jaringan dan menentukan cara menyampaikan pesan di sepanjang jaringan. Perangkat jaringan mungkin langsung terhubung ke sumber daya tujuan, atau mungkin terhubung ke perangkat jaringan lain, dalam hal ini meneruskan pesan ke perangkat berikutnya. Perangkat jaringan yang umum adalah hub, switch, router, dan gateway. Perangkat ini dapat berupa kabel, nirkabel, atau keduanya.

Hub adalah perangkat jaringan yang paling primitif. Ini beroperasi dengan menerima pesan dan meneruskannya ke semua sumber daya yang terhubung dengannya. Hub terkadang disebut sebagai multiport repeater, karena tugasnya adalah mengulang pesan yang masuk di semua port (koneksi)nya. Perhatikan bahwa ini tidak sama dengan multicast, yang akan kita bahas nanti di bagian ini.

Hub juga menangani deteksi tabrakan dengan meneruskan sinyal macet ke semua perangkat yang terhubung, jika beberapa pesan tiba pada waktu yang sama. Sinyal macet menunjukkan bahwa terjadi tabrakan pesan di antara perangkat yang terhubung dengannya. Saat ini terjadi, setiap perangkat yang mencoba berkomunikasi menunggu beberapa waktu acak sebelum mencoba mengirim ulang pesannya. Hub sebagian besar sudah usang saat ini karena perangkat unggulan seperti switch jaringan.



Gambar 1.2 Sakelar jaringan dan tabelnya.

Tabel 1.1 Contoh Tabel Perutean

Tujuan Jaringan	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	10.15.8.1	10.15.8.164	10
10.15.8.0	255.255.252.0	On-link	10.15.8.164	266
10.15.8.164	255.255.255.255	On-link	10.15.8.164	266
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	306
192.168.56.0	255.255.255.0	192.168.56.1	192.168.56.1	276
192.168.56.1	255.255.255.255	192.168.0.100	192.168.56.1	276
192.168.0.100	255.255.255.255	127.0.0.1	127.0.0.1	306
224.0.0.0	240.0.0.0	On-link	192.168.56.1	276
255.255.255.255	255.255.255.255	On-link	10.15.8.164	266

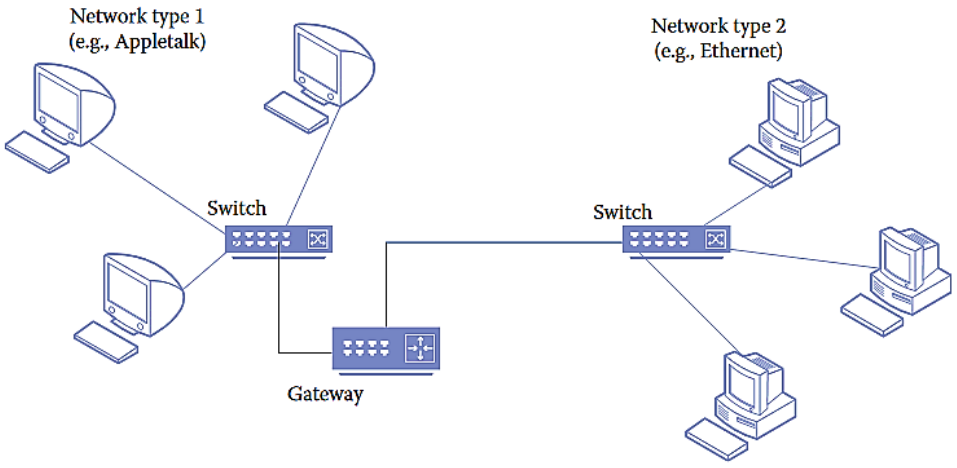
Sakelar jaringan meneruskan pesan masuk ke satu sumber daya. Sakelar menggunakan alamat tujuan pesan untuk menentukan perangkat tujuan pengiriman pesan. Alamat ini dikenal sebagai alamat tingkat rendah dan disebut sebagai alamat perangkat keras atau alamat kontrol akses media (MAC). Saklar juga dikenal sebagai jembatan MAC.

Saat perangkat terhubung ke sakelar, sakelar memperoleh alamat MAC perangkat itu dan menyimpannya dalam tabel. Tabel ini adalah daftar sederhana untuk setiap port pada sakelar, alamat perangkat keras perangkat yang terpasang disimpan. Pada Gambar 1.2, kita melihat sebuah sakelar yang menghubungkan empat perangkat dan tabel yang dipertahankan oleh sakelar tersebut. Perhatikan bahwa karena sakelar memiliki lebih dari empat port, beberapa nomor port saat ini tidak digunakan dalam tabel.

Saat menerima pesan, sakelar memeriksa alamat MAC tujuan dan meneruskan pesan ke port yang sesuai, seperti yang ditentukan dalam tabelnya. Beberapa sakelar juga dapat beroperasi pada alamat jaringan (mis., Alamat IP). Perbedaan utama antara switch dan router adalah router beroperasi pada alamat jaringan secara eksklusif dan bukan pada alamat perangkat keras. Kami akan membedakan antara jenis sakelar nanti di bab ini.

Router beroperasi pada tingkat tumpukan protokol jaringan yang lebih tinggi daripada sakelar. Router menggunakan alamat jaringan tujuan pesan untuk merutekan pesan ke langkah selanjutnya melalui jaringan. Alamat jaringan ini tergantung pada jenis protokol jaringan. Dengan asumsi TCP/IP, alamat jaringan adalah alamat Protokol Internet versi 4 (IPv4) atau Protokol Internet versi 6 (IPv6). Langkah selanjutnya tidak harus berarti perangkat tujuan. Router merutekan pesan di seluruh jaringan, sehingga diteruskan ke titik berikutnya di jaringan yang membawa pesan lebih dekat ke tujuannya. Ini mungkin ke komputer tujuan, ke switch jaringan, atau ke router lain. Oleh karena itu, router melakukan penerusan. Contoh tabel perutean jaringan ditunjukkan pada Tabel 1.1 (isi tabel perutean, termasuk istilah seperti netwmask dan antarmuka, akan dibahas nanti di bab ini). Metrik adalah biaya penggunaan rute yang ditunjukkan. Nilai ini digunakan oleh router untuk menentukan hop yang harus diambil pesan selanjutnya, saat pesan tersebut bergerak melintasi jaringan.

Gateway adalah router yang menghubungkan berbagai jenis jaringan secara bersamaan. Lebih khusus lagi, gateway memiliki kemampuan untuk menerjemahkan pesan dari satu protokol ke protokol lainnya. Ini ditangani oleh perangkat keras atau perangkat lunak yang memetakan konten nondata pesan dari protokol jaringan sumber ke protokol jaringan tujuan. Gambar 1.3 menunjukkan dua jenis jaringan area lokal (LAN) yang dihubungkan oleh gateway. Gateway seperti router, kecuali diposisikan di tepi jaringan. Di dalam LAN, sumber daya dihubungkan oleh router atau switch. Router dan gateway menghubungkan LAN bersama-sama. Seringkali, koneksi LAN ke Internet dilakukan melalui gateway daripada router.

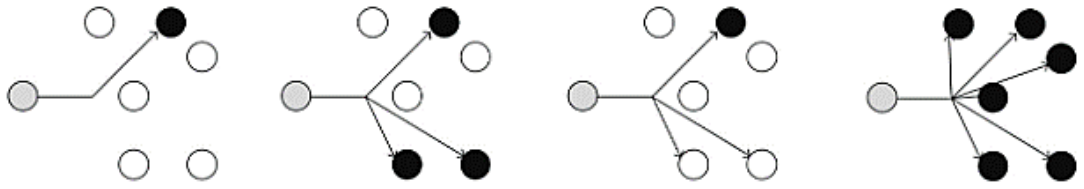


Gambar 1.3 Memposisikan gateway di edge jaringan.

Perhatikan bahwa istilah switch, router, dan gateway terkadang digunakan secara bergantian. Misalnya, sakelar yang juga menggunakan alamat IP terkadang disebut sebagai sakelar lapisan 3, meskipun beroperasi seperti router. Router terkadang disebut sebagai gateway apakah mereka menerjemahkan protokol atau tidak. Seperti yang dinyatakan sebelumnya, kita akan mengunjungi protokol nanti di bab ini, dan pada saat itu, kita akan meninjau kembali peran switch, router, dan gateway.

Bentuk khas komunikasi melalui jaringan adalah unicast. Bentuk komunikasi ini memungkinkan pesan dikirim dari satu perangkat sumber ke satu perangkat tujuan. Sumber dan tujuan biasanya akan membuka saluran komunikasi (sesi) di mana komunikasi mungkin satu arah atau dua arah (dalam hal ini, dikenal sebagai mode dupleks). Namun, ada kalanya komunikasi adalah situasi satu-ke-banyak atau banyak-ke-banyak. Ini terjadi ketika satu atau lebih perangkat berkomunikasi dengan beberapa perangkat. Artinya, ada beberapa perangkat tujuan yang menjadi tujuan pesan. Komunikasi semacam itu dikenal sebagai multicast. Hub melakukan bentuk multicast terbatas. Alasan yang lebih spesifik untuk multicast terjadi saat server melakukan streaming konten ke banyak tujuan. Daripada menduplikasi pesan di ujung server, pesan dikirim ke jaringan di mana router bertanggung jawab tidak hanya untuk meneruskan konten tetapi juga menduplikasi konten untuk dikirim ke beberapa tujuan. Contoh lain untuk multicast adalah dengan permainan komputer jaringan multipemain. Saat satu pemain melakukan operasi dari dalam perangkat lunak, semua pemain lain harus melihat gerakan itu. Komputer pemain tidak perlu menggandakan pesan untuk dikirim ke semua pemain lain. Sebaliknya, router menangani hal ini dengan menduplikasi pesan, menghasilkan multicast.

Dua bentuk komunikasi lainnya adalah broadcast dan anycast. Siaran adalah pesan yang dikirim dari satu perangkat ke perangkat lainnya di subnetwork lokalnya (kami mendefinisikan subnet nanti di bab ini). Hub adalah perangkat siaran jaringan yang tugasnya menyiarkan ke semua perangkat di jaringan lokalnya. Meskipun ini seperti multicast di mana pesan digandakan, ini adalah multicast dalam pengaturan yang sangat terbatas. Dengan kata lain, multicast adalah siaran di mana tujuan tidak dibatasi ke subnetwork lokal. Terakhir, anycast merupakan kompromi antara unicast dan multicast. Dengan anycast, ada beberapa tujuan yang berbagi alamat IP yang sama. Sebuah pesan dikirim yang dapat pergi ke salah satu tujuan ini tetapi selalu dialihkan ke tujuan terdekat. Dengan cara ini, siaran mana pun akan mencapai tujuannya dalam waktu sesingkat mungkin. Kami akan merujuk ke multicast, broadcast, dan anycast dari waktu ke waktu melalui teks. Jika kita tidak secara eksplisit menyebutkan bentuk komunikasi, asumsikan bahwa itu adalah unicast.



Gambar 1.4 Membandingkan unicast, multicast, anycast, dan broadcast.

Gambar 1.4 mengilustrasikan perbedaan antara unicast, multicast, anycast, dan broadcast. Di subnetwork ini, enam perangkat terhubung ke perangkat jaringan kami (saklar dalam hal ini). Di sebelah kiri, kami memiliki pesan unicast, di mana sakelar mengirimkan pesan ke satu perangkat. Selanjutnya, kami memiliki multicast, di mana sakelar mengirimkan pesan ke beberapa perangkat tertentu. Kemudian, kami memiliki anycast, di mana sakelar

mengirim pesan ke semua perangkat dengan alamat IP yang sama, tetapi hanya satu yang perlu menerimanya. Terakhir, di sebelah kanan, pesan disiarkan ke semua perangkat.

Server

Server kata dapat digunakan untuk menggambarkan perangkat keras atau perangkat lunak. Banyak pengguna akan menyebut server sebagai perangkat fisik, karena mereka mengasosiasikan komputer tertentu dengan tugas layanan (seperti server file atau server web). Namun, sebenarnya server adalah kombinasi dari perangkat fisik yang ditunjuk untuk menangani permintaan layanan dan paket perangkat lunak server. Dengan menyebut perangkat fisik sebagai server, kami menyiratkan bahwa perangkat tersebut tidak digunakan untuk tujuan lain apa pun. Hal ini bisa benar bisa juga tidak. Misalnya, sebuah komputer dapat diatur sebagai server web tetapi juga dapat menjadi komputer pribadi seseorang yang dia gunakan untuk komputasi biasa. Di sisi lain, perangkat lunak server sangat penting, karena tanpanya, perangkat keras hanyalah perangkat komputasi. Oleh karena itu, kita perlu menginstal, mengkonfigurasi, dan menjalankan perangkat lunak server di komputer. Dalam kebanyakan kasus, kami akan merujuk ke server sebagai perangkat lunak, karena kami sangat tertarik untuk mengeksplorasi cara kerja perangkat lunak server.

Penggunaan server dalam jaringan menunjukkan bahwa jaringan diatur menggunakan model client-server. Dalam jaringan seperti itu, sebagian besar sumber daya adalah klien, yang membuat permintaan ke server jaringan. Peran server dalam jaringan semacam itu adalah untuk melayani permintaan klien. Bentuk lain dari jaringan dikenal sebagai jaringan peer-to-peer, di mana tidak ada server, sehingga semua sumber daya dianggap sama. Kita mungkin melihat jaringan peer-to-peer sebagai jaringan di mana setiap sumber daya dapat menangani permintaan yang diberikan atau di mana tidak ada klien yang meminta sumber daya lain.

Ada banyak jenis server, dan tidak diragukan lagi, Anda pasti pernah mendengar, berinteraksi dengan, atau bahkan menginstal beberapa server ini. Buku ini membahas beberapa di antaranya dan mencakup, secara rinci, cara menyiapkan, mengonfigurasi, mengamankan, dan memantau berbagai server populer. Tabel 1.2 menyajikan daftar server jaringan. Ini tidak dimaksudkan untuk menjadi daftar lengkap, tetapi kami akan membahas banyak hal selama buku ini.

Komputasi awan telah membawa perubahan dalam cara organisasi menyediakan layanan jaringan. Alih-alih secara fisik menampung server di dalam domain mereka sendiri, banyak organisasi membeli ruang dan kekuatan pemrosesan dari perusahaan yang mendukung aktivitas semacam itu melalui cloud. Dengan cara ini, organisasi tidak perlu membeli, memasang, atau memelihara perangkat keras. Ini semua ditangani oleh perusahaan tempat mereka membeli akses. Cloud perusahaan adalah kumpulan server (terkadang disebut server farm) yang menjalankan mesin virtual (VM). VM adalah kombinasi perangkat keras, perangkat lunak, dan data, di mana server meniru komputer khusus dari beberapa platform, tersedia melalui jaringan ke pengguna akhir VM. Perangkat keras VM adalah server. Perangkat lunak VM adalah program yang melakukan emulasi. Data VM adalah lingkungan yang dibuat oleh pengguna, seperti perangkat lunak yang diinstal, file konfigurasi, dan sebagainya. Perusahaan seperti Amazon dan Microsoft dapat mendukung VM dari ratusan atau ribuan klien yang berbeda. Dalam banyak kasus, VM bukan dari komputer pengguna akhir yang

sederhana tetapi dari server seperti server web dan server proxy. Kami membahas komputasi awan di Bab 11 dan melihat Amazon Web Services (AWS) di Bab 12.

Tabel 1.2 Jenis-Jenis Server Jaringan

Jenis Server	Penggunaan/Peran	Contoh
Server aplikasi	Menjalankan perangkat lunak aplikasi di seluruh jaringan, sehingga aplikasi tidak perlu diinstal pada komputer individu. Dalam jaringan klien-server yang lebih lama, server aplikasi sering digunakan sehingga klien dapat tanpa disk (tanpa hard disk) dan dengan demikian mengurangi biaya klien.	ColdFusion, Enterprise Server, GlassFish, NetWeaver, Tomcat, WebLogic, WebSphere, Windows Server
Server basis data	Menyediakan akses jaringan ke database backend.	DB2, Informix, Microsoft SQL Server, MySQL, Oracle
ServerDHCP	Memberikan alamat IP dinamis kepada klien dalam jaringan.	DHCP Server, dnsmasq, ISC DHCP
Server email	Mentransfer pesan email antara klien lokal dan server email; memberikan klien akses ke email masuk (pesan email dapat disimpan di server, klien, atau keduanya).	Apache James, IBM Lotus Domino, Microsoft Exchange Server, Novell Groupmail, Novell NetMail, OpenSMTPD, Postfix/Sendmail
Server file	Utilitas penyimpanan bersama yang dapat diakses melalui jaringan; ada banyak subclass dari file server seperti FTP server, database server, application server, dan web server.	See the more specific types of servers
Server FTP	Mendukung File Transfer Protocol, sehingga klien dapat mengunggah dan mengunduh file ke dan dari server. FTP tidak aman, sehingga tidak umum digunakan, kecuali untuk kemungkinan transfer file anonim (memungkinkan klien untuk mentransfer file publik tanpa masuk). SFTP adalah FTP melalui koneksi SSH, sedangkan FTPS adalah FTP yang dibuat aman dengan menggunakan koneksi SSL (SSL dibahas di Bab 3).	Cerberus FTP Server, FileZilla Server, ftpd, freeFTPd, Microsoft Internet Information Services, WS FTP
Server game	Sama seperti server aplikasi, kecuali bahwa server ini didedikasikan untuk menjalankan satu game ke ribuan atau jutaan pengguna di Internet.	Varies by game
Daftar server	Jenis server email yang mengelola daftar, sehingga pesan email dapat dikirim ke semua yang ada di daftar. Fitur ini dapat dibangun ke dalam server email.	LISTSERV, Mailman, Sympa

Server nama (atau server DNS)	Menyelesaikan nama domain (alias IP) menjadi alamat IP dan/atau berfungsi sebagai cache.	BIND, Cisco Network Registrar, dnsmasq, Simple DNS Plus, PowerDNS
Server cetak	Menghubungkan klien ke printer untuk mempertahankan antrian pekerjaan cetak, seperti yang dikirimkan oleh klien. Memberikan umpan balik kepada klien tentang status pekerjaan cetak.	CUPS, JetDirect, NetBOIS, NetWare
Server proxy	Perantara antara klien web dan server web. Server proxy memiliki beberapa peran: meng-cache halaman web dalam suatu organisasi untuk penarikan yang lebih efisien; menyediakan mekanisme keamanan untuk mencegah permintaan atau tanggapan yang tidak diinginkan; dan memberikan beberapa anonimitas untuk klien web. Server proxy terbalik digunakan sebagai ujung depan ke server web untuk penyeimbangan muatan.	CC Proxy Server, Internet Security and Acceleration Server, Squid, WinGate
server SSH	Menggunakan Protokol Secure Shell untuk menerima koneksi dari komputer jarak jauh. Perhatikan bahwa FTP tidak mengizinkan enkripsi, tetapi versi aman tersedia melalui SSH seperti SFTP.	Apache MINA SSHD, Copssh, OpenSSH, Pragma Systems SSH Server, Tectia SSH Server
Server VM	Server yang membuat mesin virtual itu sendiri dapat digunakan sebagai komputer, server web, server email, server basis data, dan sebagainya.	Microsoft Hyper-V, VMware vSphere Server
server web	Bentuk server file untuk menanggapi permintaan HTTP dan mengembalikan halaman web. Jika halaman web memiliki skrip sisi server, ini dijalankan oleh server web, menjadikannya server file dan server aplikasi.	AOLserver, Apache, Internet Information Services, NGINX, OpenLiteSpeed, Oracle HTTP Server, Oracle WebLogic Server, TUX web server

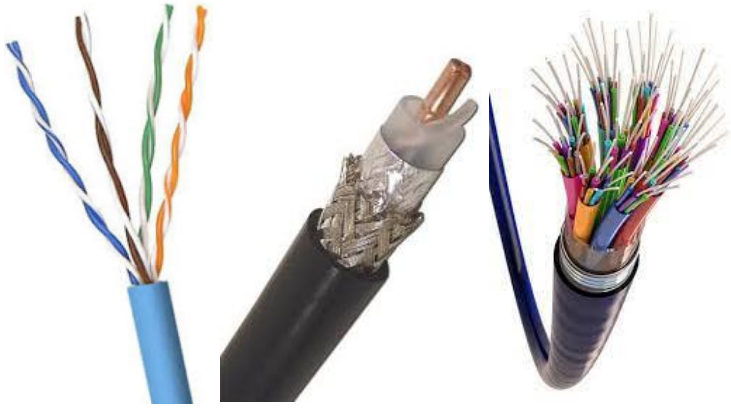
1.2 MEDIA JARINGAN

Kita perlu menghubungkan komponen jaringan komputer bersama-sama. Koneksi dapat datang dalam dua bentuk berbeda: kabel dan nirkabel. Kami menyebut koneksi ini sebagai media. Kita dapat menilai efisiensi suatu media berdasarkan lebar pitanya, yaitu jumlah data yang dapat dibawa melalui media tersebut dalam satuan waktu, seperti bit per detik.

Bentuk kabel dari konektivitas jaringan adalah pasangan kawat bengkok (atau hanya pasangan bengkok), kabel koaksial, dan kabel serat optik. Bentuk paling awal dari pasangan kawat bengkok digunakan di saluran telepon, dan ini adalah bentuk tertua dan paling primitif. Ini juga menjadi yang termurah. Meskipun perangkat tambahan telah dibuat dengan teknologi

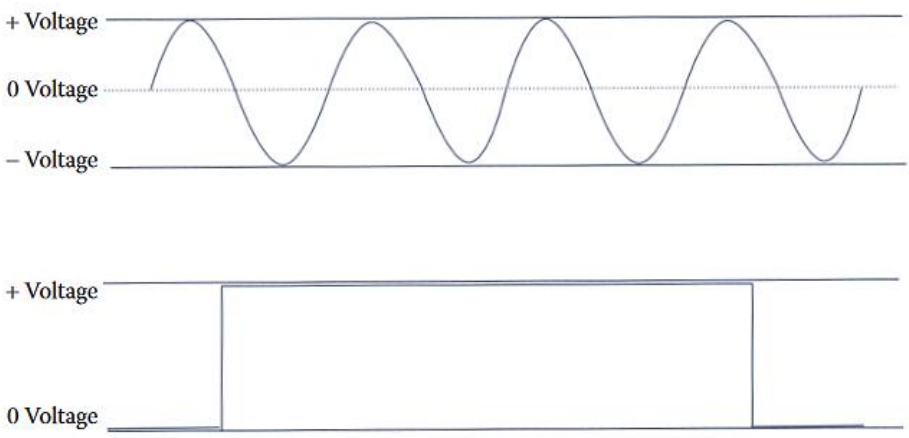
kabel terpilin untuk meningkatkan lebar pitanya, ia masih memiliki lebar pita terendah di antara bentuk kabel. Dalam banyak kasus, penggunaan kabel coaxial dan kabel fiber optic telah menggantikan kabel twisted-wire untuk jaringan komputer jarak jauh.

Baik pasangan kawat bengkok dan kabel koaksial mengandung seutas kawat di mana arus listrik dapat dibawa. Agar kabel dapat mengalirkan arus, kabel harus utuh dan terhubung ke sesuatu di ujung lainnya. Artinya, kawat membentuk sirkuit tertutup. Di satu ujung, sumber tegangan (misalnya, baterai) dimasukkan, dan arus dikatakan mengalir ke ujung kabel ke ujung lainnya. Meskipun logam apa pun dapat digunakan sebagai konduktor arus listrik, kami biasanya menggunakan kawat tembaga, yang ditemukan pada pasangan kawat bengkok dan kabel koaksial.



Gambar 1.5 bentuk kabel jaringan

Gambar 1.5 mengilustrasikan tiga bentuk kabel. Di sebelah kiri, ada empat helai pasangan kabel yang dipilin, ditempatkan menjadi satu kabel. Di tengah, ada kabel koaksial, dengan kabel keluar dari ujungnya. Di sekeliling kabel terdapat lapisan-lapisan isolasi. Di sebelah kanan, ada beberapa ribu keping kabel serat optik. Setiap kabel individu fleksibel, seperti yang mungkin Anda perhatikan dari seluruh kumpulan kabel yang tertekuk.



Gambar 1.6 AC (atas) dan DC (bawah).

Arus yang mengalir ke kabel dapat berupa salah satu dari dua bentuk. Arus yang keluar dari soket listrik kita biasanya berupa arus searah (DC). Arus searah berarti arus mengalir pada tegangan tetap. Untuk telekomunikasi, kami ingin menggunakan arus bolak-balik (AC). Arus bolak-balik terlihat seperti bentuk gelombang, yang merupakan serangkaian osilasi antara puncak dan nilai. Gambar 1.6 mengilustrasikan perbedaan antara DC dan AC. Karena catu daya kita sering dalam bentuk DC, komputer perlu mengubah dari DC ke AC saat menggunakan jaringan kabel.

Dengan DC, voltase adalah satu-satunya properti yang menarik. Namun, dengan AC, ada sejumlah nilai berbeda yang terkait dengan arus yang mengalir di kabel: amplitudo, frekuensi, periode, fase, dan panjang gelombang. Amplitudo menentukan ketinggian puncak (nilai sumbu Y pada ketinggiannya). Frekuensi menentukan jumlah gelombang (siklus) per detik. Kami menyatakan nilai ini dengan menggunakan istilah hertz (atau kilohertz, megahertz, dll.), di mana 1 hertz berarti satu siklus penuh per detik dan satu kilohertz adalah 1000 siklus per detik. Periode adalah jumlah waktu untuk satu gelombang penuh atau titik awal ke puncak ke lembah dan sebaliknya. Ini menunjukkan waktu sepanjang sumbu X yang diperlukan gelombang untuk pergi dari titik awalnya pada sumbu Y ke lokasi yang sama setelah mencapai puncak dan lembah. Frekuensi dan periode saling berkebalikan, atau periode = 1 /frekuensi dan frekuensi = 1 /periode. Istilah fase berlaku untuk beberapa bentuk gelombang yang digabungkan menjadi satu, yang dapat terjadi, misalnya, saat multiplexing sinyal. Akhirnya, panjang gelombang sama dengan periode dikalikan dengan kecepatan suara. Untuk tujuan jaringan, kami akan membatasi minat kami hanya pada frekuensi dan fase. Kabel awal yang digunakan untuk menghantarkan listrik untuk keperluan telekomunikasi (mis., telegraf dan telepon) mengalami kebisingan yang disebabkan oleh interferensi elektromagnetik dari berbagai sumber, termasuk kabel lainnya. Munculnya pasangan kawat bengkok memungkinkan sinyal dibawa dengan gangguan yang sangat berkurang. Arus yang mengalir ke bawah dua kabel secara paralel menyebabkan dua medan magnet yang membatalkan untuk melindungi konten yang dibawa ke bawah kabel. Twisted-wire pair dengan cepat menjadi dasar jaringan telepon umum dan akhirnya digunakan untuk jaringan komputer.

Bentuk terlindung masih umum digunakan untuk jaringan telepon darat, sedangkan bentuk tidak terlindung dapat ditemukan di LAN bergaya Ethernet. Keuntungan dari pasangan kawat bengkok adalah sebagai berikut: tipis dan fleksibel; beberapa pasang kawat bengkok dapat dikelompokkan menjadi satu kabel pendukung (puluhan, ratusan, dan bahkan ribuan); kabelnya murah untuk diproduksi; dan cross-talk diminimalkan. Kerugian utama adalah bandwidth yang terbatas, tetapi juga menderita dari panjang kabel maksimum yang terbatas, yang mengharuskan pengulang ditempatkan di antara panjang kabel.

Kabel koaksial terdiri dari kabel konduktor tembaga yang ditempatkan di dalam bahan isolasi fleksibel, yang dikelilingi oleh lapisan konduktor lain (foil atau jalinan tembaga). Kabel dalam dan lapisan luar menyediakan sirkuit, sedangkan lapisan luar juga berfungsi sebagai pelindung untuk mengurangi antarmuka luar. Konstruksi ini ditempatkan ke dalam kulit luar yang agak fleksibel. Kabel koaksial menawarkan bandwidth yang lebih tinggi daripada sebagian besar bentuk pasangan kawat bengkok, dan panjang kabel yang lebih panjang (melalui pasangan kawat bengkok) dimungkinkan dan karenanya membutuhkan lebih sedikit

pengulang. Kerugian dari kabel koaksial adalah lebih mahal daripada pasangan kawat bengkok. Itu juga kurang fleksibel dan lebih tebal. Jelas, kabel koaksial telah banyak digunakan dalam pemasangan kabel di bawah tanah, mengirimkan televisi kabel ke rumah tangga di seluruh dunia. Kabel koaksial juga merupakan pilihan yang menarik untuk pemasangan kabel LAN jika tidak keberatan dengan biaya yang sedikit lebih besar.

Kabel serat optik sama sekali tidak mengandalkan arus listrik, melainkan sepotong kaca. Informasi dikirim melalui kabel serat optik dengan mentransmisikan pulsa cahaya. Ini memiliki sejumlah keunggulan dibandingkan bentuk komunikasi berbasis kabel. Sebelum kita melihat ini, mari kita periksa kabel serat optik lebih dekat.

Kabel terdiri dari satu atau lebih serat optik, yang masing-masing merupakan bagian dari kaca yang diekstrusi (atau mungkin plastik). Setiap serat berdiameter sekitar 125 mikron (sekitar 0,005 inci). Sebagai perbandingan, rambut manusia tidak lebih dari 180 mikron dan seringkali lebih kecil (ironisnya, rambut manusia juga dapat digunakan seperti serat optik untuk mengirimkan cahaya). Serat terdiri dari tiga bagian. Di pusat serat, terdapat inti, yang mampu mentransmisikan pulsa cahaya pada frekuensi yang berbeda dan pada jarak yang jauh. Selanjutnya, bahan tersebut tidak membias, artinya panjang gelombang cahaya tidak boleh berubah saat cahaya bergerak. Inti dikelilingi oleh apa yang disebut bahan kelongsong, lapisan yang memantulkan kembali cahaya apa pun ke dalam serat. Di luar kelongsong, ada lapisan plastik untuk melindungi serat dari kerusakan atau kelembapan. Sebuah bundel, satu kabel, kemudian dapat terdiri dari sejumlah serat, mungkin sebanyak ribuan serat individu. Core umumnya datang dalam dua bentuk, single-mode dan multi-mode. Inti single-mode mentransmisikan sinar laser inframerah dengan panjang gelombang yang lebih besar, sedangkan inti multi-mode mentransmisikan cahaya inframerah dengan panjang gelombang yang lebih rendah.

Kabel serat optik memiliki banyak keunggulan dibandingkan kabel tembaga. Ini terdaftar sebagai berikut:

- Bandwidth yang jauh lebih besar karena satu serat optik dapat membawa sebanyak 111 gigabit per detik dengan bandwidth tipikal 10 hingga 40 gigabit per detik.
- Jarak yang jauh lebih jauh di mana sinyal dapat berjalan tanpa kehilangan atau membutuhkan pengulang.
- Jauh lebih sedikit kerentanan terhadap pengaruh luar seperti kebisingan radio atau sambaran petir. Kabel serat optik tidak terpengaruh oleh radiasi elektromagnetik seperti kabel tembaga (baik terlindung dan terisolasi atau tidak).
- Kabel optik lebih ringan dan lebih aman karena nonkonduktor.
- Risiko pencurian lebih kecil. Poin terakhir ini aneh untuk dibuat. Namun, ada banyak pencuri tembaga di masyarakat saat ini yang ingin mencuri sumber tembaga dan menjualnya. Karena kabel serat optik tidak terbuat dari tembaga, risiko pencuriannya jauh lebih kecil meskipun serat menjadi barang yang lebih mahal untuk diproduksi.

Kelemahan utama kabel serat optik adalah harganya jauh lebih mahal daripada kabel tembaga. Pengeluaran yang lebih besar lagi adalah mengganti jutaan mil pasangan kawat terpilin dari jaringan telepon umum kita dengan kabel serat optik. Banyak perusahaan telekomunikasi telah menginvestasikan miliaran dolar hanya untuk meningkatkan jaringan

telepon dan menyediakan akses Internet broadband ke rumah tangga. Namun, biaya ini adalah biaya yang dibagi di antara semua pengguna (misalnya, ini adalah bagian dari harga yang Anda bayarkan untuk menerima akses Internet di rumah, terlepas dari apakah rumah tangga Anda sebelumnya memiliki pasangan kabel berpilin atau tidak). Kabel serat optik lebih rapuh dari kabel tembaga dan lebih sulit diperbaiki jika panjang kabel dipotong.

Tidak seperti kabel tembaga, yang mentransmisikan informasi analog (misalnya suara), serat optik mentransmisikan informasi digital secara langsung (1 dan 0 dikirim sebagai pulsa cahaya). Dengan demikian, ada sedikit pekerjaan yang diperlukan di kedua ujung transmisi untuk mengirim atau menerima data komputer. Untuk kabel tembaga, informasi pertama-tama mungkin perlu dimodulasi (dikonversi dari bentuk digital ke bentuk analog) untuk transmisi dan kemudian didemodulasi (dikonversi dari bentuk analog kembali ke bentuk digital) pada saat diterima. MODEM adalah perangkat yang melakukan dua tugas ini.

Bentuk media lainnya adalah komunikasi nirkabel. Sesuai dengan namanya, bentuk komunikasi ini tidak melibatkan kabel apapun. Sebaliknya, informasi diubah menjadi bentuk gelombang dan disiarkan melalui udara. Biasanya, bentuk gelombangnya adalah frekuensi radio (RF) atau gelombang frekuensi tinggi (microwave); Namun, itu juga bisa menjadi sinyal inframerah. Komunikasi nirkabel jauh lebih terbatas dalam jarak yang dapat dilalui pesan, sehingga perlu ada titik akses nirkabel terdekat. Untuk rumah seseorang, titik akses nirkabel kemungkinan besar adalah hub nirkabel, router, atau MODEM, yang kemudian terhubung ke jalur komunikasi kabel.

Komunikasi nirkabel juga kurang dapat diandalkan daripada bentuk kabel karena ada banyak hal yang mengganggu sinyal: pintu atau dinding, perangkat yang memancarkan sinyal radionya sendiri, atau perangkat listrik di dekatnya. Selain itu, komunikasi nirkabel membutuhkan mekanisme keamanan tambahan untuk menghindari seseorang dengan mudah menguping komunikasi. Enkripsi umumnya digunakan saat ini untuk mengurangi kekhawatiran ini, tetapi hanya bermanfaat jika pengguna tahu cara menggunakannya. Praktik terbaik lainnya juga tersedia untuk mengamankan komunikasi nirkabel, seperti cat metalik pada dinding eksterior untuk membatasi jumlah kebocoran sinyal yang keluar dari gedung. Akhirnya, kecepatan komunikasi nirkabel tertinggal dari bentuk kabel. Di sisi lain, ada jauh lebih sedikit biaya dalam menggunakan komunikasi nirkabel karena tidak ada kabel untuk diletakkan (mungkin di bawah tanah) sambil menawarkan kemudahan yang lebih besar karena pengguna komputer tidak terikat pada satu lokasi.

Mari kita periksa bandwidth secara lebih formal. Istilah ini mendefinisikan jumlah data yang dapat ditransmisikan pada suatu waktu. Secara formal, bandwidth diberikan dalam satuan hertz, seperti yang diperkenalkan sebelumnya. Istilah ini menyampaikan sepersekian detik yang diperlukan untuk sejumlah transmisi, seperti 1 bit. Namun, untuk kenyamanan, kami biasanya menyebut bandwidth sebagai jumlah bit yang ditransmisikan per detik. Tabel 1.3 memberikan perbandingan kemampuan bandwidth dari bentuk transmisi yang dibahas sebelumnya. Anda dapat melihat bahwa kabel serat optik memberikan bandwidth terbesar, sedangkan penggunaan frekuensi radio nirkabel menyediakan bandwidth paling sedikit.

Mari kita coba menempatkan bandwidth dalam perspektif dan dalam konteks sejarah. Gambar jpg kecil berwarna akan berukuran sekitar 1 MByte, yaitu 8 Mbit atau sekitar 8 juta

bit. Dengan bandwidth 1 bit per detik, dibutuhkan 8 juta detik untuk mentransfer file ini ke seluruh jaringan (92 ½ hari). Tentu saja, semua bandwidth saat ini jauh lebih besar dari 1 bit per detik. Pada bandwidth teoretis maksimum untuk kabel serat optik, transmisi akan memakan waktu 0,00008 detik.

Pada pertengahan hingga akhir 1970-an, ketika komputer pribadi pertama kali dipelopori, sebagian besar bentuk telekomunikasi menggunakan jaringan telepon (yang menggunakan pasangan kabel terpilin), yang mengharuskan pengguna komputer di rumah mengakses jaringan telepon melalui MODEM. MODEM asli memiliki baud rate 300 bit per detik. Baud rate tidak sama dengan bandwidth, karena baud rate menentukan jumlah perubahan yang dapat terjadi per detik. Perubahan akan didasarkan pada ukuran unit yang ditransmisikan. Misalnya, jika kita mentransmisikan bukan bit tetapi 2 simbol bit, maka baud rate akan dua kali lipat dari bandwidth. Namun, demi perbandingan historis ini, kami akan menganggapnya setara.

Pada awal 1980-an, baud rate meningkat menjadi 1200 bit per detik, dan pada pertengahan 1980-an, naik menjadi 4800 dan kemudian 9600 bit per detik. Pada 1990-an, kecepatan meningkat lagi menjadi 14.400, 28.800, dan kemudian 56.000 bit per detik terakhir (yang merupakan batas teoretis untuk sinyal yang dibawa melalui jaringan telepon analog). Seperti yang kita lihat pada Tabel 1.3, kabel twisted-wire berpelindung dan kabel koaksial memiliki bandwidth yang berkisar antara 10 Mbps (juta bit per detik) dan 100 Mbps, dan bandwidth kabel serat optik berkisar antara 100 Mbps dan 100 Gbps (miliar bit per Kedua). Kami dapat menambahkan tiga bentuk komunikasi tambahan ini, semuanya nirkabel. Kecepatan minimum untuk jaringan seluler 3G adalah 200 Kbps (seribu bit per detik), dan kecepatan minimum untuk jaringan seluler 4G diperkirakan 100 Mbps, sedangkan komunikasi nirkabel melalui kartu jaringan nirkabel diperkirakan antara 11 Mbps (802.11 b) dan 600 Mbps (802.11n). Tabel 1.4 menunjukkan waktu transfer untuk masing-masing media yang dibahas dalam paragraf ini dan paragraf terakhir untuk file gambar 1 MB kita.

Tabel 1.3 Bandwidth Media Jaringan

Jenis	Bandwith Minimal	Bandwidth Maksimum (Teoretis).
Pasangan kawat terpilin terlindung	10 Mbps	100Mbps
Pasangan kawat bengkok tanpa pelindung	10 Mbps	1000Mbps
Kawat koaksial	10 Mbps	100Mbps
Kabel serat optik	100Mbps	100 Gbps
Nirkabel menggunakan frekuensi radio	9 Kbps	54Mbps

Tabel 1.4 waktu download untuk berbagai bandwidth

Medium	Waktu Download
MODEM (300 bps)	7 Jam, 24 Menit, 27 Detik
MODEM (1200 bps)	1 Jam, 51 Menit, 7 Detik
MODEM (14.400 bps)	13 Menit, 53 Detik
MODEM (28.800 bps)	9 Menit, 16 Detik
MODEM (56.000 bps)	2 Menit, 23 Detik.
MODEM (300 mbps)	0.027 detik
Kabel Twisted dan Coaxial-minimum/maksimum	0.8 detik/0.08 detik
Kabel Twisted (Unshielded)-maximum	0.008 detik
Fiber Optic – minimum/maximum	0.08 detik / 0.00008 detik
3G	0.67 detik
4G	0.08 detik
Wireless (802.11n)	0.013 detik

Perangkat keras jaringan

Kami telah membahas perangkat jaringan yang merekatkan jaringan bersama. Namun, bagaimana komputer individu mendapatkan akses ke jaringan? Ini dilakukan oleh salah satu dari dua perangkat, kartu antarmuka jaringan (NIC) atau MODEM. Di sini, kami mengeksplorasi apa kedua perangkat ini.

NIC digunakan untuk menghubungkan komputer ke jaringan komputer. Ada berbagai jenis NIC tergantung pada jenis jaringan yang digunakan. Misalnya, kartu Ethernet digunakan untuk menghubungkan komputer ke jaringan Ethernet, sedangkan Adaptor Token Ring digunakan untuk menghubungkan komputer ke jaringan Token Ring. Karena bentuk paling umum dari jaringan lokal saat ini adalah Ethernet, kami biasanya menggunakan kartu Ethernet sebagai NIC.

NIC adalah papan sirkuit elektronik yang meluncur ke salah satu slot ekspansi komputer pada motherboard. Kartu itu sendiri minimal akan berisi sirkuit yang diperlukan untuk berkomunikasi melalui jaringan. Dengan demikian, kartu jaringan harus memiliki alat untuk mengubah data biner yang disimpan di komputer menjadi jenis informasi yang diperlukan untuk media jaringan (misalnya, sinyal elektronik atau pulsa cahaya). Selain itu, data yang dikirim ke jaringan harus diubah dari urutan byte (atau kata) menjadi urutan bit. Artinya, NIC menerima data dari memori sebagai kata atau byte dan kemudian mengirimkan bit secara berurutan atau menerima bit secara berurutan dari jaringan dan mengkompilasinya kembali menjadi byte dan kata untuk dikirim ke memori.

Dengan biaya pembuatan komponen elektronik yang sangat berkurang, NIC telah meningkat secara dramatis dari waktu ke waktu. Saat ini, NIC akan menyertakan banyak fitur lain seperti penanganan interupsi, akses memori langsung untuk memindahkan data secara langsung ke atau dari memori tanpa intervensi CPU, dan mekanisme penanganan tabrakan, untuk beberapa nama. Selain itu, NIC modern seringkali memiliki banyak buffer untuk menyimpan paket masuk atau keluar sehingga sebagian pesan dapat disimpan hingga pesan lengkap tiba. NIC juga dapat menangani beberapa tugas komunikasi sederhana seperti

mengonfirmasi penerimaan pesan. NIC akan memiliki alamat perangkat kerasnya sendiri yang unik, umumnya dikenal sebagai alamat MAC.

NIC diproduksi dengan alamat ini, tetapi hari ini, alamat ini dapat diubah oleh sistem operasi atau ditetapkan secara acak. Dua alasan untuk mengizinkan perubahan alamat MAC adalah untuk meningkatkan keamanan (untuk menghindari spoofing alamat MAC) dan untuk mendukung virtualisasi jaringan. Komputer mungkin memiliki beberapa NIC jika diinginkan. Dalam kasus seperti itu, komputer diberi beberapa alamat yang berbeda (mis., Beberapa alamat IP).

NIC memiliki port atau koneksi yang menonjol dari komputer. Jaringan secara fisik terhubung ke port ini melalui beberapa bentuk plug. Banyak jenis colokan yang digunakan, tetapi yang paling umum digunakan saat ini dikenal sebagai konektor RJ-45, yang terlihat seperti colokan telepon (konektor RJ-11) tetapi sedikit lebih besar (RJ-45 memiliki empat pasang). kabel, sedangkan RJ-11 memiliki tiga pasang kabel, sehingga konektor RJ-45 lebih besar).

Banyak NIC yang lebih baru juga dapat menangani komunikasi nirkabel. NIC nirkabel adalah salah satu yang mengirimkan sinyal radio ke perangkat terdekat yang disebut titik akses. Titik akses terhubung ke bagian kabel jaringan. Jadi, NIC berkomunikasi ke access point, yang kemudian berkomunikasi dengan switch atau router ke sumber daya lain.

Sinyal radio yang dipancarkan NIC dapat berupa salah satu dari empat frekuensi. Yang paling umum adalah sinyal frekuensi radio tinggi pita sempit. Sinyal radio dibatasi dalam jarak biasanya tidak lebih dari beberapa ribu kaki sebelum sinyal menurun ke titik di mana ada kehilangan data. Dua alternatif adalah menggunakan sinar infra merah atau sinar laser. Kedua pendekatan ini membutuhkan komunikasi line-of-sight. Artinya, perangkat tempat NIC harus berada dalam garis pandang yang jelas dengan titik akses. Ini adalah kelemahan yang membatasi pergerakan perangkat sehingga jarang digunakan untuk komputer tetapi sangat mungkin digunakan untuk perangkat stasioner atau saat perangkat berada dalam jarak yang lebih dekat (beberapa kaki), seperti remote control televisi.

Istilah MODEM adalah singkatan dari MODulation DEModulation. Kedua istilah ini mengungkapkan konversi informasi digital menjadi informasi analog (modulasi) dan informasi analog menjadi informasi digital (demodulasi). Kedua operasi konversi ini diperlukan jika kita ingin mengkomunikasikan informasi biner (data komputer) melalui media analog seperti jaringan telepon umum. MODEM akan terhubung ke saluran telepon Anda dengan cara tertentu, dan sebelum data dapat dikirim, itu akan dimodulasi. Secara khusus, data biner akan digabungkan menjadi sebuah nada, dan nada tersebut akan disiarkan melalui saluran telepon analog. Di ujung penerima, nada akan didengar oleh MODEM dan didemodulasi kembali ke data biner.

Masalah dengan menggunakan MODEM adalah ketergantungan pada bandwidth yang relatif rendah yang tersedia melalui jaringan telepon umum. Kecepatan MODEM asli dibatasi hingga 300 bit per detik. Selama beberapa dekade, kecepatan meningkat tetapi mencapai kecepatan maksimum 56.000 bit per detik, karena saluran telepon memaksakan pembatasan ini. Alasan pembatasan ini adalah saluran telepon diatur untuk membawa rentang frekuensi yang sedikit lebih besar daripada jangkauan pendengaran manusia. Dengan demikian, nada

yang dapat disiarkan MODEM dibatasi berada dalam kisaran itu. Karena jumlah nada yang dapat dipancarkan MODEM terbatas, ada jumlah data yang dapat ditransmisikan per nada juga terbatas.

Sebelum komunikasi broadband tersedia di rumah tangga, pilihan lain untuk menghindari saluran telepon bandwidth rendah adalah dengan menggunakan saluran langsung ke rumah tangga, yang dikenal sebagai digital subscriber loop (DSL), atau saluran kabel koaksial ke rumah Anda yang Anda gunakan. digunakan untuk membawa sinyal televisi kabel. DSL masih menggunakan jaringan telepon, tetapi saluran khusus memberikan bandwidth yang jauh lebih besar daripada 56K yang disebutkan sebelumnya. Untuk berkomunikasi melalui kabel televisi kabel, Anda memerlukan sejenis MODEM yang disebut modem kabel. Saat ini, modem kabel dapat mencapai kecepatan broadband dan tidak lagi diperlukan untuk melakukan modulasi dan demodulasi, sehingga namanya agak ketinggalan jaman. DSL masih digunakan di mana broadband tidak tersedia, seperti di daerah pedesaan.

Seperti disebutkan dalam Tabel 1.4, kami lebih memilih bandwidth yang lebih tinggi daripada yang dapat disediakan oleh MODEM melalui saluran telepon atau bahkan DSL. Saat ini, kami cenderung menggunakan NIC di perangkat kami dan mengandalkan perangkat mirip MODEM yang menghubungkan perangkat rumah kami ke penyedia layanan Internet (ISP) kami. Koneksi ini mungkin melalui serat optik, kabel koaksial, atau pasangan kawat bengkok langsung ke rumah Anda. Apapun metode yang digunakan, data yang dikirimkan sekarang adalah data biner, memungkinkan kita untuk menghindari modulasi dan demodulasi.

1.3 JENIS JARINGAN

Sekarang kita memiliki beberapa pemahaman dasar tentang apa itu jaringan, mari kita membedakan antara jenis-jenis jaringan. Kami telah menyebutkan jaringan klien-server versus jaringan peer-to-peer. Ini adalah salah satu klasifikasi jenis jaringan. Dua cara lain untuk mengklasifikasikan jaringan adalah berdasarkan ukuran dan bentuknya. Bentuk jaringan didasarkan pada topologi. Kami menjelajahinya di Bagian 1.2.1. Ukuran jaringan tidak dimaksudkan untuk menyampaikan jumlah perangkat yang terhubung, tetapi jarak relatif antara perangkat seperti di dalam ruangan, di dalam gedung, melintasi kota, atau lebih besar. Kami menjelajahi berbagai jenis jaringan menurut ukurannya di Bagian 1.2.2. Perlu diingat bahwa bentuk jaringan tradisional adalah di mana semua sumber daya berada dalam jarak yang relatif dekat, seperti di dalam satu gedung atau bahkan satu lantai gedung. Kami akan secara formal mendefinisikan jaringan yang komponennya berada dalam satu pengaturan lokal sebagai LAN.

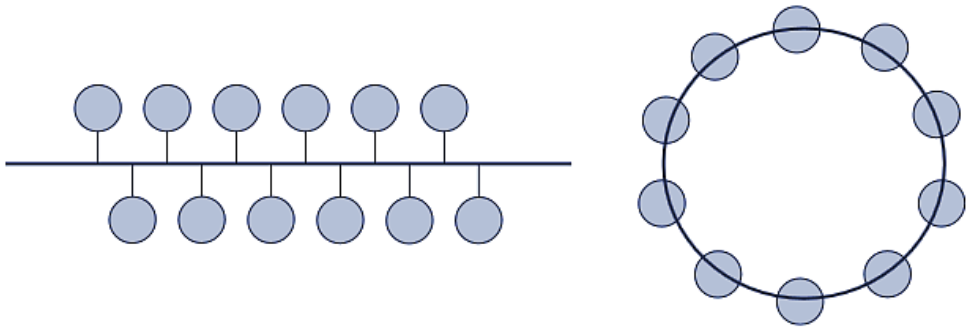
Sebagian besar jaringan di planet ini adalah bentuk LAN, dan sebagian besar menggunakan beberapa bentuk koneksi kabel; namun, beberapa perangkat mungkin memiliki koneksi nirkabel ke jaringan. Secara historis, perangkat dalam LAN sering dihubungkan dengan kabel twisted-wire atau kabel koaksial, dan beberapa LAN masih menggunakan teknologi ini. Saat ini, lebih umum menemukan LAN menggunakan kabel serat optik untuk konektivitas.

Topologi Jaringan

Bentuk LAN biasanya disebut sebagai topologinya. Topologi menentukan cara perangkat jaringan terhubung bersama. LAN awal (dari 1960-an hingga 1980-an) mungkin

telah terhubung sepanjang satu jalur, yang dikenal sebagai bus. Formasi ini menghasilkan topologi bus, terkadang juga disebut sebagai jaringan Ethernet, karena Ethernet awalnya menggunakan topologi bus. Pesaing awal lainnya adalah topologi ring, di mana setiap sumber daya terhubung langsung ke tetangganya, sehingga sumber daya akan terhubung ke dua sumber daya lainnya. Gambar 1.7 mengilustrasikan dua topologi ini, di mana bus ditampilkan di sebelah kiri dan ring di sebelah kanan.

Topologi bus menderita masalah yang signifikan. Karena semua sumber daya berbagi satu baris, hanya satu pesan yang dapat ditransmisikan pada panjangnya setiap saat. Jika beberapa sumber mencoba menggunakan jalur pada saat yang sama, tabrakan akan terjadi. Teknologi Ethernet mengembangkan strategi untuk menangani tabrakan yang dikenal sebagai akses berganda carrier-sense dengan deteksi tabrakan (CSMA/CD). Perangkat yang ingin menggunakan jaringan akan mendengarkan jaringan terlebih dahulu untuk menentukan apakah sedang digunakan secara aktif. Jika tidak, itu tidak hanya akan menempatkan pesannya ke jaringan tetapi juga akan mendengarkan jaringan. Jika pesan ini bertabrakan dengan yang lain, maka jaringan sebenarnya akan berisi gabungan dua (atau lebih) pesan. Oleh karena itu, apa yang ada di jaringan sekarang bukanlah apa yang ditempatkan mesin ke jaringan. Jika benturan terdeteksi, perangkat pengirim mengirimkan sinyal kemacetan untuk memperingatkan semua perangkat lain bahwa telah terjadi benturan. Perangkat yang menyebabkan tabrakan, bersama dengan perangkat lain yang menunggu, akan menunggu beberapa waktu secara acak sebelum mencoba lagi. Topologi bus rentan terhadap kemacetan lalu lintas ketika jaringan tumbuh cukup besar sehingga sumber daya dapat mencoba menggunakan jaringan pada waktu yang bersamaan.

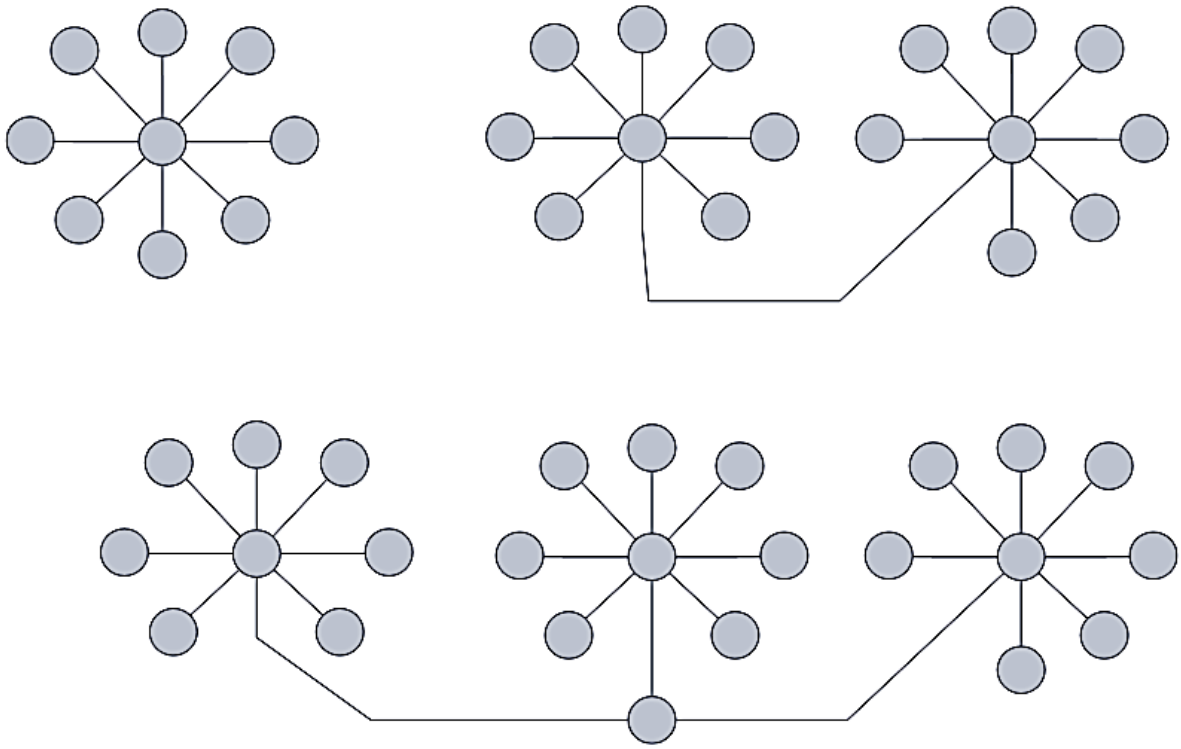


Gambar 1.7 Topologi bus dan ring.

Jaringan ring tidak mengalami kemacetan karena pesan tidak berbagi satu jaringan melainkan dikirim dari node ke node hingga mencapai tujuannya. Namun, jaringan dering akan menderita dalam dua cara. Pertama, jika ada satu node yang gagal, itu dapat mengisolasi sebagian dari jaringan dan mencegah beberapa sumber daya berkomunikasi dengan yang lain. Selain itu, jika deringnya besar, latensi komunikasi meningkat, karena pesan apa pun mungkin akan memiliki jalur yang lebih panjang untuk mencapai tujuannya. Cincin bisa searah atau dua arah; cincin dua arah membantu mengurangi masalah ini sampai batas tertentu. Misalnya, pertimbangkan cincin satu arah dari lima sumber daya, A, B, C, D, dan E, di mana A mengirim ke B, yang mengirim ke C, dan seterusnya, sementara E mengirim ke A. Jika E turun, B, C, dan D sama sekali tidak dapat berkomunikasi dengan A. Dalam ring dua arah, jika E turun, B, C, dan

D masih dapat berkomunikasi dengan A dengan melewatkan sinyal dalam arah yang berlawanan di sekitar ring.

Topologi ketiga, yang lebih mahal, adalah jaringan bintang. Dalam topologi seperti itu, semua sumber daya terhubung ke satu titik pusat komunikasi. Jenis topologi ini jauh lebih efisien daripada topologi ring, karena semua pesan mencapai tujuannya hanya dalam dua (atau lebih sedikit) lompatan. Jaringan bintang juga lebih kuat daripada jaringan cincin karena kegagalan setiap simpul tidak memutuskan jaringan, kecuali tentu saja itu adalah simpul pusat. Namun, biaya topologi bintang adalah Anda sekarang harus mendedikasikan sumber daya untuk membentuk titik pusat jaringan. Ini biasanya adalah perangkat jaringan (mis., Hub dan sakelar), yang saat ini tidak mahal dan sangat berharga untuk investasi; namun, itu juga bisa berupa komputer atau bahkan server, yang akan menjadi peralatan yang lebih mahal. Kelemahan lain dari topologi star adalah hub/switch pusat akan memiliki jumlah koneksi yang terbatas (misalnya, 16), sehingga jumlah sumber daya yang membentuk jaringan akan terbatas. Ada varian jaringan bintang seperti bintang yang diperluas di mana dua hub atau sakelar terhubung. Gambar 1.8 mengilustrasikan beberapa bentuk topologi bintang. Contoh kiri atas adalah jaringan bintang standar. Di sebelah kanan, jaringan dua bintang terhubung ke bus. Di bagian bawah gambar, jaringan bintang tiga terhubung sendiri ke jaringan bintang, atau ada jaringan bintang hierarkis, di mana hub jaringan bintang tiga terhubung ke hub lain.



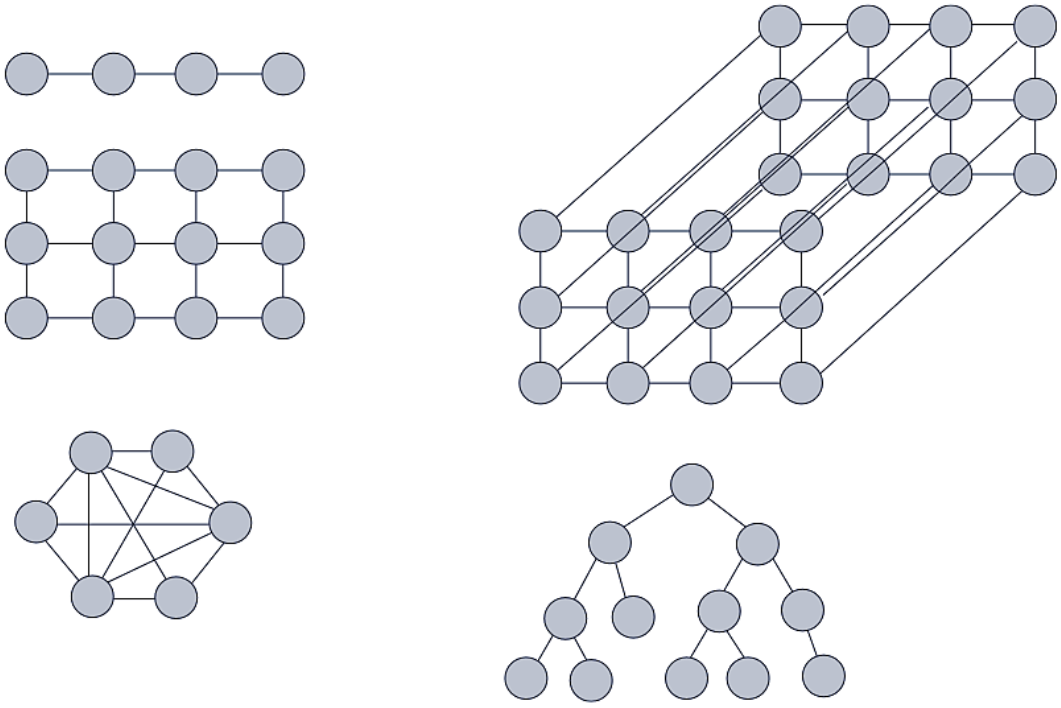
Gambar 1.8 berbagai organisasi topologi bintang.

Bentuk keempat dari topologi jaringan adalah mesh atau topologi tetangga terdekat. Seperti jaringan dering, tetangga terhubung langsung. Jaringan cincin dapat dianggap sebagai tetangga terdekat satu dimensi (1D). Jaringan mesh dapat memiliki dimensi yang lebih besar. Misalnya, jaringan dua dimensi (2D) akan menghubungkan sumber daya hingga empat

tetangga (bayangkan ini sebagai jaring atau matriks tempat sumber daya terhubung ke sumber daya di atas, bawah, kiri, dan kanannya). Jaringan tiga dimensi (3D) menambahkan dimensi lain hingga enam koneksi. Jaringan empat dimensi (4D) sering disebut hypercube karena lebih sulit untuk dibayangkan. Paling ekstrim, kami memiliki jaringan yang terhubung penuh, di mana setiap sumber daya memiliki koneksi langsung ke setiap sumber daya lainnya. Ini sangat mahal dan, dalam banyak kasus, merupakan pemborosan konektivitas karena ada beberapa sumber daya yang mungkin tidak pernah berkomunikasi dengan sumber daya lain. Varian lain dari topologi tetangga terdekat adalah topologi pohon, di mana simpul-simpul di tingkat teratas terhubung ke beberapa sumber daya di tingkat berikutnya. Gambar 1.9 menunjukkan beberapa tata letak tetangga terdekat ini (jaring 1D dan 2D di kiri atas, jala 3D di kanan atas, jala hierarkis atau pohon di kanan bawah, dan jala yang terhubung sepenuhnya di kiri bawah).

Saat ini, jaringan bintang dengan sakelar pusat atau router adalah bentuk yang paling umum karena menawarkan kompromi yang sangat baik antara pengurangan lalu lintas pesan, jaringan yang kuat, dan komunikasi latensi rendah, semuanya tanpa biaya yang besar. Biasanya, jaringan bintang mungkin menggabungkan 10 hingga 20 sumber daya (batas atas adalah jumlah koneksi yang tersedia di sakelar). Jaringan area lokal kemudian dihubungkan dengan menghubungkan beberapa sakelar ke satu atau lebih router.

Mari kita asumsikan bahwa kita memiliki sejumlah sumber daya komputer (komputer, printer, server file, dan jenis server lainnya) dan sejumlah sakelar. Sakelar masing-masing memiliki 25 port fisik, di mana salah satu dari 25 port ini dicadangkan untuk terhubung ke router. Kami menghubungkan 24 sumber daya ke sakelar dan kemudian menghubungkan sakelar ke router dengan koneksi ke-25. Jika router kami juga mengizinkan hingga 24 koneksi, jaringan lokal kami akan terdiri dari 1 router, 24 sakelar, dan hingga 576 sumber daya.



Gambar 1.9 Topologi Tetangga Terdekat (Mesh)

Perutean, seperti sakelar, mungkin memiliki satu sambungan tambahan, sehingga kita dapat menghubungkan perute ke gateway (titik keberadaan kita di Internet).

Klasifikasi Jaringan

Ide untuk menghubungkan switch/router secara bersamaan dikenal sebagai chaining. Sayangnya, ada batasan praktis berapa banyak mata rantai yang dapat muncul dalam sebuah rantai. Misalnya, Anda tidak akan dapat mengembangkan contoh sebelumnya menjadi puluhan ribu sumber daya. Pada titik tertentu, kita akan membutuhkan repeater untuk meningkatkan sinyal yang datang dari sumber daya yang paling jauh, dan pada akhirnya, kita juga akan membutuhkan router.

Mari kita perjelas konsep chaining dengan sebuah contoh. Pertimbangkan LAN organisasi besar yang khas. Kami akan berasumsi bahwa organisasi ada di satu atau beberapa gedung, tetapi mungkin ada di beberapa lantai dan dengan ratusan hingga ribuan sumber daya untuk terhubung ke LAN. Sebuah lantai mungkin panjangnya ratusan kaki. Kami mungkin menemukan beberapa lusin sumber daya di lantai itu. Di satu area lantai, sumber daya lokal terhubung ke sakelar. Area lantai lainnya memiliki sakelar lain untuk mendukung sumber daya di dekatnya. Sakelar ini (bersama dengan yang lainnya) akan terhubung ke router pusat. Router ini kemudian dapat terhubung ke router serupa di lantai lain gedung. Kami akan memiliki koneksi serupa di gedung terdekat lainnya. Bangunan akan dihubungkan melalui router tambahan. LAN organisasi kami bukan lagi LAN tunggal tetapi LAN dari LAN. Setiap lantai dari setiap gedung mungkin merupakan satu LAN. Alternatifnya, jika ada cukup sumber daya di satu lantai, atau jika gedungnya sangat besar, kita mungkin menemukan banyak LAN di satu lantai. LAN organisasi terdiri dari LAN yang lebih kecil. Ini bisa berulang, di mana LAN yang lebih kecil terdiri dari LAN yang lebih kecil lagi.

Jadi, kita melihat bahwa LAN hanyalah sebuah istilah untuk sebuah jaringan yang mungkin dapat membentang hingga ribuan kaki atau terkandung dalam satu area kecil seperti sebuah rumah. Untuk lebih memahami ukuran LAN, beberapa istilah telah diperkenalkan. LAN sering mengacu pada jaringan beberapa lusin hingga beberapa ratus sumber daya dalam jarak dekat. Jaringan yang lebih kecil adalah jaringan area pribadi (PAN), yang akan kita temukan di banyak rumah tangga. PAN mungkin menghubungkan beberapa komputer (termasuk laptop dengan kartu nirkabel) dan printer ke router pusat, yang juga menghubungkan jaringan internal rumah ke jaringan ISP melalui kabel serat optik. Di sisi ekstrim lainnya, ada jaringan area kampus (CAN), yang merupakan jaringan yang membentang di banyak gedung organisasi. Istilah kampus tidak harus membatasi bentuk ini hanya untuk universitas, perguruan tinggi, dan sekolah, tetapi juga dapat diterapkan untuk perusahaan yang diadakan di beberapa gedung di satu lokasi. Sumber daya terjauh dalam CAN mungkin berjarak sekitar satu atau dua mil.

Di sisi lain, kami beralih dari LAN ke jaringan area luas (WAN). Sedangkan LAN akan sering terkandung dalam, mungkin, beberapa mil persegi, WAN adalah sesuatu yang lebih besar. Ada beberapa kelas WAN. Pertama, ada jaringan area metropolitan (MAN). MAN mungkin mencakup beberapa mil atau seluruh wilayah metropolitan. Misalnya, beberapa kota memiliki jaringan sendiri yang menyediakan sumber daya seperti kalender acara, akses gratis

ke sumber informasi, akses gratis ke Internet, dan email gratis. MAN sering dihubungkan dengan kabel berbasis Ethernet tetapi mungkin juga memiliki bentuk koneksi lain, termasuk nirkabel, dalam bentuk sinyal radio atau gelombang mikro. London, Inggris, dan Jenewa, Swiss, adalah dua kota yang memiliki MAN. Pada lingkup yang lebih besar adalah WAN yang terdiri dari situs-situs tertentu dalam suatu negara atau di seluruh dunia. Ini mungkin termasuk, misalnya, kantor pemerintah dan/atau militer AS, universitas beserta kampus cabangnya, atau jaringan berbayar seperti America On-line dan CompuServe. Bentuk ketiga dari WAN adalah Internet itu sendiri. Perbedaan antara MAN, WAN dari situs tertentu, dan Internet hampir tidak ada artinya saat ini karena sebagian besar koneksi jarak jauh sedemikian rupa sehingga baik Anda berada di jaringan terbatas atau tidak, Anda masih memiliki akses ke Internet. Jadi, kita mungkin menganggap MAN atau WAN berbayar sebagai salah satu tempat Anda memiliki akses penuh ke Internet sambil juga memiliki akses ke sumber daya tertentu yang tidak dapat diakses oleh orang lain.

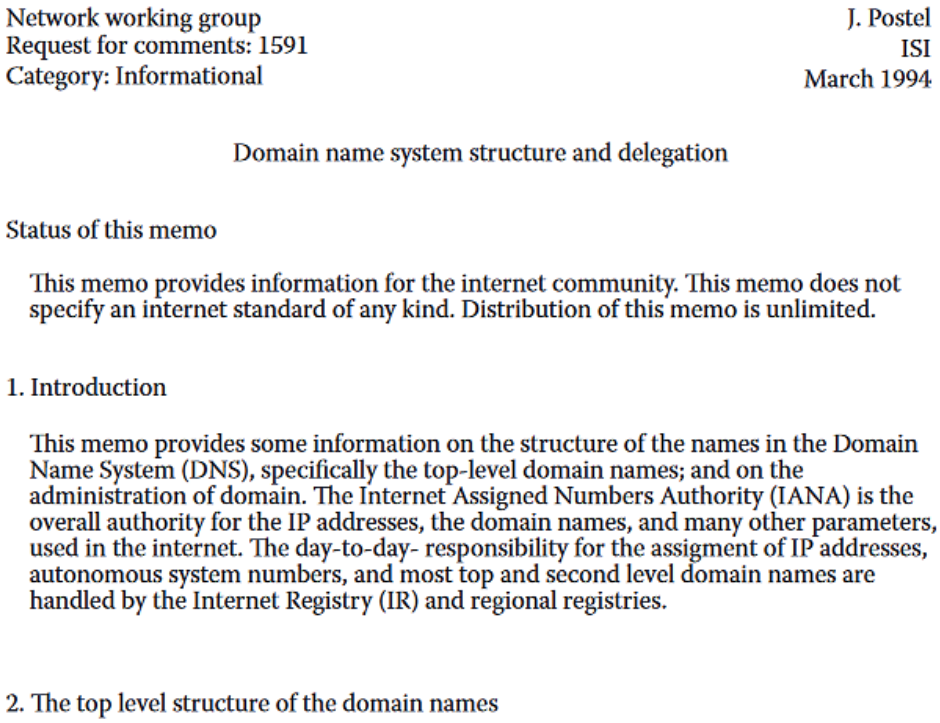
BAB 2
PROTOKOL JARINGAN

Saat Anda mempelajari jaringan, ada istilah membingungkan tertentu yang akan muncul berulang kali. Ini termasuk permintaan komentar (RFC), standar, protokol, tumpukan protokol, dan model. Meskipun semua istilah ini memiliki arti yang terkait, mereka adalah istilah yang terpisah dan penting, jadi kita perlu membahasnya sebelum melanjutkan. Seperti yang dinyatakan sebelumnya, RFC adalah singkatan dari permintaan komentar. RFC dikeluarkan ketika sebuah kelompok (atau individu) yang bekerja di bidang teknologi ingin berbagi beberapa pendekatan inovatif untuk memecahkan masalah di dalam bidang tersebut. Mereka menerbitkan RFC untuk meminta umpan balik. Sebagai contoh, pada implementasi Advanced Research Projects Agency Network (ARPANET), peneliti memposting RFC untuk kebutuhan jaringan seperti menulis perangkat lunak host, mendefinisikan antarmuka antara komputer dan apa yang disebut sebagai prosesor pesan antarmuka (IMP) dan mengidentifikasi berbagai jenis pesan data. Ini awalnya diterbitkan dalam RFC 1, 7, dan 42, masing-masing, antara tahun 1969 dan 1970. Ada ribuan RFC, dan Anda dapat memeriksanya di www.rfc-editor.org. Gambar 1.10 menunjukkan bagian pertama dari RFC 1591, yang merupakan RFC mengenai struktur dan delegasi DNS. Setelah grup memperoleh umpan balik melalui RFC, langkah selanjutnya adalah menggunakan komentar bersama dengan proposal mereka sendiri untuk menentukan serangkaian standar yang harus dipatuhi oleh setiap orang di lapangan. Mengingat standar ini, peneliti lain dapat mulai mengimplementasikan solusi mereka sendiri. Selama implementasi mengikuti standar, implementasi harus dapat digunakan oleh siapa pun di lapangan dengan mengikuti standar yang sama.

Ini membawa kita ke istilah protokol. Protokol, sebagaimana didefinisikan dalam bahasa Inggris, adalah sistem aturan yang menjelaskan perilaku atau komunikasi yang tepat. Protokol komunikasi, atau protokol jaringan, adalah seperangkat aturan yang menentukan bagaimana data akan dikirim, dialamatkan, diarahkan, dan ditafsirkan. Protokol memenuhi standar yang ditentukan untuk masalah yang diberikan. Misalnya, satu protokol adalah *Domain Name System* (DNS). RFC awal untuk DNS diterbitkan pada tahun 1983 di bawah RFC 882 dan RFC 883. DNS, protokolnya, pertama kali diterapkan pada tahun 1984. Perhatikan bahwa RFC untuk DNS di masa mendatang telah menggantikan RFC 882.

Sebagian besar protokol jaringan menyelesaikan berbagai tugas mereka tidak melalui satu terjemahan pesan ke dalam sinyal jaringan yang dibawa melalui jaringan tetapi melalui serangkaian terjemahan. Ini dilakukan dengan memiliki lapisan dalam protokol. Ketika ada beberapa lapisan, kami biasanya mengacu pada rangkaian pemetaan sebagai tumpukan protokol karena setiap lapisan ditangani oleh protokol yang berbeda. Sebuah lapisan tunggal dalam sebuah protokol adalah seperangkat aturan tersendiri untuk menerjemahkan pesan ke dalam bentuk baru, baik untuk mempersiapkannya untuk transmisi di seluruh jaringan atau untuk mempersiapkan pesan yang dikirimkan untuk disajikan kepada pengguna. Satu istilah tambahan adalah model. Ketika kita berbicara tentang tumpukan protokol *Open System*

Interconnection (OSI), kita sebenarnya mengacu pada model karena OSI tidak pernah diimplementasikan secara fisik. Beberapa juga merujuk ke TCP/IP sebagai model berlapis karena sebenarnya ada banyak implementasi untuk TCP/IP.



Network working group
Request for comments: 1591
Category: Informational

J. Postel
ISI
March 1994

Domain name system structure and delegation

Status of this memo

This memo provides information for the internet community. This memo does not specify an internet standard of any kind. Distribution of this memo is unlimited.

1. Introduction

This memo provides some information on the structure of the names in the Domain Name System (DNS), specifically the top-level domain names; and on the administration of domain. The Internet Assigned Numbers Authority (IANA) is the overall authority for the IP addresses, the domain names, and many other parameters, used in the internet. The day-to-day- responsibility for the assigment of IP addresses, autonomous system numbers, and most top and second level domain names are handled by the Internet Registry (IR) and regional registries.

2. The top level structure of the domain names

Gambar 2.1 Sebagian dari RFC 1591.

Jadi, setelah kita melihat jenis-jenis jaringan, kita dapat fokus pada apa yang membuat jaringan bekerja. Kami telah melihat beberapa perangkat kerasnya, tetapi perangkat keras hanyalah sebagian dari cerita. Agar dua sumber daya dapat berkomunikasi, mereka harus berbicara dalam bahasa yang sama. Semua komunikasi melalui jaringan pada akhirnya dipecah menjadi bit, tetapi perangkat di jaringan perlu memahami apa arti bit tersebut. Kami memerlukan cara untuk menginformasikan perangkat keras dan perangkat lunak saat memeriksa pesan agar dapat menginterpretasikan informasi seperti:

- Kepada siapa pesan ditujukan (alamat tujuan)?
- Perangkat lunak aplikasi apa yang harus memanfaatkan pesan tersebut?
- Berapa lama pesannya?
- Apakah pesan sudah lengkap atau hanya satu paket dari sekian banyak.
- Apakah ada kesalahan dalam pesan yang dikirimkan dan bagaimana menangani kesalahan tersebut.
- Bagaimana seharusnya koneksi dibuat antara sumber daya sumber dan tujuan (jika perlu)?
- Apakah pesan menyertakan data yang disandikan dan/atau dienkripsi.

Detail seperti itu diserahkan kepada protokol jaringan yang digunakan untuk mengemas dan mengirimkan pesan.

Ketika sebuah pesan akan dikirim melalui jaringan, pertama-tama dimulai sebagai produk dari beberapa aplikasi. Oleh karena itu, lapisan tertinggi dalam protokol jaringan biasanya adalah lapisan aplikasi. Pesan ini harus diubah menjadi format yang tepat. Lapisan aplikasi menyerahkan pesan ke lapisan lain, yang akan menambahkan detail lebih lanjut tentang cara penanganan pesan. Misalnya, adalah umum untuk memecah pesan menjadi paket-paket yang lebih kecil. Lapisan yang lebih rendah kemudian harus bertanggung jawab untuk memecah pesan menjadi paket-paket. Setiap paket harus diberi alamat dan diberi nomor secara berurutan sehingga paket dapat disusun kembali sesuai urutannya. Lapisan lain bertanggung jawab atas operasi ini. Protokol komunikasi kemudian beroperasi sebagai serangkaian lapisan. Beberapa lapisan dapat diimplementasikan oleh satu protokol, tetapi ada kemungkinan juga bahwa protokol yang berbeda diterapkan pada tingkat yang berbeda.

Dalam kebanyakan kasus, saat pesan bergerak ke bawah tumpukan protokol, informasi ditambahkan ke awal pesan. Konten tambahan ini dikenal sebagai header, dan setiap lapisan tumpukan protokol dapat menambahkan headernya sendiri. Dalam beberapa kasus, informasi tambahan dapat ditambahkan ke akhir pesan, dalam hal ini informasi tambahan dikenal sebagai footer. Saat menerima pesan, tumpukan protokol beroperasi secara terbalik, menghapus header (dan/atau footer) saat pesan bergerak ke atas lapisan.

Banyak protokol komunikasi yang tersedia. Protokol jaringan yang paling umum digunakan adalah TCP/IP, sebuah rangkaian protokol seperti yang disebutkan sebelumnya. TCP/IP terdiri dari empat lapisan di mana setiap lapisan dapat diimplementasikan melalui sejumlah kemungkinan protokol individu. Namun, TCP/IP bukan satu-satunya paket protokol jaringan.

2.1 PROTOKOL PENGENDALIAN TRANSMISI/*PROTOKOL INTERNET*

TCP/IP terdiri dari dua rangkaian protokol terpisah, TCP atau *Transmission Control Protocol* dan IP atau *Internet Protocol*. Bersama-sama, mereka terdiri dari empat lapisan. Dari atas ke bawah, kita memiliki lapisan aplikasi, lapisan transport, lapisan Internet, dan lapisan tautan. TCP menangani dua lapisan teratas, sedangkan IP menangani dua lapisan terbawah. Pada setiap layer terdapat banyak protokol yang dapat diterapkan. Artinya, setiap layer tidak hanya memiliki satu tetapi banyak implementasi yang berbeda, bergantung pada kebutuhan di balik pesan asli yang akan dikirim.

Buku ini secara singkat menjelaskan dua topik yang menjelaskan cara kami menangani perangkat di jaringan TCP/IP. Semua jaringan dan perangkat dijelaskan melalui alamat jaringan. Bentuk alamat yang paling umum adalah alamat *Internet Protocol version* 4 (IPv4); namun, kami juga menggunakan alamat IPv6 di seluruh bagian Internet. Alamat IPv4 adalah alamat 32-bit dan IPv6 adalah alamat 128-bit. Selain itu, pesan yang dikirim dari satu perangkat ke perangkat lainnya dikirim menggunakan beberapa jenis protokol. Perangkat penerima akan menggunakan protokol itu untuk menafsirkan urutan bit dalam pesan. Protokol ini dilambangkan dengan nomor port. Nomor port tidak hanya menunjukkan bagaimana menginterpretasikan bit tetapi juga bagaimana memanfaatkan pesan. Artinya, port menentukan aplikasi untuk digunakan seperti browser web untuk digunakan saat

perangkat menerima pesan yang disandikan menggunakan *Hypertext Transfer Protocol* (HTTP).

Interkoneksi Sistem Terbuka

Tidak seperti TCP/IP, model OSI bukanlah rangkaian protokol karena tidak diimplementasikan. Sebaliknya, ini disebut sebagai model dengan tujuan menawarkan struktur kepada arsitek jaringan untuk ditargetkan ketika mengembangkan protokol baru. Itu dikembangkan kira-kira pada waktu yang sama dengan TCP/IP dan memiliki banyak komponen yang tumpang tindih atau identik. Namun, OSI menyertakan beberapa fitur yang tidak ada di TCP/IP. OSI terdiri dari tujuh lapisan. Sebagian besar fungsionalitas dari tujuh lapisan ini dapat dipetakan ke empat lapisan TCP/IP. OSI, seperti TCP/IP, menentukan bagaimana komunikasi jaringan dipecah menjadi unit individu (paket) dan bagaimana pada setiap lapisan informasi tambahan ditambahkan ke pesan atau paket sebagai header. Hasilnya adalah paket OSI, ketika ditransmisikan, berisi enam header (lapisan bawah tidak menambahkan header), diikuti oleh data yang merupakan bagian dari pesan yang ditangkap dalam paket ini. Dalam sub-bagian ini, kami mengeksplorasi tujuh lapisan dan peran mereka. Tabel 1.5 mencantumkan tujuh lapisan dan, untuk setiap lapisan, perannya dan unit data yang digunakan.

Tujuh lapisan dikategorikan menjadi dua jenis lapisan: lapisan host dan lapisan media. Lapisan host memberikan detail yang mendukung data yang menyusun pesan. Lapisan media memberikan detail yang digunakan dalam transmisi aktual dan penanganan setiap paket. Saat pesan bergerak ke bawah tumpukan protokol, itu diubah, dengan konten header ditambahkan padanya. Header ini dihapus dari paket saat paket bergerak ke atas lapisan di tujuan (atau di lokasi perantara).

Tabel 2.1 Tujuh Lapisan OSI

Nama Lapisan	Tipe Lapisan	Satuan Data Lapisan	Peran
7. Aplikasi	Lapisan tuan rumah	Data	Perangkat lunak aplikasi menghasilkan/menampilkan pesan
6. Presentasi			Format netral aplikasi
5. Sesi			Menjaga komunikasi antar host
4. Transportasi		Segments	Berurusan dengan kesalahan dan pemesanan paket
3. Jaringan	Lapisan media	Packets/datagrams	Pengalamatan dan kontrol lalu lintas
2. Tautan data		Bit/frame	Koneksi antara dua node jaringan
1. Fisik		Bit	Transmisi jaringan

Lapisan atas adalah lapisan aplikasi. Di sini, beberapa perangkat lunak aplikasi menginisialisasi komunikasi dengan menghasilkan pesan awal. Pesan ini mungkin merupakan permintaan HTTP, seperti yang dihasilkan oleh browser web untuk mendapatkan halaman web, atau mungkin beberapa server email bersiap untuk mengirim pesan email. Saat menerima pesan, pada lapisan inilah pesan dikirim ke pengguna akhir melalui perangkat lunak aplikasi yang sesuai.

Lapisan aplikasi harus dilihat bukan sebagai perangkat lunak (misalnya, Mozilla Firefox, Microsoft Outlook, dan *WinSock File Transfer Protocol* [WS-FTP]) melainkan sebagai protokol yang akan digunakan oleh pesan, seperti HTTP, FTP , atau Protokol Transfer Surat Sederhana (SMTP). Layanan lain yang mungkin digunakan pada lapisan ini adalah bentuk berbagi sumber daya (misalnya, printer), layanan direktori, layanan autentikasi seperti Protokol Akses Direktori Ringan (LDAP), terminal virtual seperti yang digunakan oleh SSH atau telnet, permintaan DNS, real-streaming waktu data (mis., RTSP), dan transmisi soket aman (*Transport Layer Security* [TLS]/*Secure Sockets Layer* [SSL]).

Lapisan presentasi bertanggung jawab untuk menerjemahkan pesan khusus aplikasi menjadi pesan netral. Pengkodean karakter, kompresi data, dan enkripsi dapat diterapkan untuk memanipulasi pesan asli. Salah satu bentuk pengkodean karakter adalah mengubah file yang disimpan dalam *Extended Binary Coded Decimal Interchange Code* (EDCDIC) menjadi *American Standard Code for Information Interchange* (ASCII). Bentuk lainnya adalah menghapus karakter khusus seperti "\0" dari program C/C++. Data hierarkis, seperti *Extensible Markup Language* (XML), dapat diratakan di sini. Perlu dicatat bahwa TCP/IP tidak memiliki fungsi yang setara dengan kemampuan lapisan ini untuk mengenkripsi dan mendekripsi pesan. Sebagai gantinya, dengan TCP/IP, Anda harus menggunakan protokol enkripsi di atas (atau sebelum membuat) paket, atau Anda harus membuat terowongan tempat pesan terenkripsi dapat diteruskan.

Tanggung jawab lapisan sesi adalah untuk membangun dan memelihara sesi antara sumber daya dan sumber daya tujuan. Untuk membuat sesi, diperlukan beberapa bentuk jabat tangan. Model OSI tidak menentukan bentuk jabat tangan. Dalam TCP/IP, ada jabat tangan tiga arah pada lapisan transport TCP, yang melibatkan permintaan, pengakuan, dan pengakuan atas pengakuan. Sesi, juga disebut koneksi, dapat berupa mode dupleks penuh (di mana kedua sumber daya dapat berkomunikasi satu sama lain) atau mode setengah dupleks (di mana komunikasi hanya dalam satu arah).

Setelah sesi ditetapkan, sesi tetap terbuka sampai salah satu dari beberapa situasi muncul. Pertama, sumber daya asal mungkin mengakhiri sesi. Kedua, jika sumber tujuan telah membuka sesi dan tidak mendengar kabar dari sumber sumber dalam beberapa waktu, koneksi ditutup. Jumlah waktu didasarkan pada nilai default yang dikenal sebagai batas waktu. Hal ini biasanya terjadi pada server yang bertindak sebagai mesin tujuan di mana nilai timeout adalah seluruh server (yaitu, apakah nilai default ditetapkan untuk server untuk semua komunikasi?). Kemungkinan ketiga adalah bahwa sumber telah melampaui jumlah pesan yang diizinkan yang telah ditentukan sebelumnya untuk satu sesi. Ini juga digunakan oleh server untuk mencegah kemungkinan perangkat sumber mencoba memonopoli penggunaan server, seperti penolakan serangan layanan. Kemungkinan keempat adalah server menutup koneksi karena beberapa pelanggaran lain dalam komunikasi. Jika koneksi terputus secara tidak normal (salah satu kasus yang tercantum di atas kecuali yang pertama), lapisan sesi mungkin merespons dengan mencoba membangun kembali koneksi. Beberapa pesan yang diteruskan dari satu sumber daya jaringan ke yang lain mungkin berhenti di lapisan sesi (alih-alih berpindah ke lapisan aplikasi). Ini akan mencakup jenis operasi otentikasi dan otorisasi. Selain itu, seperti disebutkan sebelumnya, pemulihan sesi akan dilakukan di lapisan ini. Ada banyak

protokol yang dapat mengimplementasikan lapisan sesi. Di antaranya adalah *Password Authentication Protocol* (PAP), *Point-to-Point Tunneling Protocol* (PPTP), *Remote Procedure Call Protocol* (RPC), *Session Control Protocol* (SCP), dan *Socket Secure Protocol* (SOCKS). Tiga lapisan yang dijelaskan sejauh ini melihat pesan itu sendiri. Mereka memperlakukan pesan sebagai unit data yang berbeda.

Lapisan transport memiliki dua peran. Peran pertama adalah membagi pesan asli (yang panjangnya bervariasi, bergantung pada aplikasi dan isi pesan) menjadi segmen data berukuran tetap. Kedua, lapisan ini bertanggung jawab untuk menjaga transmisi yang dapat diandalkan antara dua titik akhir melalui pengakuan penerimaan paket individual dan melalui multiplexing (dan demultiplexing) beberapa paket yang dibawa dalam sinyal yang sama. Jika sebuah paket gagal diterima, lapisan ini dapat meminta transmisi ulangnya. Intinya, lapisan ini bertanggung jawab untuk memastikan penerimaan yang akurat dari semua paket.

OSI mendefinisikan lima kelas paket berdasarkan kebutuhan aplikasi. Kelas-kelas ini mendikte apakah pesan memerlukan koneksi atau dapat tanpa koneksi, apakah paket dipastikan bebas dari kesalahan dan/atau menggunakan segala bentuk penanganan kesalahan, apakah paket dapat dimultipleks, apakah pengiriman ulang paket yang hilang atau hilang adalah tersedia, dan apakah kontrol aliran eksplisit diperlukan. TCP/IP mengabaikan kelas-kelas ini dan sebagai gantinya menawarkan dua jenis paket: paket TCP dan *paket User Datagram Protocol* (UDP).

Lapisan berikutnya adalah lapisan jaringan. Pada lapisan ini, kita akhirnya mulai mempertimbangkan kebutuhan pengiriman paket melalui jaringan, yaitu bagaimana paket ini akan dikirim melalui jaringan. Unit data di sini adalah paket, juga disebut sebagai datagram. Pada lapisan ini, pengalamatan jaringan dilakukan. Dalam TCP/IP, alamat jaringan adalah alamat IP (apakah IPv4 atau IPv6 atau keduanya), tetapi model OSI tidak menentukan metode pengalamatan tertentu. Sebaliknya, ada beberapa protokol berbeda yang dapat digunakan di sini untuk jaringan tertentu. Ini termasuk, misalnya, IPv4 dan IPv6, dan *juga Internet Control Message Protocol* (ICMP) dan *Internetwork Packet Exchange* (IPX), untuk beberapa nama. *Address Resolution Protocol* (ARP) memetakan alamat lapisan jaringan ke alamat lapisan data link dan menjembatani lapisan jaringan ini dengan lapisan bawah berikutnya.

Lapisan jaringan diperlukan untuk mendukung gaya transmisi connection-oriented dan connectionless. Router beroperasi pada lapisan ini untuk merutekan pesan dari satu lokasi ke lokasi lain. Ini adalah tanggung jawab router untuk mengambil paket yang masuk, memetakannya ke lapisan ini untuk mendapatkan alamat tujuan, mengidentifikasi rute terbaik untuk mengirim paket dalam perjalanannya ke tujuan, dan meneruskan pesan. Menariknya, karena pesan didekomposisi menjadi paket dan paket ditransmisikan secara individual, paket dari pesan yang sama dapat menemukan rute yang berbeda dari sumber ke perangkat tujuan. Meskipun lapisan ini menangani penerusan paket ke lokasi jaringan berikutnya, lapisan ini tidak menjamin pengiriman yang andal.

Seperti dibahas dalam Bagian sebelumnya, sakelar beroperasi pada alamat perangkat keras (kita akan melihatnya di lapisan berikutnya). Namun, ada switch layer 3 yang beroperasi pada layer jaringan, membuatnya identik dengan router. Lapisan terendah kedua dari model OSI adalah lapisan data link. Lapisan ini bertanggung jawab atas pengiriman yang andal antara

dua node jaringan fisik. Istilah andal di sini berarti bebas dari kesalahan dari sudut pandang bahwa jaringan itu sendiri berfungsi, sehingga paket yang ditempatkan di node sumber akan diterima di node tujuan. Tingkat penanganan kesalahan ini tidak termasuk kesalahan yang mungkin muncul selama transfer, yang mengakibatkan data salah (yang akan ditangani pada lapisan transport). Pada layer 2, konten yang dipindahkan disebut sebagai frame (bukan paket). Protokol lapisan 2 termasuk mode transfer asinkron (ATM), ARP (sebelumnya disebutkan), X.25, dan Point-to-Point Protocol (PPP). Baik ATM maupun X.25, protokol lama, dijelaskan dalam konten online yang menyertai bab ini.

Lapisan tautan data terdiri dari dua sublapisan, lapisan MAC dan lapisan *Logical Link Control* (LLC). Sublapisan MAC memelihara daftar semua perangkat yang terhubung pada tingkat jaringan lokal ini, menggunakan alamat perangkat keras masing-masing perangkat, yang dikenal sebagai alamat MAC. Ini adalah nilai 6-byte yang biasanya ditampilkan kepada pengguna dalam notasi heksadesimal seperti 01-E3-35-6F-9C-00. Alamat-alamat ini ditetapkan oleh manufaktur dan merupakan nilai unik (walaupun dapat juga dihasilkan secara acak oleh sistem operasi Anda). Nilai 6-byte (48 bit) memungkinkan untuk 248 kombinasi alamat yang berbeda, yang jumlahnya lebih besar dari 280 triliun.

Bingkai yang dimaksudkan untuk LAN (subnet) tertentu dikirim ke perangkat lapisan 2 (saklar). Alamat MAC tujuan frame digunakan untuk mengidentifikasi perangkat tertentu dalam jaringan/subnet lokal ini untuk mengirim frame. Tidak seperti router, yang meneruskan paket ke router lain (yaitu, ke jaringan atau subnetwork lain), switch pada dasarnya adalah titik akhir. Jadi, switch hanya perlu mengidentifikasi data layer 2 dari frame, dan oleh karena itu, frame tidak bergerak lebih jauh ke tumpukan protokol di switch.

Sublapisan LLC bertanggung jawab untuk multiplexing pesan dan menangani persyaratan pesan tertentu seperti kontrol aliran dan permintaan pengulangan otomatis karena kesalahan pengiriman/penerimaan. Multiplexing adalah ide menggabungkan beberapa pesan menjadi satu transmisi untuk berbagi media yang sama. Ini berguna dalam jaringan yang padat lalu lintas karena banyak pesan dapat dikirim tanpa penundaan. Merupakan tanggung jawab sublapisan LLC untuk menggabungkan pesan yang akan dikirim dan memisahkannya saat diterima. Lapisan terendah dari model OSI adalah lapisan fisik. Ini adalah lapisan media aktual tempat paket akan dikirim. Data pada lapisan ini terdiri dari bit-bit individual yang dikirim atau diterima. Bit harus direalisasikan dengan cara tertentu, tergantung pada medianya. Misalnya, tegangan ditempatkan pada kabel tembaga, sedangkan pulsa cahaya dikirim melalui kabel serat optik. Perangkat dapat terhubung ke media melalui konektor-T untuk jaringan bus atau melalui hub.

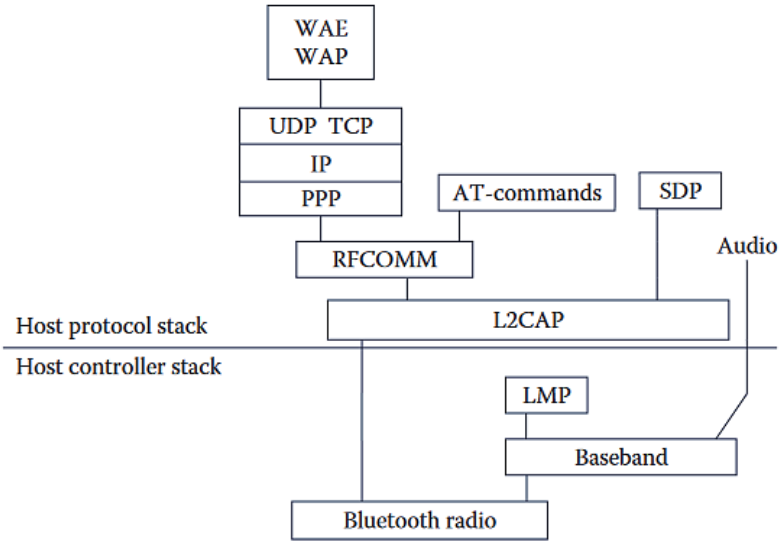
Lapisan fisik yang pada dasarnya sama dengan pengecualian mungkin jenis media yang tersedia (misalnya, Bluetooth menggunakan frekuensi radio). bukan media fisik untuk membawa sinyal). Saat menerima paket, model OSI menjelaskan bagaimana paket dibawa lapis demi lapis, hingga mencapai tingkat yang sesuai. Switch hanya akan melihat paket melalui lapisan 2, sedangkan router akan melihat paket melalui lapisan 3. Di ujung sumber daya tujuan, permintaan transmisi ulang dan kesalahan dapat naik ke lapisan 4 dan hilangnya sesi akan ditangani pada lapisan 5 ; jika tidak, paket digabungkan kembali pada layer 4 dan akhirnya dikirimkan ke beberapa aplikasi pada layer 7.

Tak ayal, nama Bluetooth sudah tidak asing lagi bagi Anda. Teknologi nirkabel ini, yang dimulai pada tahun 1994, merupakan kombinasi perangkat keras dan protokol untuk mengizinkan komunikasi suara dan data antar sumber daya. Tidak seperti protokol jaringan standar, dengan Bluetooth satu sumber daya adalah perangkat master yang berkomunikasi dengan perangkat pendukung. Pesan tunggal apa pun dapat dikirim ke hingga tujuh perangkat budak. Sebagai contoh, headset bebas genggam dapat mengirimkan pesan lisan Anda ke ponsel atau mobil Anda, sedangkan mouse nirkabel dapat mengirimkan data ke komputer Anda. Komunikasi Bluetooth dilakukan melalui sinyal radio di sekitar 2400 MHz tetapi terbatas pada jarak dekat hanya beberapa puluh kaki. Bandwidth Bluetooth diperkirakan hingga 3 megabit per detik, tergantung pada mode transmisi yang digunakan.

Tumpukan protokol Bluetooth dibagi menjadi dua bagian: tumpukan pengontrol host yang menangani masalah waktu dan tumpukan protokol host yang menangani data. Tumpukan protokol diimplementasikan dalam perangkat lunak, sedangkan tumpukan pengontrol diimplementasikan dalam firmware pada perangkat keras tertentu seperti mouse nirkabel. Kedua tumpukan ini ditunjukkan pada Gambar 2.2 Tumpukan teratas adalah tumpukan protokol host, yang terdiri dari banyak protokol berbeda. Yang paling signifikan adalah LLC dan Protokol Lapisan Adaptasi (L2CAP) ("2" menunjukkan dua L). Lapisan ini menerima paket baik dari komunikasi frekuensi radio (RFCOMM), *Telephony Control Protocol—Binary* (TCS BIN), atau lapisan *Service Discovery Protocol* (SDP) dan kemudian meneruskan paket ke *Link Manager Protocol* (LMP) atau antarmuka pengontrol host (HCI), jika tersedia. L2CAP harus dapat membedakan antara protokol-protokol lapisan atas ini, karena tumpukan pengontrol tidak. Protokol tingkat yang lebih tinggi (misalnya, lingkungan aplikasi nirkabel [WAE] dan AT-Commands) bergantung pada perangkat, sedangkan RFCOMM adalah protokol transport yang berhubungan dengan frekuensi radio. Tumpukan protokol pengontrol sebagian besar berbasis perangkat keras dan bukan minat khusus sehubungan dengan isi buku teks ini.

Paket Bluetooth memiliki ukuran default 672 byte tetapi dapat berkisar dari 48 byte hingga 64 kilobyte. Bidang paket menggunakan penyelarasan Little Endian (Little Endian dijelaskan di Lampiran B, terdapat di situs web buku teks Taylor & Francis Group/CRC Press). Karena ukuran pesan mungkin lebih besar dari ukuran paket maksimum, L2CAP harus memecah pesan yang lebih besar menjadi paket-paket dan memasang kembali serta mengurutkan paket-paket individu pada saat penerimaan. Untuk menjaga komunikasi dengan banyak perangkat, L2CAP menggunakan pengidentifikasi saluran (CID) untuk setiap perangkat yang berkomunikasi dengannya. Pengidentifikasi saluran ditetapkan secara unik untuk setiap perangkat jarak jauh. Mereka dapat digunakan kembali jika perangkat tidak lagi berkomunikasi dengan host (namun, beberapa CID dicadangkan). Setiap saluran diberi salah satu dari tiga mode berdasarkan lapisan atas protokol. Mode-mode ini adalah mode dasar, mode kontrol aliran, dan mode transmisi ulang.

Lapisan L2CAP juga menangani multiplexing dari beberapa paket data serta kehandalan. Penanganan error dilakukan dengan menggunakan metode *cyclic redundancy check* (CRC), paket yang dijatuhkan ditangani dengan meminta pengiriman ulang, dan timeout mengakibatkan paket dibilas dari lapisan ini.



Gambar 2.2 Susunan protokol Bluetooth

2.2 FRAME RELAY

Tujuan asli dari protokol Frame Relay adalah untuk mendukung infrastruktur *Integrated Services Digital Network* (ISDN). *Integrated Services Digital Networks* menyediakan komunikasi untuk suara digital, video, dan gabungan data, melalui jaringan telepon umum (jaringan circuit switched, berlawanan dengan packet switching dari sebagian besar jaringan komputer). Namun, dengan keberhasilan Frame Relay di ISDN, kemudian diperluas untuk menyediakan komunikasi suara dan data melalui jaringan *packet-switched*, baik area lokal maupun area luas. Faktanya, keuntungan dari Frame Relay adalah untuk menyediakan cara transmisi suara/video/data digital dari LAN ke titik akhir melalui WAN dengan biaya rendah. Lebih khusus lagi, LAN terhubung ke jaringan Frame Relay melalui saluran khusus. Baris ini berakhir pada switch Frame Relay, yang menghubungkan LAN ke LAN lain.

Komunikasi Frame Relay memecah pesan menjadi unit data berukuran variabel yang dikenal sebagai frame. Frame diketik berdasarkan jenis data yang dienkapsulasi (misalnya, video, suara, dan data). Pengguna akhir dapat memilih tingkat layanan yang memprioritaskan pentingnya bingkai berdasarkan jenisnya. Misalnya, jika data suara dianggap lebih signifikan, data tersebut akan memiliki prioritas lebih tinggi saat ditransmisikan melalui saluran yang menghubungkan LAN ke sakelar Frame Relay.

Seperti yang dirasakan bahwa data yang ditransmisikan menggunakan Frame Relay akan lebih sedikit rawan kesalahan daripada bentuk koneksi jaringan lainnya, Frame Relay tidak berusaha memperbaiki kesalahan, sehingga mengurangi jumlah data tambahan yang ditempatkan ke dalam bingkai. Titik akhir yang menerima bingkai yang salah hanya menjatuhkannya dan meminta agar dikirim ulang. Data video dan suara seringkali dapat bertahan dari beberapa bingkai yang terjatuh atau salah, tidak seperti data email atau halaman web. Di satu sisi, perbedaan ini mirip dengan apa yang kita lihat antara TCP dan UDP.

Mari kita lihat frame sebagai bagian dari data yang akan dikirim. Frame Frame Relay standar terdiri dari empat bidang berbeda: bendera, alamat, data, dan urutan pemeriksaan

bingkai (FCS). Bendera digunakan untuk sinkronisasi di tingkat tautan data jaringan. Semua frame akan dimulai dan diakhiri dengan pola bit 01111110. Untuk memastikan bahwa urutan bit ini unik di setiap frame, jika urutan bit ini muncul di bidang lain mana pun, itu harus diubah dengan menggunakan pendekatan disebut bit stuffing dan destuffing.

Bidang alamat adalah 16 bit. Ini terdiri dari pengidentifikasi koneksi tautan data (DLCI) 10-bit, alamat tambahan, bit kontrol, dan empat bit kontrol kemacetan. Bit DLCI menunjukkan sirkuit virtual sehingga frame ini dapat dimultipleks dengan frame lain dari pesan yang berbeda. DLCI dirancang dengan panjang 10 bit tetapi dapat diperpanjang melalui penggunaan bidang alamat yang diperluas untuk menunjukkan DLCI yang lebih panjang. Ini mungkin terjadi jika 10 bit tidak cukup untuk menyandikan semua sirkuit virtual (10 bit menyediakan 1024 pengidentifikasi berbeda). Berikutnya adalah bit perintah/respons (C/R). Bit ini saat ini tidak digunakan. Akhirnya, ada tiga bit yang didedikasikan untuk informasi kontrol kemacetan: pemberitahuan kemacetan maju-eksplisit (FECN), pemberitahuan kemacetan mundur-eksplisit (BECN), dan membuang kelayakan (DE) bit. FECN diatur ke 1 untuk menunjukkan ke perangkat akhir bahwa kemacetan terdeteksi dalam perjalanan. BECN diatur ke 1 untuk menunjukkan bahwa kemacetan terdeteksi kembali dari tujuan ke sumber. Alasan kedua bit ini adalah untuk membantu mengontrol kualitas pengiriman dengan menghindari atau mengurangi jumlah akumulasi data yang mungkin terjadi di dalam jaringan itu sendiri. Akhirnya, bit DE menunjukkan apakah frame yang diberikan kurang penting dan dengan demikian dapat dijatuhkan jika jaringan dipastikan akan padat.

Bidang data berisi data yang dikirimkan dan memiliki panjang variabel, hingga 4096 byte. Data dalam bidang ini ditempatkan di sini oleh lapisan yang lebih tinggi dalam protokol (misalnya, lapisan aplikasi). Terakhir, FCS berisi nilai CRC untuk seluruh frame, seperti yang dihitung oleh perangkat pengirim. Ini memberikan kemampuan untuk deteksi kesalahan oleh perangkat tujuan. Bidang ini panjangnya 2 byte.

Alternatif untuk format bingkai standar dikenal sebagai bingkai Antarmuka Manajemen Lokal (LMI). Kami tidak akan membahas detail bingkai LMI, tetapi bingkai ini lebih baik daripada bingkai standar dalam beberapa hal. Misalnya, bingkai LMI menyediakan alamat global dan informasi status untuk sinkronisasi bingkai dalam sirkuit virtual. Ini juga menyediakan bidang untuk jenis bingkai.

Frame Relay telah menjadi sangat populer tetapi semakin tidak didukung oleh ISP karena teknologi yang lebih murah seperti kabel serat optik mencapai lebih banyak titik akhir. Faktanya, Frame Relay sebagai topik telah dihapus dari ujian Cisco Certified Network Administrator (CCNA) yang populer.

2.3 ETHERNET

Ethernet adalah kombinasi perangkat keras dan protokol tingkat rendah. Ethernet memulai pengembangan di Xerox PARC pada tahun 1970-an tetapi dirilis oleh 3Com pada tahun 1980. Ethernet adalah keluarga teknologi yang digunakan untuk desain dan implementasi LAN. Meskipun berasal dari tahun 1980, ini masih merupakan bentuk LAN yang paling umum digunakan. Seiring waktu, model Ethernet telah ditingkatkan untuk

memanfaatkan teknologi yang lebih baru sehingga dapat ditingkatkan terkait aspek-aspek seperti bandwidth. Hari ini, kami bahkan menemukan Ethernet digunakan di MAN.

Ethernet sebagian besar berkaitan dengan hanya lapisan terendah dari jaringan, lapisan fisik dan lapisan data link. Pada lapisan fisik, Ethernet menentukan sejumlah variasi kabel dan pensinyalan yang berbeda. Ini termasuk penggunaan ketiga bentuk kabel yang umum: kabel twisted pair, kabel koaksial, dan kabel serat optik. Kabel Ethernet asli adalah kabel twisted-pair, menjaga jaringan Ethernet lebih murah daripada banyak pesaing, tetapi memiliki bandwidth atas terbatas 3 megabit per detik. Kemudian, dengan peningkatan yang dilakukan pada bentuk kabel ini, nama yang lebih baru telah disediakan dengan menggunakan peringkat bandwidth kabel: 10BASE-T, 100BASE-T, dan 1000BASE-T (10 untuk megabit per detik, 100 untuk 100 megabit per detik, dan 1000 masing-masing untuk 1000 megabit atau satu gigabit per detik). Hari ini, 10GBASE-T dan 100GBASE-T juga tersedia.

Selain kabel dan sumber daya komputer, jaringan Ethernet membutuhkan repeater untuk menangani degradasi sinyal. Kami menggunakan istilah pengulang di awal bab ini saat kami memperkenalkan hub. Tugas hub adalah menyampaikan pesan ke sumber daya lain. Dengan demikian, ia mengulangi pesan ke semua komponen yang terhubung dengannya. Dalam jaringan Ethernet, pengulangan diperlukan untuk meningkatkan sinyal agar dapat terus berjalan melintasi jaringan. Panjang maksimal kabel yang dapat digunakan tanpa repeater diperkirakan sekitar 300 kaki (kira-kira 100 meter). Repeater, juga dikenal sebagai extender Ethernet, akan memungkinkan jaringan Ethernet mencakup ukuran yang lebih besar. Saat ini, jarak maksimum antara repeater Ethernet diperkirakan sekitar 6000 kaki. Repeater awal hanya memiliki dua port sehingga repeater membentuk jaringan topologi bus. Namun, dengan munculnya hub sejati yang digunakan sebagai repeater, jaringan Ethernet dapat menggunakan topologi bintang.

Untuk terhubung ke jaringan, sumber daya memerlukan adaptor. Ada adaptor khusus yang dibuat untuk banyak mainframe dan komputer mini pada zaman itu. Pada tahun 1982, kartu adaptor tersedia untuk PC IBM.

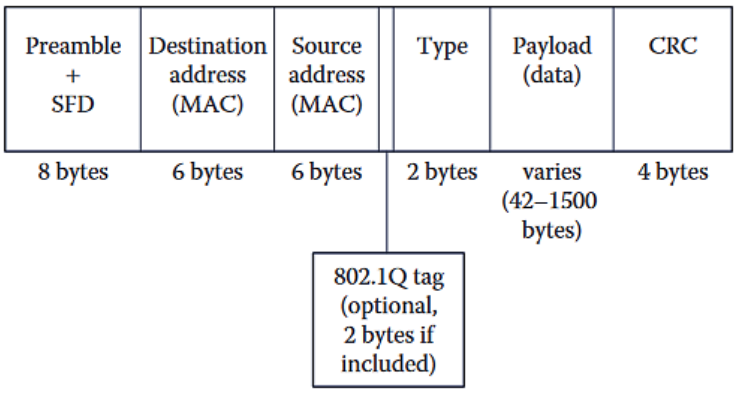
Karena jaringan Ethernet awal menjadi topologi bus, ada risiko beberapa pesan ditempatkan di jaringan pada saat yang bersamaan. Untuk menangani hal ini, Ethernet memperkenalkan CSMA/ CD, yang beroperasi pada lapisan data link. Setiap sumber daya yang mencoba untuk menempatkan pesan di jaringan pertama-tama akan mendengarkan jaringan untuk lalu lintas. Jika ada lalu lintas, sumber daya akan menunggu dan mencoba lagi pada slot waktu berikutnya, hingga jaringan tersedia. Namun, bagaimana jika sumber daya lain juga menunggu? Saat jaringan tersedia, jika semuanya mencoba menggunakan jaringan yang sekarang bebas, pesan mereka akan bertabrakan. Oleh karena itu, setiap sumber daya tidak hanya mendengarkan untuk memastikan jaringan bebas tetapi juga mendengarkan setelah pesan sumber daya ditempatkan di jaringan untuk kemungkinan tabrakan. Saat mendeteksi tabrakan, sumber daya mengirimkan pesan khusus yang disebut sinyal kemacetan. Sinyal ini menginformasikan semua sumber daya dari tabrakan. Sumber daya apa pun yang mencoba menggunakan jaringan sekarang mengetahui tentang tabrakan tersebut. Setiap sumber kemudian menunggu beberapa waktu secara acak sebelum mencoba lagi. Karena jumlah acak harus berbeda perangkat demi perangkat, upaya kedua sumber daya tidak boleh dilakukan

pada waktu yang sama dengan sumber daya lain yang terlibat dengan tabrakan yang sama. Sekali lagi, sumber daya mendengarkan jaringan terlebih dahulu, dan kemudian, jika atau saat bebas, ia mencoba lagi.

Perhatikan bahwa meskipun pengulang membantu memperluas ukuran fisik jaringan, semua perangkat dalam jaringan bus rentan terhadap tabrakan. Dengan demikian, semakin banyak sumber daya yang ditambahkan ke jaringan, semakin besar kemungkinan tabrakan. Ini membatasi jumlah sumber daya yang secara praktis dapat dihubungkan ke LAN. Perpindahan dari topologi bus ke topologi bintang, di mana titik pusatnya adalah sebuah saklar, dapat mengurangi masalah ini. Pada tahun 1989, sakelar Ethernet pertama tersedia untuk jaringan Ethernet.

Untuk menyelesaikan Ethernet, kami sekarang secara singkat fokus pada lapisan data link. Di lapisan ini, Ethernet membagi pesan jaringan menjadi paket-paket, juga dikenal sebagai bingkai. Deskripsi paket ditunjukkan pada Gambar 2.3. Di sini, kita melihat bahwa paket berisi pembukaan 56-bit. Bagian ini hanya terdiri dari bit 1 dan 0 bergantian, seperti pada 10101010, diulang untuk total 7 byte. Pembukaan digunakan untuk menyinkronkan paket yang masuk dari jaringan dengan jam komputer. Ini diikuti oleh pembatas frame awal 1-byte yang terdiri dari angka biner 171 (10101011). Perhatikan bagaimana urutan ini hampir identik dengan byte preamble, kecuali bahwa bit terakhir adalah 1. Ini memberitahu komputer penerima bahwa informasi paket akan segera dimulai.

Berikutnya dalam paket adalah header paket. Header berisi dua jenis informasi, alamat dan jenis paket. Alamatnya adalah alamat MAC untuk komputer tujuan dan komputer sumber. Bidang ini masing-masing 6 byte. Opsional, sebuah tag 802.1Q dari 2 byte diperbolehkan. Ini benar-benar digunakan jika jaringan adalah LAN virtual atau berisi komponen LAN virtual. Jika tidak, bidang ini dihilangkan. Bagian terakhir dari header adalah bidang 2-byte yang menentukan jenis bingkai, ukuran bagian data (muatan), dan, untuk muatan besar, protokol yang digunakan. Jenis-jenis frame adalah frame Ethernet II, frame Novell, frame LLC dan frame *Subnetwork Access Protocol.*



Gambar 2.3 format paket Ethernet.

Muatan mengikuti, yang berukuran setidaknya 42 byte tetapi bisa sebesar 1500 byte. Namun, paket khusus yang dikenal sebagai bingkai jumbo dapat memiliki sebanyak 9000 byte.

Mengikuti bagian data adalah bagian deteksi kesalahan yang dikenal sebagai FCS. Ini adalah segmen 4-byte yang berisi nilai CRC, seperti yang dihitung oleh perangkat pengirim. Perangkat penerima akan menggunakan nilai CRC untuk memastikan integritas data. Saat mendeteksi kesalahan, perangkat penerima menjatuhkan paket dan meminta transmisi ulang. Paket sekarang sudah lengkap, dan berikut ini akan menjadi celah antar paket. Namun, dalam beberapa kasus, lapisan fisik dapat menambahkan urutan end-of-frame-nya sendiri. Ethernet penting baik secara historis maupun dalam jaringan saat ini.

BAB 3
PENGANTAR INTERNET

Pada bagian ini, kita melihat pengantar di Internet. Buku ini berfokus pada apa yang membuat Internet berfungsi, pada tingkat yang dangkal. Kami menjelajahi tulang punggung Internet, router Internet, TCP/IP, dan DNS. Internet adalah jaringan fisik. Dengan sendirinya, itu tidak menyediakan layanan. Alih-alih, layanan disediakan melalui server yang menjalankan aplikasi dan protokol dikembangkan sehingga layanan dapat disediakan, apa pun jenis komputer atau jaringan yang Anda gunakan untuk berkomunikasi dengan server tersebut. Internet adalah mekanisme dimana komunikasi dilakukan dari satu lokasi ke lokasi lain, dari komputer peminta ke server, dan kembali ke komputer peminta. Jadi, kita perlu memahami bahwa Internet dikembangkan secara independen dari layanan yang kita gunakan melalui Internet.

3.1 APA ITU ALAMAT JARINGAN?

Seperti disebutkan sebelumnya, protokol jaringan akan melarang sarana pengalamatan. Model OSI menentukan dua lapisan alamat: alamat lapisan 2 dan alamat lapisan 3. Alasan perbedaan ini adalah bahwa alamat lapisan 2 hanya dapat dikenali di dalam jaringan tertentu berdasarkan tabel yang disimpan oleh sakelar jaringan. Sakelar mendapatkan alamat setiap kali perangkat baru terhubung ke jaringan lokal sakelar itu. Berbeda dengan itu, router perlu menggunakan bentuk alamat yang berbeda. Jadi, kami membedakan antara alamat lapisan 2, yang biasanya disebut alamat perangkat keras, dan alamat lapisan 3, yang akan kami sebut alamat jaringan.

Alamat perangkat keras juga dikenal sebagai alamat MAC, dan mereka ditugaskan ke antarmuka tertentu pada saat antarmuka dibuat (namun, sebagian besar alamat MAC perangkat modern dapat disesuaikan oleh sistem operasi). Sebagai gantinya, alamat lapisan 3 diberikan oleh perangkat saat terhubung ke jaringan. Ini dapat ditetapkan secara statis atau ditetapkan secara dinamis. Jika ditetapkan secara dinamis, alamat tetap berada di perangkat tersebut saat perangkat berada di jaringan. Jika perangkat dihapus dari jaringan atau dimatikan, alamat dapat dikembalikan ke jaringan untuk dibagikan ke perangkat lain.

Protokol yang paling terkenal, TCP/IP, menggunakan dua jenis alamat, IPv4 dan IPv6. Alamat IPv4 terdiri dari 32 bit yang dibagi menjadi empat oktet. Setiap oktet adalah angka 8-bit atau nilai desimal dari 0 hingga 255. Meskipun alamat IPv4 disimpan dalam biner di komputer kami, kami melihatnya sebagai empat nilai desimal, dengan titik yang memisahkan setiap oktet, seperti pada 1.2.3.4 atau 10.11. 51.201. Notasi ini sering disebut notasi desimal bertitik (DDN).

Dengan 32 bit untuk menyimpan alamat, ada kemungkinan 232 (4.294.967.296) kombinasi yang berbeda. Namun, tidak semua kombinasi digunakan (kita akan membahasnya di Bab 3), tetapi meskipun kita menggunakan semua kemungkinan kombinasi, 4 miliar alamat unik masih belum cukup untuk kebutuhan Internet saat ini. Hal ini sebagian karena kita telah

melihat perkembangan perangkat yang ditambahkan ke Internet, selain dari komputer pribadi dan mainframe: superkomputer, server, router, perangkat seluler, smart TV, dan konsol game. Karena setiap perangkat ini memerlukan alamat IP yang unik, kami telah mencapai titik di mana alamat IP telah habis. Salah satu cara untuk menghindari masalah kehabisan alamat IP adalah melalui terjemahan alamat jaringan (NAT). Namun, ke depan, strategi yang lebih baik adalah melalui IPv6. Alamat IPv6 panjangnya 128-bit dan menyediakan alamat yang cukup untuk 2128 perangkat. 2128 adalah angka yang sangat besar (tulis 3 diikuti dengan 38 nol dan Anda akan mendekati 2128). Diperkirakan akan ada cukup alamat IPv6 untuk menangani kebutuhan kita selama beberapa dekade atau abad (atau lebih). Satu kesamaan antara alamat IPv4 dan IPv6 adalah bahwa kedua bentuk alamat dibagi menjadi dua bagian: alamat jaringan dan alamat di jaringan (Anda mungkin menganggap ini sebagai kode pos untuk menunjukkan kota dan lokasi di dalam kota, dan alamat jalan untuk menunjukkan lokasi dalam wilayah setempat).

Ini adalah router yang beroperasi pada alamat IP. Ini menerima paket, memeriksa header paket untuk alamat tujuan paket, dan memutuskan bagaimana meneruskan paket ke lokasi jaringan lain. Setiap penerusan adalah upaya untuk mengirim paket lebih dekat ke tujuan. Ini dikenal sebagai jaringan packet-switched untuk membedakannya dari jaringan circuit-switched dari sistem telepon umum. Dalam jaringan circuit-switched, sebuah jalur telah dibuat sebelumnya sebelum koneksi dibuat, dan jalur tersebut dipertahankan hingga koneksi berakhir. Namun, dalam jaringan packet-switched, paket dikirim melalui jaringan berdasarkan keputusan dari router yang mereka jangkau. Dengan cara ini, paket pesan dapat menemukan jalur yang berbeda dari sumber ke tujuan. Dalam perjalanan, beberapa paket mungkin hilang (dijatuhkan) dan karena itu memerlukan transmisi ulang. Meskipun hal ini tampak seperti merugikan, fleksibilitas packet switching memungkinkan jaringan untuk terus beroperasi bahkan jika node (router atau sumber daya lainnya) menjadi tidak tersedia. Karena IPv6 adalah protokol yang lebih baru, tidak semua router mampu meneruskan paket IPv6, tetapi router yang lebih baru mampu melakukan ini.

Selain alamat MAC dan alamat IP, ada skema pengalamatan lain yang sedang atau telah digunakan. Dengan Bluetooth, hanya perangkat keras khusus yang dapat berkomunikasi dengan protokol ini. Perangkat ini, seperti NIC, dilengkapi dengan alamat perangkat keras saat diproduksi. Ini dikenal sebagai BD_ADDR. Perangkat master akan mendapatkan BD_ADDR pada saat perangkat pertama kali berkomunikasi dan menggunakan BD_ADDR selama komunikasi di masa mendatang.

X.25 (dijelaskan dalam bacaan online) menggunakan alamat 8 digit yang terdiri dari kode negara 3 digit dan nomor jaringan 1 digit, diikuti dengan nomor terminal hingga 10 digit. Model OSI melarang dua alamat yang berbeda: lapisan 2 (alamat perangkat keras) dan lapisan 3 (titik akses layanan jaringan [NSAP] alamat). Namun, model OSI tidak melarang implementasi khusus untuk alamat ini, dan karenanya, dalam praktiknya, kami melihat alamat MAC dan IP, masing-masing, digunakan. Ethernet dan Token Ring, karena mereka hanya mengimplementasikan sampai lapisan 2, hanya menggunakan pengalamatan perangkat keras.

Salah satu bentuk pengalamatan terakhir yang perlu diperhatikan adalah yang digunakan di Internetwork Packet Exchange (IPX). Ini adalah bagian dari rangkaian protokol

IPX/SPX yang digunakan dalam jaringan Novell NetWare dan ditemukan di banyak jaringan mesin berbasis Microsoft Windows yang menggunakan OS Windows (sampai Windows 95). Alamat IPX terdiri dari tiga bidang: alamat jaringan 4-byte, diikuti dengan nomor node 6-byte, diikuti dengan nomor soket 2-byte. Ini identik dengan alamat jaringan alamat IP, alamat mesin, dan alamat port, masing-masing (kita akan membahas port di Bab 3). Namun, tidak seperti alamat IP, di mana sebagian dari seluruh alamat didedikasikan untuk jaringan dan sebagian didedikasikan untuk alamat perangkat, di sini, kedua bagian tersebut dipisahkan secara jelas. Soket ditugaskan sebagai bagian dari protokol sehingga, misalnya, nomor soket 9091 (dalam heksadesimal) membawa paket TCP, 9092 membawa paket UDP, dan 0455 membawa data NetBIOS.

3.2 PENANGANAN KESALAHAN

Ada banyak bentuk penanganan kesalahan di komputer. Beberapa bentuk ada di dalam komputer untuk menangani masalah transfer data sederhana, seperti bit paritas. Bentuk lain digunakan untuk menyediakan redundansi data dalam penyimpanan (perangkat RAID). Untuk telekomunikasi, diperlukan suatu mekanisme yang tidak memakan waktu komputasi tetapi memberikan akurasi yang tinggi dalam mendeteksi kesalahan. Banyak bentuk protokol dapat menggunakan deteksi kesalahan sehingga setiap paket yang mengandung kesalahan dapat ditransmisikan ulang. Ini penting jika paket berisi data yang harus benar, seperti pesan email atau permintaan HTTP.

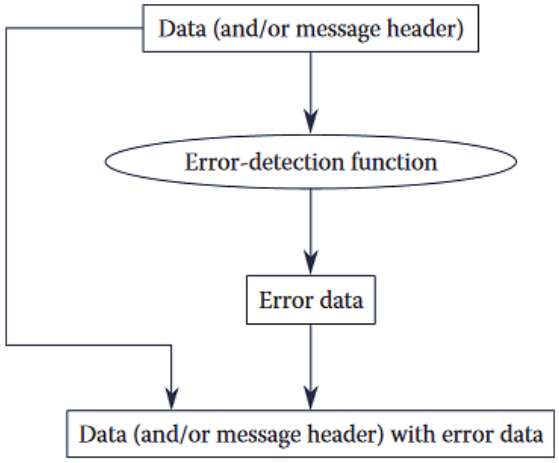
Ada banyak alasan mengapa paket yang ditransmisikan tiba dengan data yang salah. Alasan paling umum berkisar pada sifat media yang digunakan untuk mengirimkan data dan panjang media itu. Misalnya, kawat tembaga membawa data elektromagnetik. Hal ini dapat dipengaruhi (terganggu) oleh sejumlah benda eksternal yang berbeda, termasuk kondisi atmosfer (misalnya petir), semburan matahari, dan medan elektromagnetik yang kuat di sekitarnya. Kabel tembaga juga memiliki panjang yang sangat terbatas dimana sinyal elektronik dapat dibawa tanpa menggunakan beberapa bentuk repeater. Komunikasi nirkabel dapat diganggu oleh sinyal radio lain pada frekuensi yang sama serta oleh kondisi atmosfer. Namun, kabel serat optik pun tidak bebas dari kesalahan. Kesalahan selalu dapat muncul saat mentransmisikan data, terutama saat kecepatan transmisi mencapai miliaran bit per detik. Oleh karena itu, komunikasi jaringan memerlukan alat pendeteksi kesalahan. Ini ditangani dengan menghitung beberapa bentuk data deteksi kesalahan. Data ini kemudian dilampirkan ke pesan di suatu tempat (biasanya sebagai kolom tambahan di header pesan). Pengiriman pesan terdiri dari dua bagian: pesan dan informasi koreksi kesalahan. Biasanya, mekanisme diatur hanya untuk mendeteksi kesalahan. Kode koreksi kesalahan adalah konsep lain dan dapat digunakan untuk telekomunikasi, tetapi dalam praktiknya, kode tersebut tidak digunakan untuk telekomunikasi karena membutuhkan lebih banyak waktu untuk menghitung dan memerlukan lebih banyak bit, sehingga membuat pesan menjadi lebih lama.

Deteksi kesalahan ditangani di lapisan OSI 4. Pada lapisan ini, informasi deteksi kesalahan dihitung oleh komputer pengirim dan ditambahkan ke pesan. Pesan kemudian ditransmisikan. Penerima memecah pesan menjadi beberapa bagian, mengidentifikasi data deteksi kesalahan. Itu kemudian melakukan pemeriksaan pada pesan dengan menentukan

apakah pesan dan data deteksi kesalahan cocok. Jika penerima tidak menemukan kecocokan, ia mengetahui bahwa kesalahan telah terjadi di suatu tempat selama traversal pesan melalui jaringan dan meminta pesan untuk dikirim ulang. Jika data deteksi kesalahan cocok dengan pesan, asumsinya adalah tidak ada kesalahan yang muncul. Ini mungkin asumsi yang salah, seperti yang akan kita lihat ketika kita melihat checksum nanti, tetapi biasanya kesalahan akan menyebabkan data deteksi kesalahan yang salah.

Bentuk paling umum dari algoritma pendeteksi kesalahan yang digunakan dalam komunikasi data adalah checksum dan CRC. Meskipun keduanya adalah ide yang sama, mereka diimplementasikan dengan cara yang berbeda. Kami akan membahas keduanya, dimulai dengan yang lebih sederhana, checksum.

Asumsikan bahwa pesan terdiri dari n byte dan kami ingin checksum kami disimpan dalam 1 byte. Karena setiap byte data biner dapat mewakili bilangan desimal bilangan bulat positif (cukup konversi dari biner ke desimal), kami mengambil n byte dan menjumlahkannya. Kami membaginya dengan nilai 28 (256) dan mengambil sisanya. Sisa pembagian dihitung menggunakan operator modulo. Menghitung sisa sebenarnya dilakukan untuk kita setiap kali pembagian dilakukan (perangkat keras komputer menghitung hasil bagi dan sisanya pada waktu yang sama).



Gambar 3.1 Menambahkan data deteksi kesalahan ke pesan.

Mari kita perhatikan sebagai contoh bahwa pesan kita adalah kata "Halo" (tanpa tanda kutip). String dari lima karakter ini disimpan dalam ASCII dengan menggunakan 1 byte per karakter (ASCII sebenarnya mewakili karakter dalam 7 bit, dengan bit ke-8 sering tidak digunakan). Demikian pesan kami 01001000 01100101 01101100 01101100 01101111

Lima byte dalam desimal ini masing-masing adalah 72, 101, 108, 108, dan 111, dan jumlahnya adalah 500, atau bilangan biner 111110100. Perhatikan bahwa jumlah kita disimpan dalam 9 bit, bukan 8 bit. Sekarang, kita membaginya dengan 256 dan mengambil sisanya. Sisanya, bila dibagi dengan 256, akan selalu antara 0 dan 255. Dengan demikian, kami menjamin bahwa hasil checksum dapat disimpan dalam 1 byte. Dalam hal ini, 500/256 adalah 1, dengan sisa 244. Checksum kita adalah 244, atau 11110100. Perhatikan bahwa angka ini sama dengan jumlah 8 bit paling kanan.

Perhatikan bagaimana jumlah kita dan checksum kita terkait karena checksum kita sebenarnya hanya 8 bit paling kanan dari jumlah kita. Pertimbangkan dua angka lain, yang jika dibagi dengan 256, menghasilkan sisa yang sama: 756 (yaitu 1011110100) dan 1012 (yaitu 1111110100). Kedua angka ini serta 500 memberikan sisa 244 (11110100). Faktanya, 756 adalah 500 + 256, yang kita peroleh dengan menambahkan 10 di sebelah kiri bilangan biner untuk 244, dan 1012 adalah 500 + 512, yang kita peroleh dengan menambahkan bit 11 di sebelah kiri bilangan biner untuk 244 .Dalam ketiga kasus (500, 756, dan 1012), 8 bit paling kanan adalah sama. Dari sini kita dapat menduga bahwa dengan checksum 8-bit, angka yang terpisah tepat 256 satu sama lain akan memiliki checksum yang sama. Diberikan dua pesan, jika jumlahnya tepat 256 terpisah, maka mereka memiliki checksum yang sama.

Mengapa kita harus peduli bahwa dua kemungkinan pesan memiliki jumlah yang terpisah satu sama lain? Bayangkan pesan yang dikirimkan di atas tidak seperti yang ditunjukkan melainkan 00001000 00100101 00101100 00101100 01101111

Di sini, jumlahnya adalah 244, jadi, checksumnya adalah 244. Jika kami mengirim "Halo" dan menerima pesan di atas, itu masih terlihat benar berdasarkan checksum. Ini adalah pesan yang sangat berbeda, karena 1 terdepan dalam empat byte pertama dihilangkan, menghasilkan pesan ASCII "\b ,,o" (ruang belakang, pemisah unit, dua koma, dan satu-satunya karakter yang bertahan, "o"). Untuk alasan ini, kita mungkin ingin menggunakan checksum yang lebih besar, misalnya 16 bit. Checksum 16-bit memerlukan penambahan nilai 16-bit secara bersamaan, lalu membaginya dengan 65.356 (216). Untuk contoh ASCII kita, kita kemudian dapat menambahkan 0100100001100101 (nilai ASCII untuk “h” dan “e”) menjadi 0110110001101100 (nilai ASCII untuk “l” dan “l”) dan 0110111100000000 (nilai ASCII dari “o” diikuti dengan byte kosong).

Karena pembagian adalah proses yang memakan waktu, cara lain untuk melakukan perhitungan checksum adalah menjumlahkan byte (atau 16-bit, jika kita menggunakan checksum 16-bit) secara bersamaan. Kami kemudian mengambil barang bawaan yang dihasilkan oleh penjumlahan dan menambahkan barang bawaan itu ke penjumlahan. Akhirnya, kami mengambil komplemen satu dari jumlah ini. Untuk menghitung bilangan komplemen satu, kita membalik semua bit (setiap 1 menjadi 0 dan setiap 0 menjadi 1). Mari kita periksa ini dengan pesan yang sama "Halo" menggunakan checksum 8-bit. Seperti yang telah kita lihat, penjumlahan dari 5 byte di “Hello” adalah 111110100. Sebagai angka 8-bit, kita memiliki 11110100 dengan carry 1 (bit terdepan adalah 1). Menjumlahkan carry dan jumlahnya memberi kita 11110100 + 1 = 11110101. Pelengkap satu dari ini adalah 00001010. Checksum kita adalah 00001010 atau angka desimal 10 (jangan bingung dengan angka biner).

Kami dapat menguji checksum kami dengan menjumlahkan pesan asli dan checksum. Kami kemudian menambahkan carry ke jumlah kami. Terakhir, kita ambil nilai komplemen satu, dan hasilnya harus 0. Mari kita lihat apakah ini berhasil. Kita sudah tahu bahwa jumlah dari 5 byte tersebut adalah 111110100; untuk ini, kita tambahkan 00001010 untuk memberi kita 111111110. Carry di sini sekali lagi adalah 1 bit terdepan. Kami menambahkan 8 bit pertama dari jumlah kami dan membawa kami, dan kami mendapatkan 11111110 + 1 = 11111111. Mengambil komplemen satu dari ini memberi kami 00000000. Oleh karena itu, jika

kami menerima 5 byte dan checksum dengan benar, kami melihat bahwa ada tidak ada kesalahan.

Meskipun checksum adalah cara yang mudah dihitung dan cara yang kuat untuk mendeteksi kesalahan, metode CRC adalah cara yang lebih disukai untuk menangani deteksi kesalahan komunikasi jaringan. Ada beberapa alasan untuk ini. Alasan utama adalah bahwa checksum adalah pendekatan yang lebih tua dan kurang canggih sehingga lebih rentan terhadap masalah tidak mendeteksi kesalahan jika ada dalam pesan. Selain itu, checksum berguna saat mendeteksi kesalahan bit tunggal tetapi kurang berguna saat pesan berisi kesalahan banyak bit (yaitu, banyak bit diubah).

Pemeriksaan redundansi siklik menggunakan ide yang mirip dengan checksum yang menghasilkan hasil n-bit untuk ditempelkan ke pesan. Namun, perhitungannya berbeda. Alih-alih penjumlahan dan pembagian, kami berulang kali melakukan operasi OR (XOR) eksklusif pada bit pesan, bersama dengan nilai kunci yang ditentukan.

Sebelum kita memulai pemeriksaan ini, mari kita pahami tentang operasi XOR. XOR adalah operasi bit-wise yang dilakukan pada dua bit. Hasil XOR adalah 1 jika kedua bit berbeda (satu bit adalah 1 dan bit lainnya adalah 0) dan 0 jika kedua bit sama (kedua bit adalah 1 atau kedua bit adalah 0). Karena XOR adalah operasi sederhana untuk dilakukan perangkat keras komputer, serangkaian operasi XOR dapat dilakukan dengan sangat cepat.

Agar CRC berfungsi, pertama-tama kita harus menentukan kuncinya. Kuncinya adalah nilai n-bit, dimana nilai tersebut harus disepakati oleh pihak pengirim dan penerima. Kami akan berasumsi bahwa kunci dimulai dengan dan diakhiri dengan 1 bit, karena kunci seperti itu lebih mudah digunakan dan memberikan hasil yang lebih baik. Perhatikan bahwa meskipun pihak pengirim dan penerima tidak harus menyepakati kunci, kunci dimasukkan ke dalam protokol, seperti yang akan dijelaskan nanti. Algoritme bekerja sebagai berikut, dengan asumsi kunci n bit.

1. Pad pesan dengan n-1 0 bit di akhir (di sebelah kanan) dan hapus semua awalan 0 dari pesan, sehingga bit pertama pesan adalah 1.
2. Sejajarkan kunci di bawah pesan di ujung paling kiri (baik kunci maupun pesan dimulai dengan 1 bit, karena kami menghapus semua awalan 0).
3. Lakukan XOR antara n bit dari bagian pesan tersebut dan kunci n-bit. Perhatikan bahwa hasil XOR akan selalu dimulai dengan 0, karena bit pertama dari kunci dan bit pertama dari pesan keduanya adalah 1 (1 XOR 1 adalah 0).
4. Sejajarkan kembali kunci dengan 1 bit pertama dari hasil di langkah 3. Ulangi langkah 3 dan 4 hingga 1 bit awal pesan memiliki bit lebih sedikit daripada kunci.

Data deteksi kesalahan adalah bit yang tersisa. Mari kita lihat sebuah contoh. Kami akan kembali menggunakan pesan "Halo" bersama dengan kunci 8-bit yang nilainya 10000111. Untuk langkah 1, kami harus sedikit memodifikasi pesan dengan menghilangkan 0 di depan dan menambahkan 7 nol di akhir. Jadi, pesan kami berubah dari 01001000 01100101 01101100 01101100 01101111 ke 10010000110010101101100011011000110111100000000.

Sekarang, kami menyelaraskan kunci ke pesan, seperti yang dinyatakan dalam langkah 2, dan menerapkan langkah 3 hingga 4 berulang kali, hingga mencapai kondisi terminasi (hasil XOR memiliki bit lebih sedikit daripada kunci). Untuk melanjutkan contoh ini, kita

menggunakan notasi M untuk pesan, K untuk kunci, dan X untuk hasil operasi XOR antara M dan K. K disejajarkan di bawah pesan. Kita menghitung X dan kemudian menurunkan sisa M untuk membentuk M baru. Perhatikan bahwa setelah iterasi ke-5, harus ada nol di depan yang dibawa ke M (angka 0 ditunjukkan dengan garis bawah di bawah nilai iterasi ke-5 untuk M), tetapi kita tidak membawa nol di depan, jadi kita mulai dengan urutan bit berikutnya setelah nol (11011000).

```
M      10010000110010101101100011011000110111100000000
K      10000111
X      00010111
M         1011111001010110110001101100011011110000000
K         10000111
X         00111001
M           1110010101011011000110110001101111000000
K           10000111
X           01100010
M           1100010010110110001101100011011110000000
K           10000111
X           01000011
M           1000011101101100011011000110111100000000
K           10000111
X           00000000
M                  1101100011011000110111100000000
K                  10000111
X                  01011000
M                   1011000110110001101111000000
K                   10000111
X                   00110110
M                     1101101011000110111100000000
K                     10000111
X                     01011101
M                      1011101110001101111000000
K                      10000111
X                      00111100
M                        1111001000110111100000000
K                        10000111
X                        01110101
M                         1110101001101111000000
K                         10000111
X                         01101101
M                          1101101011011110000000
K                          10000111
X                          01011010
M                           1011010110111100000000
K                           10000111
X                           00110010
M                             1100101011110000000
K                             10000111
X                             01001101
M                              100110111110000000
K                              1000111
X                              0001010
M                                 110111110000000
```

```
K                    1000111
X                    0101000
M                     10100010000000
K                     10000111
X                     00100101
M                       100101000000
K                       1000111
X                       0001100
M                          110000000
K                          10000111
X                          01000111
M                           10001110
K                           10000111
X                           00001001
```

Hasil XOR terakhir kita, 00001001, kurang dari kunci kita, 10000111, jadi kita selesai. Pesan, ketika kunci CRC 8-bit diterapkan, memberi kita nilai 00001001. Kita akan menyebut nilai ini C. Kita membubuhkan C ke pesan sebelum mengirimkannya. Di sisi penerima, berdasarkan protokol yang digunakan, penerima mengetahui untuk menghapus nilai CRC, C atau 00001001, dari pesan. Penerima menghitung M + C untuk mendapatkan M2. Sekarang, penerima melewati proses yang sama seperti di atas menggunakan M2 dan K, bukan M dan K. Hasilnya adalah sisa 00000000. Jika demikian, tidak terjadi kesalahan. Jika sisanya selain 00000000, kesalahan muncul.

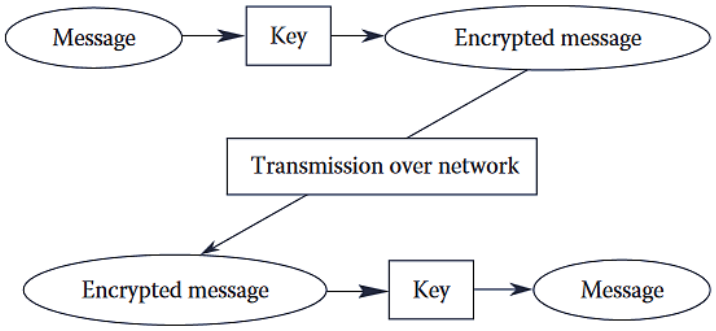
Dalam praktiknya, banyak bentuk telekomunikasi menggunakan CRC 16-bit atau 32-bit yang masing-masing dikenal sebagai CRC-16 dan CRC-32. Artinya, mereka menggunakan CRC dengan panjang kunci 16-bit atau 32-bit. Keduanya X.25 dan Bluetooth menggunakan versi CRC-16 yang dikenal sebagai CRC-16-CCITT, dengan kunci yang ditentukan sebagai 1000000100001. Ethernet menggunakan CRC-32, dengan kunci yang ditentukan sebagai 100110000010001110110110111. TCP/IP, mengirimkan TCP atau UDP paket, menggunakan checksum 16-bit.

3.3 TEKNOLOGI ENKRIPSI

Kami sekarang melihat sekilas enkripsi dan teknologi enkripsi. Alasan utama kami memerlukan enkripsi adalah karena pesan yang dikirim di Internet (dan melalui sebagian besar jaringan) secara default berupa teks biasa. Artinya, bit yang menyusun bagian data dari sebuah pesan hanyalah bit ASCII (atau Unicode). Ada kemungkinan, melalui sejumlah mekanisme yang berbeda, sebuah pesan dapat dicegat oleh pihak ketiga antara sumber dan tujuannya. Misalnya, sebuah router dapat diprogram untuk meneruskan pesan baik ke lompatan logis berikutnya di sepanjang jaringan maupun ke alamat IP tertentu. Alternatifnya, router jaringan dapat diprogram ulang sehingga alamat IP yang disimpannya diubah. Dikenal sebagai keracunan ARP (antara lain istilah), itu bisa memungkinkan seseorang untuk mengambil kendali ke mana router itu meneruskan pesannya. Namun kemungkinan lain adalah bahwa jaringan nirkabel dapat disadap jika seseorang berada di dekatnya dan dapat melihat lalu lintas pesan dari komputer seseorang ke titik aksesnya.

Ancaman bahwa pesan dapat dicegat melalui Internet mungkin tidak menjadi perhatian khusus jika bukan karena menggunakan Internet untuk mengakses informasi rahasia. Jika tidak aman, seseorang mungkin bisa mendapatkan kata sandi ke salah satu akun Anda dan, dengan kata sandi itu, masuk ke akun Anda untuk mentransfer uang, membelanjakan uang, atau melakukan sesuatu yang jahat kepada Anda. Alternatifnya, pesan tidak aman mungkin menyertakan kartu kredit atau nomor jaminan sosial, yang memungkinkan seseorang mencuri identitas Anda.

Untuk mengatasi masalah Internet yang tidak aman, kami beralih ke enkripsi data. Enkripsi adalah proses menerjemahkan pesan ke dalam bentuk kode dan dengan demikian mempersulit pihak ketiga untuk membaca pesan tersebut. Dekripsi adalah proses kebalikan dari mengambil pesan kode dan menerjemahkannya kembali ke bentuk aslinya (Gambar 1.14). Seperti yang akan kita lihat dalam diskusi berikut, kita mungkin memiliki satu kunci yang digunakan untuk enkripsi dan dekripsi atau dua kunci terpisah.



Gambar 3.2 Mengenkripsi dan mendekripsi pesan.

Enkripsi dan dekripsi telah ada selama orang perlu berbagi informasi rahasia dalam bentuk yang dikomunikasikan. Dalam perang abad ke-20, enkripsi dan dekripsi dilakukan dengan menggunakan kode pengganti. Dalam kode seperti itu, setiap huruf akan diganti dengan huruf lain. Misalnya, kode pengganti umum dikenal sebagai kode Caesar, di mana bilangan bulat digunakan untuk menunjukkan jarak dimana karakter harus diubah. Misalnya, kode dengan jarak 1 akan mengubah "halo" menjadi "ifmmp". Ini juga dikenal sebagai kode rotasi + 1. Kode yang sedikit lebih rumit adalah sandi Vigenere, di mana beberapa kode Caesar diterapkan, di mana jarak untuk setiap huruf ditunjukkan dengan kata kunci khusus. Tanpa mengetahui teknik penggantian yang digunakan (apakah rotasi, kata kunci, atau algoritma penggantian yang lebih kompleks), seseorang masih dapat memecahkan kode dengan mencoba semua kemungkinan kombinasi huruf. Ini dikenal sebagai pendekatan brute-force untuk dekripsi. Sayangnya, kode pengganti terlalu mudah dipecahkan oleh komputer, yang dapat mencoba jutaan kombinasi atau lebih dalam satu detik.

Namun, saat ini, kami menggunakan jenis teknologi enkripsi yang lebih canggih. Perhatikan bahwa pesan kita diubah menjadi pesan biner (karena kita mengirimkan informasi melalui komputer). Kami mengambil pesan dan memecahnya menjadi unit masing-masing n bit. Kami menggunakan n bilangan bulat. Untuk setiap 1 bit dalam pesan, kami menambahkan bilangan bulat yang sesuai. Misalnya, jika kita memiliki n = 8 dan angka kita adalah 1, 2, 5, 10,

21, 44, 90, dan 182, maka pesan 10110010 akan dienkripsi sebagai jumlah dari 1, 5, 10, dan 90 atau 106, dan jika pesan kita adalah 00011101, pesan kita akan dienkripsi sebagai jumlah dari 10, 21, 44, dan 182 atau 257. Kita akan memanggil nomor kita (1, 2, 5, 10, 21, dll.) kunci kita.

Saat menerima pesan sebagai urutan bilangan bulat satu per setiap 8 bit pesan, kami mengembalikan nilai aslinya. Bagaimana? Perhatikan bahwa kunci angka kita semakin bertambah. Artinya, setiap angka lebih besar dari jumlah semua angka sebelumnya yang disatukan. 5 lebih besar dari 1 + 2. 10 lebih besar dari 1 + 2 + 5. 21 lebih besar dari 1 + 2 + 5 + 10. Hal ini memungkinkan kita untuk mengidentifikasi bilangan yang menyusun bilangan yang kita terima (katakanlah 257) dengan mencari angka terbesar dalam kunci kita yang dapat dikurangi dari angka yang diterima. Pada 257, kita tahu pasti ada 182, karena 182 adalah bilangan terbesar yang kurang dari atau sama dengan 257, dan semua bilangan yang kurang dari 182, jika dijumlahkan, akan berjumlah kurang dari 182. Artinya, jika kita melakukan tidak menggunakan 182 sebagai bagian dari pesan kita, kita tidak dapat mendekati 257 karena 1 + 2 + 5 + 10 + 21 + 44 + 90 < 182. Jadi, kita kurangi 182 dari 257, menghasilkan 75. Angka terbesar berikutnya dalam kunci ≤ 75 adalah 44, jadi sekarang kita tahu bahwa 257 mencakup 182 dan 44, menyisakan 31. Angka berikutnya harus 21, menyisakan 10. Angka berikutnya harus 10. Oleh karena itu, 257 = 10 + 21 + 44 + 182, atau angka ke-4, ke-5, ke-6, dan ke-8 pada kunci, yang memberi kita 257, sehingga nilai 8-bit yang dikirimkan harus 00011101. Kita telah berhasil memulihkan pesan aslinya!

Pendekatan yang baru saja kami jelaskan ini dikenal sebagai enkripsi kunci privat atau enkripsi kunci simetris. Kuncinya hanya diketahui oleh satu orang, dan orang itu menggunakan kunci untuk mengenkripsi dan mendekripsi data. Ini bagus jika kita ingin mengenkripsi file kita di komputer kita untuk penyimpanan yang aman, karena dengan mengetahui kuncinya, kita dapat mendekripsi file kapan saja kita mau. Atau, jika kita berbagi kunci dengan seseorang, kita dapat mengenkripsi pesan, mengirimkannya melalui Internet, dan orang tersebut dapat menggunakan kunci yang sama untuk mendekripsinya. Namun, pendekatan ini tidak dapat bekerja untuk e-commerce. Kenapa ini? Perusahaan yang menjalankan situs web (katakanlah amazon.com) harus dapat mendekripsi pesan Anda, sehingga dapat memperoleh nomor kartu kredit Anda. Namun, Anda juga harus memiliki kunci agar dapat mengenkripsi pesan untuk memulai. Jika perusahaan berbagi kunci dengan Anda, apa yang dapat mencegah Anda mencegat pesan orang lain dan menggunakan kunci untuk mendekripsi komunikasi mereka dengan amazon.com?

Dengan kunci privat, kita juga bisa membuat kunci lain yang disebut kunci publik. Kunci publik tidak memiliki properti yang semakin bertambah seperti yang kita lihat dengan daftar di atas. Misalnya, kunci 8 dapat memiliki nilai 31, 6, 22, 58, 109, 4, 77, dan 21. Jika amazon.com memberi Anda kunci publik untuk mengenkripsi pesan, Anda tidak dapat menggunakan untuk dengan mudah mendekripsi pesan. Tentu saja, jika Anda menggunakan kunci ini, pesan 00011101 tidak akan dienkripsi menjadi 257 melainkan 192. Jika amazon.com menggunakan kunci privatnya untuk mendekripsi 192, ia tidak akan mendapatkan 00011101. Untuk mengatasi masalah ini, ada angka tambahan yang dihasilkan bersama dengan kunci publik, di

mana angka-angka ini digunakan untuk mengubah 192 menjadi 257. Amazon.com kemudian menerapkan kunci privatnya untuk mendekripsi pesan tersebut.

Bentuk enkripsi ini disebut enkripsi kunci publik (juga enkripsi kunci asimetris). Berikut adalah konsep-konsep secara ringkas.

- Seseorang membuat kunci pribadi yang dirahasiakan.
- Dari kunci privat, dia dapat membuat kunci publik dan berbagi kunci publik.
- Sebuah pesan diubah dari bentuk biner aslinya menjadi urutan angka dengan menggunakan kunci publik.
- Angka-angka ini ditransmisikan secara terbuka di seluruh jaringan.
- Saat menerima nomor, mereka diubah menjadi nomor yang berbeda.
- Nomor baru tersebut digunakan dengan kunci privat untuk mendekripsi pesan.

Untuk mendukung enkripsi kunci publik, kami memiliki infrastruktur kunci publik (PKI). PKI terdiri dari pedoman teknologi dan kebijakan untuk membuat dan mengelola kunci publik dan penyebarannya. Ini termasuk, misalnya, kemampuan untuk menghasilkan sertifikat digital (yang menyematkan kunci publik di dalamnya), menyatakan sertifikat tersebut sebagai asli, dan mengingatkan pengguna ketika sertifikat dicabut.

Anda mungkin bertanya-tanya mengapa seseorang tidak dapat menggunakan kunci publik untuk mendekripsi pesan aslinya. Misalnya, jika Anda tahu bahwa pesannya adalah 192, tidak bisakah Anda mengidentifikasi angka dari kunci publik yang berjumlah 192? Jawabannya iya, tapi tidak semudah jika angkanya semakin bertambah. Inilah alasannya. Dengan 257 dan kunci privat, kita tahu pasti ada 182 di dalamnya, karena ini adalah bilangan terbesar ≤ 257. Tanpa sifat penjumlahan yang semakin bertambah itu, melihat 192, dapatkah kita mengatakan bahwa 109 harus menjadi bagian dari bilangan?

Mari kita lihat apa yang terjadi jika kita menggunakan pendekatan dekripsi yang sama. Jadi, kita ambil 192 dan kurangi 109, hasilnya 83. Angka terbesar berikutnya adalah 77, jadi kita kurangi 77 dari 83, jadi 6. Kurangi 6, jadi 0. Jadi, 192 adalah 109 + 77 + 6, yang mana memberi kita pesan 01001010. Namun, ini bukan pesan aslinya (00011101).

Kami masih dapat mendekripsi pesan tersebut, tetapi tidak semudah yang kami kira. Katakanlah kita ingin mendekripsi 192. Kita tidak tahu angka apa yang harus dijumlahkan untuk menghasilkan 192, jadi kita mencoba semua kemungkinan. Kita mungkin mencoba 31 dengan sendirinya, 31 + 6, 31 + 22, 31 + 6 + 22, dan seterusnya. Dengan 8 angka, total ada 28 = 256 kombinasi untuk dicoba. Beberapa dari 256 kombinasi berbeda ini ditunjukkan pada Tabel 3.1. Untuk seseorang, ini mungkin memakan waktu beberapa menit. Untuk komputer, ini akan memakan waktu sepersekian detik.

Tabel 3.1 Beberapa kombinasi dari delapan nilai kunci publik

31 + 6	31 + 22	31 + 6 + 22	31 + 58	31 + 109	31 + 4
31 + 77	31 + 21	31 + 6 + 22	31 + 6 + 58	31 + 6 + 109	31 + 6 + 4
31 + 6 + 77	31 + 6 + 21	31 + 22 + 58	31 + 22 + 109	31 + 22 + 4	31 + 22 + 77
31 + 22 + 21	31 + 58 +109	31 + 58 + 4	31 + 58 + 77	31 + 58 + 21	31 + 109 + 4
31 + 109 + 77	31 + 109 + 21	31 + 4 + 77	31 + 4 + 22	31 + 77 + 21	…
31 + 6 + 22 + 58	31 + 6 + 22 + 109	31 + 6 + 22 + 4	…	31 + 109 + 77 + 21	31 + 4 + 77 + 21
6 + 22	…	31 + 6 + 22 + 58 + 109 + 4 + 77		31 + 6 + 22 + 58 + 109 + 4 77 + 21	

Agar enkripsi berfungsi, kita perlu memastikan bahwa orang tidak hanya dapat mendekripsi pesan kode jika mereka tidak memiliki kunci privat, tetapi juga bahwa komputer tidak dapat melakukannya dengan mudah dengan mencoba semua kombinasi angka. Karena itu kami ingin menggunakan kunci yang ukurannya lebih besar dari 8. Berapa ukuran yang baik untuk kunci enkripsi kami? Terus terang, semakin besar semakin baik. Jika kunci 8 dapat didekripsi dengan pendekatan kasar, seperti yang ditunjukkan di atas, dalam 256 percobaan atau kurang, bagaimana dengan kunci 10? 210 adalah 1024, yang masih cukup mudah untuk komputer. Bagaimana dengan kunci 20? Ada 220 kombinasi nilai kunci yang berbeda. Ini memberi kami nilai sedikit lebih dari 1 juta kombinasi. Karena komputer pribadi modern beroperasi dengan kecepatan miliaran instruksi per detik, ukuran kunci ini pun terlalu lemah. Kunci 30 terlalu lemah karena 230 hanya sekitar 1 miliar. Superkomputer beroperasi dengan kecepatan triliunan operasi per detik, membuat 240 dan bahkan 250 dan 260 terlalu lemah. Sebaliknya, kami lebih suka kunci yang ukurannya lebih besar, mungkin 128 atau bahkan 256.

Jadi, sekarang, kita melihat jenis teknologi enkripsi yang tersedia: kunci publik dan kunci privat, dengan opsi ukuran kunci. Sejumlah algoritme tersedia untuk kita, sebagian besar diimplementasikan dalam perangkat lunak sehingga kita sebagai pengguna tidak perlu khawatir tentang matematika di balik enkripsi dan dekripsi. Standar Enkripsi Data (DES), yang dikembangkan pada tahun 1970-an, adalah algoritma enkripsi kunci pribadi yang menggunakan ukuran kunci 56-bit; ini terlalu kecil untuk kebutuhan kita hari ini. Standar Enkripsi Lanjutan (AES) adalah algoritme tindak lanjut yang menggunakan ukuran kunci 128-bit sebagai gantinya. Triple DES adalah varian dari DES yang menggunakan tiga kunci terpisah masing-masing 56 bit. Meskipun setiap kunci hanya 56 bit, kombinasi penerapan tiga kunci membuatnya jauh lebih sulit untuk didekripsi dengan pendekatan brute-force. Selain algoritma enkripsi kunci pribadi ini, ada juga algoritma Message-Digest yang dikenal sebagai MDn, di mana n adalah angka, misalnya MD5. Ini adalah algoritme enkripsi 128-bit yang menerapkan fungsi hash (pembagian, menyimpan sisanya mirip dengan yang kami lakukan saat menghitung checksum). Bentuk enkripsi lain menggunakan kumpulan SHA: SHA-0, SHA-1, SHA-2, dan SHA-3. SHA adalah singkatan dari algoritma hash yang aman. SHA beroperasi mirip dengan MDn dengan mengubah urutan bit pesan menjadi nilai hash. SHA-0 tidak pernah digunakan dan SHA-1 ditemukan memiliki kelemahan keamanan, sehingga SHA-2 lebih sering digunakan saat keamanan sangat penting.

Kami selanjutnya dapat meningkatkan enkripsi kunci pribadi dengan memanfaatkan nilai nonce. Nilai ini adalah nomor acak (atau nomor acak semu) yang dihasilkan satu kali untuk komunikasi terenkripsi tunggal. Setelah pengirim mengirimkan pesannya, nilai nonce tidak digunakan lagi. Dengan nilai nonce, kode enkripsi menjadi lebih sulit dipecahkan karena kode khusus ini hanya digunakan satu kali, sehingga kumpulan data yang mungkin coba dianalisis oleh algoritme dekripsi tidak tersedia. Sebagai tindakan pencegahan keamanan lainnya, nilai nonce mungkin disertai stempel waktu sehingga hanya dapat diterapkan dalam jangka waktu yang wajar. Setelah batas waktu tersebut berlalu, nilai nonce tidak lagi dapat digunakan oleh klien yang mencoba mengenkripsi pesan.

Untuk enkripsi kunci publik, ada beberapa algoritma yang tersedia. Wired Equivalent Privacy (WEP) dirilis pada tahun 1999; itu menggunakan 26 nilai 10 digit sebagai kunci. Itu

telah digantikan oleh WPA dan WPA2. WPA adalah singkatan dari Wi-Fi Protected Access, dan seperti namanya, WPA digunakan untuk komunikasi nirkabel. WPA2 menggantikan WPA, yang seperti SHA-1 dan DES, ternyata memiliki kelemahan keamanan yang signifikan. Anda mungkin telah menggunakan WPA2 untuk mengamankan jaringan nirkabel rumah Anda sendiri. Sebuah fitur yang dikenal sebagai Wi-Fi Protected Setup masih memiliki kekurangan yang diketahui, jadi sebaiknya hindari penggunaan ini. Algoritme enkripsi kunci publik lainnya digunakan dalam SSH, program shell aman Unix/Linux. Algoritme spesifik yang digunakan dilambangkan sebagai SSH-1, SSH-2, atau OpenSSH. *Secure Sockets Layer* (SSL) dan *Transport Layer Security* (TLS) adalah sepasang protokol yang digabungkan untuk menyediakan enkripsi kunci publik dari dalam TCP/IP. Kami telah mencatat bahwa TCP/IP tidak memiliki sarana keamanannya sendiri, sehingga SSL/TLS telah ditambahkan. Kedua protokol ini beroperasi di lapisan transport TCP (lapisan kedua dari paling atas). SSL/TLS sebenarnya dapat menangani enkripsi kunci publik atau privat.

Terakhir, HTTP Secure (HTTPS) adalah variasi dari protokol HTTP yang digunakan untuk mendapatkan halaman web dari server web. Dengan HTTPS, kami menambahkan sertifikat digital, yang diberikan kepada pengguna akhir sebelum komunikasi aman dimulai. Sertifikat ini berisi kunci publik serta informasi untuk memastikan bahwa sertifikat tersebut milik situs web atau organisasi yang diklaim sebagai miliknya. Jika Anda terhubung ke server web menggunakan HTTPS dan Anda tidak mendapatkan kembali sertifikat, maka Anda mendapatkan sertifikat yang salah atau kedaluwarsa. Di sisi lain, jika Anda terhubung ke server web menggunakan HTTPS dan Anda menerima sertifikat yang tidak ditandatangani, Anda diperingatkan bahwa situs tersebut mungkin tidak dapat dipercaya. Jika Anda setuju untuk melanjutkan komunikasi Anda dengan situs, atau jika sertifikat situs dapat diterima, maka Anda memiliki kunci publik yang tersedia untuk mengenkripsi pesan Anda.

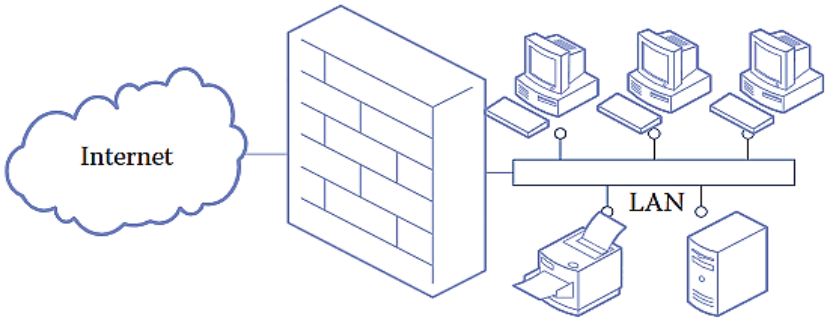
3.4 FIREWALL

Firewall, meskipun bukan komponen jaringan, sangat penting untuk keamanan sumber daya dalam jaringan. Firewall telah menjadi penting di dunia saat ini dengan ketergantungan kita pada Internet untuk e-commerce karena semakin banyak orang yang mencoba menyerang jaringan dan sumber dayanya baik sebagai bentuk perang dunia maya, tindakan kriminal, atau sekadar untuk melihat apa yang terjadi. mereka bisa melakukannya.

Firewall dapat berupa perangkat lunak atau perangkat keras. Sebagai perangkat lunak, itu adalah program atau layanan sistem operasi yang berjalan di komputer. Sebagai perangkat keras, seringkali merupakan perangkat jaringan yang ditempatkan di dekat atau di tepi jaringan. Router dan gateway, misalnya, dapat berfungsi sebagai firewall jika diprogram untuk melakukannya. Perhatikan bahwa sebagian besar firewall berorientasi perangkat keras akan melakukan fungsi jaringan lainnya, dan pada kenyataannya, firewall perangkat keras adalah perangkat lunak yang berjalan pada perangkat jaringan non-komputer.

Tugas firewall adalah menganalisis setiap pesan yang masuk dari jaringan atau dikirim ke jaringan untuk menentukan apakah pesan itu harus melewati firewall. Dengan cara ini, serangan dari luar dapat dicegah jika ada pesan yang terlihat mencurigakan. Pesan keluar dapat dicegah jika, misalnya, suatu organisasi tidak ingin jenis pesan tertentu dikirim (seperti

pesan yang dikirim ke domain tertentu seperti Facebook). Lihat Gambar 3.3 untuk ilustrasi firewall jaringan.



Gambar 3.3 LAN dengan firewall.

Firewall dianggap filter. Peran mereka adalah untuk memfilter pesan jaringan sehingga hanya pesan yang diizinkan yang diteruskan. Dengan kata lain, firewall menyaring semua pesan yang dianggap tidak perlu atau berpotensi berbahaya. Firewall akan terdiri dari aturan. Aturan pada intinya adalah pernyataan jika-maka. Aturan, misalnya, mungkin mengatakan bahwa pesan apa pun yang masuk ke Facebook akan diblokir atau pesan apa pun yang masuk melalui port yang belum dibuka akan diblokir. Aturan ini dapat menganalisis hampir semua aspek pesan. Berikut adalah daftar untuk mengilustrasikan beberapa aspek yang dapat diperiksa oleh aturan. Perhatikan bahwa aturan dapat menguji salah satu atau kombinasi dari atribut ini.

- Alamat sumber atau port (pesan masuk)
- Alamat atau port tujuan (pesan keluar)
- Jenis paket (UDP atau TCP)
- Protokol paket (misalnya, HTTP, FTP, dan DNS)
- Antarmuka yang menghubungkan perangkat ke jaringan (untuk komputer)
- Ukuran pesan
- Waktu hari/hari dalam seminggu
- Pengguna yang menginisiasi pesan (jika tersedia, ini memerlukan autentikasi)
- Status pesan (apakah ini pesan baru, sebagai tanggapan atas pesan lain, dll.)
- Jumlah pesan dari sumber ini (atau ke tujuan ini)

Menanggapi aturan yang dibuat, firewall dapat melakukan salah satu dari beberapa hal. Itu dapat memungkinkan pesan melewati firewall dan ke jaringan atau ke komputer. Itu bisa menolak pesan. Penolakan dapat mengakibatkan pengakuan dikirim kembali ke sumber sehingga sumber tidak melanjutkan pengiriman pesan (berpikir bahwa tujuan belum diakui karena saat ini tidak tersedia). Penolakan juga dapat mengakibatkan pesan dijatuhkan begitu saja tanpa mengirimkan pemberitahuan. Jika firewall adalah perangkat jaringan (misalnya, router), pesan dapat diteruskan atau ditolak. Akhirnya, firewall dapat mencatat kejadian tersebut.

Kami tidak membahas lebih detail di sini karena kami perlu lebih memahami TCP/IP terlebih dahulu. Kami akan mempertimbangkan bentuk keamanan lain saat kami menelusuri buku teks, termasuk menggunakan server proxy untuk memfilter halaman web yang masuk.

3.5 CACHE JARINGAN

Kata cache mengacu pada simpanan barang berharga terdekat. Kami menerapkan kata cache dalam banyak cara dalam komputasi. Cache yang paling umum digunakan adalah sejumlah kecil memori cepat yang ditempatkan pada unit pemrosesan pusat (CPU atau prosesor). CPU harus mengakses memori utama untuk mengambil instruksi program dan data saat menjalankan program. Memori utama (memori akses acak dinamis [DRAM]) jauh lebih lambat daripada CPU. Jika tidak ada bentuk memori yang lebih cepat, CPU harus terus-menerus berhenti untuk menunggu tanggapan DRAM.

Cache CPU, atau SRAM (RAM statis), lebih cepat dari DRAM dan hampir secepat atau sama cepatnya dengan kecepatan CPU. Dengan menempatkan cache pada prosesor, kita dapat menyimpan sebagian dari kode program dan data di sana dan tidak perlu mengakses DRAM kecuali apa yang dibutuhkan CPU tidak ada dalam cache. Namun, cache lebih mahal, jadi kami hanya menempatkan sedikit pada prosesor. Ini adalah trade-off ekonomis (serta trade-off ruang; prosesor tidak memberi kita cukup ruang untuk menempatkan cache yang sangat besar).

Faktanya, komputer seringkali memiliki beberapa cache SRAM. Pertama, setiap inti (dari CPU multicore) akan memiliki cache instruksi internal dan cache data internal sendiri. Kami menyebutnya sebagai cache L1. Selain itu, kedua cache mungkin juga memiliki cache kecil yang dikenal sebagai Translation Lookaside Buffer (TLB) untuk menyimpan informasi paging, yang digunakan untuk menangani memori virtual. Selain cache L1, sebuah prosesor mungkin memiliki cache L2 yang lebih besar dan terpadu. Komputer mungkin juga memiliki cache L3 yang lebih besar, baik di dalam chip maupun di luar chip di motherboard.

Kumpulan cache, DRAM, dan memori virtual (disimpan di hard disk) dikenal sebagai hirarki memori. CPU melihat cache L1 terlebih dahulu, dan jika item tersebut ditemukan, tidak diperlukan pengaksesan lebih lanjut. Jika tidak, CPU akan bekerja menuruni hierarki memori, hingga item yang dicari ditemukan: cache L2, cache L3, DRAM, dan memori virtual. Jika item ditemukan lebih rendah dalam hierarki, item tersebut disalin ke semua level yang lebih tinggi bersama dengan lokasi data yang berdekatan. Misalnya, item yang ditemukan di lokasi memori X akan dikelompokkan ke dalam blok dengan item X + 1, X + 2, dan X + 3. Saat X ditemukan, X akan dipindahkan ke tingkat hierarki memori yang lebih tinggi bersama dengan konten yang ditemukan di X + 1, X + 2, dan X + 3. Sayangnya, saat Anda menaikkan hierarki memori, lebih sedikit ruang yang tersedia, sehingga memindahkan sesuatu harus membuang sesuatu. Strategi penggantian cache harus digunakan untuk menangani situasi ini.

Selain strategi penggantian, kami juga memiliki masalah cache yang dikenal sebagai konsistensi cache. Jika datum bersama (yaitu, datum yang dapat digunakan oleh salah satu core kapan saja) telah dipindahkan ke cache L1 satu core dan kemudian dipindahkan ke cache L1 core lain, apa yang terjadi jika satu core memodifikasi datum? Nilai yang dimilikinya segar, sedangkan nilai inti lain di cache-nya kotor. Jika kami tidak memperbarui cache lain sebelum datum digunakan oleh inti lain, kami memiliki datum kedaluwarsa yang sedang digunakan. Penampilan datum ini di memori juga sudah ketinggalan zaman. Jika inti pertama membawa sesuatu yang baru ke cache L1-nya dan memilih untuk membuang entri yang berisi datum yang dimodifikasi ini, datum perlu disimpan ke memori. Kebijakan tulis cache menentukan

apakah penulisan ini akan dilakukan pada saat datum diperbarui dalam cache atau pada saat datum dibuang dari cache.

Apakah hierarki memori ada hubungannya dengan jaringan? Tidak terlalu! Namun, ada banyak cache yang memiliki tujuan serupa dengan cache L1, L2, dan L3. Artinya, dengan menyimpan data yang sering digunakan secara lokal, kita dapat mengurangi jumlah komunikasi jaringan yang diperlukan. Meskipun cache jaringan sering disimpan di hard disk, mereka memiliki dua kesamaan dengan cache perangkat keras L1/L2/L3: kebutuhan akan strategi penggantian dan masalah konsistensi cache. Cache jaringan biasanya tidak memerlukan kebijakan tulis, karena kami biasanya mendapatkan data dari server (sumber atau otoritas), sehingga komputer lokal kami tidak akan memodifikasi konten.

Kami melihat banyak bentuk cache jaringan. Peramban web kami memiliki cache sendiri, di mana konten halaman web yang baru dilihat disimpan. Cache browser web disimpan di hard disk komputer kita. Server proxy digunakan dalam suatu organisasi untuk menyimpan konten web yang baru diambil atau diakses untuk dibagikan di antara semua klien dalam jaringan. Saat kami mengunjungi sebuah situs web, browser web kami pertama-tama melihat cache browser lokal kami untuk melihat apakah konten disimpan di sana. Jika demikian, browser web memuat konten dari hard disk lokal dan tidak berkomunikasi dengan server web. Jika tidak, maka pesan yang dikirim dari browser web kami ke Internet akan disadap oleh server proxy kami (jika ada). Jika konten berada di sana, itu dikembalikan ke browser kami, dan permintaan kami tidak dilanjutkan lebih jauh. Mungkin juga terdapat cache yang terletak di Internet pada berbagai server dan/atau router.

Bagian lain dari data yang dapat di-cache adalah alamat IP untuk pemetaan alias IP (data resolusi alamat). DNS kami menangani pemetaan untuk kami. Kami menggunakan server nama DNS, yang didistribusikan di Internet. Namun, mengingat frekuensi komputer kami melakukan resolusi alamat, beberapa alamat/alias IP umum dapat di-cache di komputer lokal kami atau server lokal (server nama DNS organisasi kami) untuk menghemat waktu. Jenis cache ini mengalami masalah yang sama dengan cache browser web/server proxy.

Selain membutuhkan strategi pengganti untuk cache kita jika sudah penuh, konsistensi cache menjadi masalah. Bagaimana cache server lokal atau proxy kami mengetahui jika konten telah diperbarui?

Di bab-bab selanjutnya, kita akan meninjau kembali gagasan tentang cache DNS dan server proxy (perhatikan bahwa server proxy melakukan lebih dari sekadar konten cache). Kami juga akan melihat bagaimana kami dapat mengontrol berapa lama item tetap berada dalam cache melalui perangkat lunak server untuk server web, server proxy, dan server nama.

BAB 4
MEMBANGUN JARINGAN AREA LOKAL

Bab ini memberikan rincian lebih lanjut tentang beberapa tingkat yang lebih rendah dari tumpukan protokol jaringan. Itu dilakukan dengan fokus pada pembangunan jaringan area lokal (LAN), baik kabel maupun nirkabel.

4.1 PENDAHULUAN

Bentuk paling awal dari jaringan komputer dirancang dan digunakan oleh organisasi yang akan menggunakannya. Pada tahun 1950-an, perusahaan dengan beberapa komputer mainframe mungkin ingin komputer mainframe berkomunikasi satu sama lain. Pada tahun 1949, Angkatan Udara AS membangun MODEM pertama (singkatan dari MOdulation DEModulation), sehingga komputer mereka dapat berkomunikasi melalui saluran telepon. Bell Telephones mengembangkan MODEM mereka sendiri untuk menghubungkan koleksi komputer mereka, yang disebut SAGE, pada tahun 1953. Mereka kemudian mengkomersialkan MODEM mereka pada tahun 1958. MODEM memungkinkan komunikasi jarak jauh antar situs. Bagaimana dengan organisasi yang memiliki dua atau lebih mainframe di satu situs yang ingin mereka hubungkan secara langsung? MODEM, dengan kecepatannya yang lambat, bukanlah jawabannya. Sebaliknya, beberapa bentuk kabel langsung dapat dikirimkan komunikasi lebih cepat sementara tidak memerlukan penggunaan saluran telepon.

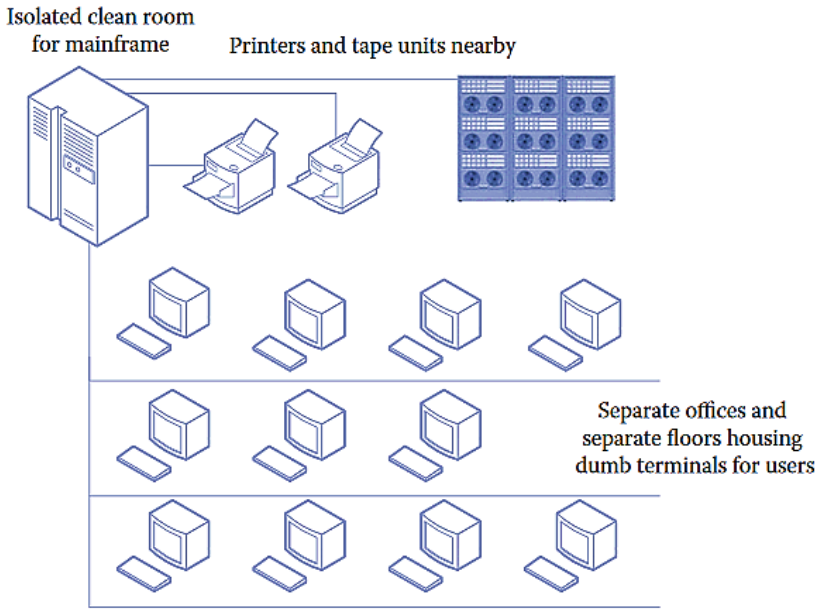
Pada awal 1960-an, sistem operasi komputer mainframe menggunakan salah satu atau kedua multiprogramming dan multitasking. Istilah-istilah ini menyatakan kemampuan komputer untuk menjalankan banyak program secara bersamaan. Ini berarti bahwa proses yang berjalan berada di memori dan prosesor komputer mati dengan cepat di antaranya. Prosesor hanya dapat menjalankan satu proses setiap saat, tetapi karena kecepatannya, mematikan di antara banyak proses memberikan ilusi menjalankan program secara bersamaan.

Dalam multiprogramming, peralihan antar proses terjadi hanya jika proses saat ini membutuhkan input atau output yang memakan waktu. Ini juga dikenal sebagai multitasking kooperatif. Dalam preemptive multi-tasking, peralihan antar proses terjadi ketika batas waktu yang telah ditentukan berlalu, sehingga prosesor dapat dibagi di antara banyak proses yang tumpang tindih. Multiprogramming mendahului multitasking preemptive, tetapi sistem multitasking apa pun mampu melakukan keduanya.

Dengan munculnya sistem operasi pemrosesan bersamaan, ada kebutuhan untuk menghubungkan banyak pengguna ke mainframe, daripada pengguna yang datang ke pusat komputer dan mengantri, menunggu untuk mendapatkan akses ke komputer. Pendekatan yang diambil adalah membekali setiap pengguna dengan perangkat input/output sederhana yang disebut terminal bisu. Inputnya melalui keyboard, dan outputnya melalui monitor, tetapi terminal bisu tidak memiliki memori atau prosesor. Sebagai gantinya, semua perintah akan dikirim dari terminal ke mainframe dan setiap tanggapan yang dikirim kembali ke terminal

akan ditampilkan. Hal ini menyebabkan cara desentralisasi untuk mengontrol mainframe, atau bentuk pertama dari LAN. Lihat Gambar 4.1.

Dalam beberapa kasus, koneksi antara dumb terminal dan mainframe (atau perangkat lain seperti line printer) akan menjadi koneksi point-to-point. Dengan koneksi point-to-point, ada sedikit kebutuhan untuk beberapa overhead jaringan yang dibutuhkan jaringan modern. Misalnya, jaringan point-to-point tidak memerlukan alamat sumber atau tujuan karena pesan apa pun dimaksudkan untuk ujung koneksi yang lain. Namun, pada kenyataannya, sebagian besar jaringan akan berisi banyak terminal bisu, bersama dengan perangkat penyimpanan dan printer. Hal ini akan membuat jaringan point-to-point antara setiap perangkat dan mainframe menjadi sangat mahal, karena mainframe memerlukan port untuk setiap koneksi. Masalah ini menjadi lebih parah jika organisasi memiliki banyak mainframe.



Gambar 4.1 Terminal dump terhubung ke satu mainframe.

LAN awal sangat eksklusif, masing-masing dibangun oleh organisasi yang menggunakannya. Organisasi tidak hanya perlu menemukan cara untuk menghubungkan sumber daya secara fisik bersama-sama, tetapi mereka juga harus menerapkan mekanisme yang diperlukan untuk pengalamatan, penanganan tabrakan, arbitrasi jaringan, penanganan kesalahan, dan sebagainya. Karena kerumitan jaringan yang dihasilkan, banyak organisasi mengembangkan jaringan mereka sebagai rangkaian lapisan, dengan aturan untuk memetakan pesan dari lapisan ke lapisan. Dengan demikian, protokol jaringan telah dibuat. Baru pada awal 1970-an upaya apa pun dilakukan untuk membakukan protokol jaringan ini. Teori packet-switching, diperkenalkan pada awal 1960-an, juga berkontribusi pada pengembangan jaringan komputer. Ditunjukkan dengan sukses dalam implementasi *Advanced Research Projects Agency Network* (ARPANET), diadopsi pada tahun 1970-an sebagai pendekatan yang akan digunakan oleh LAN yang diperkenalkan pada tahun 1970-an.

Dengan kumpulan teknologi yang sekarang tersedia untuk jaringan komputer, termasuk protokol, dengan standardisasi dan dengan keberhasilan pengalihan paket, perusahaan bersedia untuk mencoba memasarkan teknologi LAN mereka secara komersial. Perkembangan ini terjadi pada tahun 1970-an, dengan pesaing utamanya adalah Ethernet dan Token Ring, keduanya dirilis pada tahun 1980-an.

Faktor lain mempengaruhi kebutuhan dan kesuksesan LAN. Pada awal 1970-an, komputer pribadi pertama dijual. Pada awal hingga pertengahan 1980-an, penjualan komputer pribadi meroket, terutama karena organisasi dan bisnis bersedia membeli komputer pribadi daripada harus mengulur waktu di mainframe organisasi lain. Sekarang, organisasi-organisasi ini perlu menghubungkan komputer pribadi mereka dan sumber daya lainnya bersama-sama.

Bagian dari kesuksesan komputer pribadi berasal dari keputusan yang dibuat oleh IBM ketika mereka masuk ke pasar komputer pribadi pada awal 1980-an. Pada tahun 1981, IBM merilis PC IBM menggunakan arsitektur terbuka (komponen siap pakai dan arsitektur yang dipublikasikan). Keterbukaan IBM PC memungkinkan perusahaan lain untuk membangun komputer yang menggunakan komponen yang sama, sehingga perangkat lunak yang ditulis untuk IBM PC dapat berjalan di komputer mereka. Karena itu, PC IBM dan komputer yang kompatibel dengan PC memperoleh persentase pangsa pasar yang signifikan. IBM mendukung bentuk jaringan Token Ring pada awal 1980-an. Meskipun demikian, banyak vendor dan bisnis memilih Ethernet karena lebih murah dan lebih mudah untuk diukur.

Pada akhir 1980-an, Ethernet mulai mengungguli Token Ring sebagai teknologi pilihan untuk LAN. Pada 1990-an, dengan sedikit pengecualian, Ethernet telah memenangkan pertempuran LAN. Ini mulai berubah antara tahun 2000 dan 2005 saat teknologi nirkabel diperkenalkan. Hari ini, kita umumnya melihat dua bentuk LAN: LAN Ethernet kabel dan LAN nirkabel. Kami menjelajahi teknologi Ethernet di Bagian 4.2 dan teknologi nirkabel di Bagian 4.3 bab ini.

4.2 ETHERNET

Saat ini, hampir semua LAN kabel didasarkan pada teknologi Ethernet. Namun, Ethernet bukan hanya satu teknologi tetapi sejumlah teknologi berbeda yang telah dikembangkan selama bertahun-tahun. Kesamaan yang dimiliki semua teknologi Ethernet adalah bahwa semuanya didasarkan pada standar yang ditetapkan oleh Institute of Electrical and Electronics Engineers (IEEE) (teknologi Ethernet paling awal tidak diatur oleh standar IEEE), di bawah kategori 802.3. Selama bertahun-tahun, standar ini telah menentukan kecepatan transmisi yang diharapkan melalui kabel jaringan dan panjang kabel yang diharapkan berfungsi, serta menentukan batas yang dapat diterima untuk cross-talk (gangguan dua atau lebih komunikasi dilakukan melalui saluran komunikasi tunggal seperti kabel Ethernet). Ethernet sebagian besar berkaitan dengan hanya lapisan terendah yang didefinisikan dalam model *Open System Interconnection* (OSI): lapisan fisik dan lapisan data link.

Ethernet Pada Lapisan Fisik

Lapisan fisik dari setiap jaringan adalah media fisik dimana pesan dibawa. Untuk Ethernet, ini dilakukan dengan menggunakan beberapa bentuk kabel. Bentuk paling awal dari Ethernet secara ketat menggunakan kabel koaksial sebagai media bersama (yaitu, jaringan bus tunggal). Seiring berjalannya waktu, Ethernet memasukkan kawat bengkok ke dalam standarisasinya, memberi pengguna Ethernet fleksibilitas untuk menyediakan bandwidth yang lebih tinggi atau biaya yang lebih rendah (kami membandingkan keduanya nanti). Saat ini, kabel serat optik digunakan, sedangkan kabel koaksial tidak. Kabel Ethernet asli disebut sebagai ThickNet atau 10Base5. Bentuk kabel koaksial ini memiliki lapisan pelindung jalinan ekstra. Notasi 10Base5 memberi kita dua informasi berguna tentang kabel ini. Angka 10 menunjukkan kecepatan transmisi maksimum (10 megabit per detik) dan angka 5 menunjukkan panjang kabel maksimum, tanpa memerlukan pengulang (500 meter).

Pada tahun 1985, IEEE menyediakan spesifikasi untuk alternatif berbiaya lebih rendah, thinnet, juga disebut sebagai 10Base2. Bentuk kabel koaksial yang lebih tipis ini memiliki kecepatan transmisi maksimum yang sama tetapi panjang segmen maksimum yang lebih terbatas hanya 200 meter (pada kenyataannya, dalam praktiknya dirasakan bahwa segmen tidak boleh lebih dari 185 meter).

Institut Insinyur Listrik dan Elektronika juga menyediakan standar untuk kabel Ethernet menggunakan pasangan kawat bengkok. Ini dikenal sebagai 10Base-T, atau lebih umum sebagai kabel kategori 5 (yang merupakan pasangan terpilin yang dapat digunakan untuk jaringan apa pun, bukan khusus Ethernet). Sekali lagi, 10 pada namanya mengacu pada tingkat transmisi. T menunjuk pasangan bengkok daripada koaksial, alih-alih menentukan panjang maksimum. Dalam kasus 10Base-T, panjang maksimumnya adalah 100 meter. Dengan demikian, 10Base-T membatasi panjang segmen lebih dari kabel koaksial. Karena biaya 10Base-T yang lebih murah, ini menjadi lebih populer daripada kabel koaksial. Karena teknologi meningkat dan menawarkan kecepatan bit yang lebih besar, 10Base-T yang ditingkatkan, sedangkan 10Base2 dan 10Base5 menjadi usang. Peningkatan ini disebut sebagai 100Base-TX dan 1000Base-T. 100Base-TX dikenal sebagai Fast Ethernet karena merupakan peningkatan besar pertama pada kecepatan Ethernet. Di sini, sesuai namanya, kecepatan transmisi meningkat dari 10 menjadi 100 Mb per detik. X menunjukkan bahwa, sebenarnya, ada versi tambahan lainnya seperti 100Base-FX dan 100Base-SX (keduanya menggunakan kabel serat optik) dan 100Base-BX (melalui untaian serat optik tunggal). Dengan 1000Base-T, kabel Ethernet meningkatkan laju transmisi dengan faktor 10 lainnya dan sekarang disebut oleh beberapa orang sebagai Gigabit Ethernet. Sedangkan 10Base-T dan 100Base-T adalah kabel kategori 5, kabel Gigabit Ethernet dapat menggunakan kabel twisted-pair atau serat optik. Versi kawat bengkok disebut sebagai kabel kategori 6.

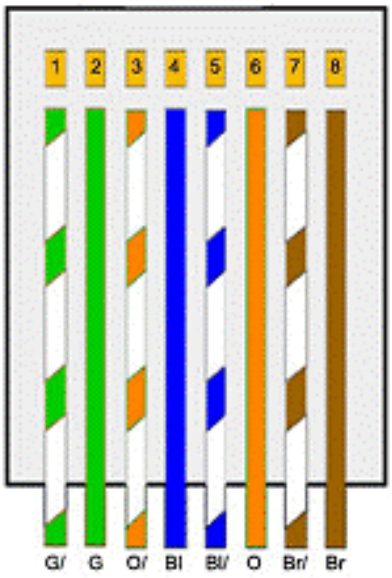
Tabel 4.1 Jenis Kabel Ethernet

Tipe Kabel	Kecepatan Transmisi	Tahun / Standart	Panjang Segmen Maksimum	Kegunaan
10Base-2	10 Mb/s	1985	Dibawah 200 meters	Usang
10Base-5	10 Mb/s	1980	500 meters	Usang

10Base-T	10 Mb/s	1985/IEEE 802.3i	100 meters	Usang
100Base-TX	100 Mb/s	1995/ IEEE 802.3z	100 meters	Masih digunakan ketika biaya merupakan faktor
100Base-SX, 100Base-FX	100 Mb/s	2000/ IEEE 802.3ac	275-500 meters	Masih digunakan
1000Base-T	1000 Mb/s	2000/IEEE 802.3ab	100 meters	Umum
1000Base-SX	1000 Mb/s	2000/IEEE 802.3ae	300-500+ meters	Umum
10GBaseCX4, -10GBase-SR dan lainnya	10 Gb/s	2002/ IEEE 802.3ae	100 meters	Umum
40GBase-T, 100GBase-KP4, dan lainnya	40Gb/s, 100 Gb/s	2010/ IEEE 802.3eba	100+ meters	Umum

Ada varian Gigabit Ethernet yang tidak menggunakan kabel tembaga, yang menggunakan nama seperti 1000Base-LX, 1000Base-SX, dan lainnya (kami tidak akan menyebutkan semuanya). Saat ini, ada versi kabel tembaga yang mencapai kecepatan hingga 10 Gb/dtk dan versi serat optik yang mencapai kecepatan 10 Gb/dtk, 40 Gb/dtk, dan bahkan 100 Gb/dtk.

Tabel 4.1 menyajikan perbandingan berbagai bentuk kabel ini. Ini menunjukkan perkiraan tahun di mana jenis itu distandarisasi, kecepatan dan panjang transmisi maksimum, dan apakah jenis kabel masih digunakan atau tidak. Perhatikan bahwa meskipun bentuk kabel berbasis serat pada Tabel 4.1 menunjukkan bahwa panjang segmen maksimum harus 100 hingga 500 meter, salah satu bentuk kabel serat optik yang dikenal sebagai kabel mode tunggal dapat mencapai panjang puluhan kilometer.



Keterangan
1. Putih Hijau
2. Hijau
3. Putih Orange
4. Biru
5. Putih Biru
6. Orange
7. Putih Coklat
8. Coklat

Gambar 4.2 Jack RJ45 (Versi Pertama).

Mari kita lihat lebih dekat kabel twisted-pair kategori 5 dan kategori 6. Kabel kategori 5 terdiri dari empat pasang bengkok, digabungkan dalam satu kabel. Empat pasang dalam

kabel kategori 5 diberi kode warna. Pasangan 1 memiliki satu kabel warna biru dan satu kabel warna putih/biru. Pasangan 2 terbelah, sehingga kedua kabelnya berada di kedua sisi pasangan 1 dan berwarna oranye dan putih/oranye. Pasangan 3 muncul di sebelah kiri pasangan 1 dan 2 dan berwarna hijau dan putih/hijau. Terakhir, pasangan 4 berada di sebelah kanan pasangan 1 dan 2 dan berwarna coklat dan putih/coklat. Delapan kabel diakhiri dengan jack RJ45. Perhatikan bahwa dalam varian lain, pasangan 3 mengelilingi pasangan 1 dan pasangan 2 berada di sebelah kiri pasangan 1 dan 3. Gambar 2.2 mengilustrasikan varian pertama dari jack RJ45. Dalam berkomunikasi melalui kategori 5 dan kategori 6, pasangan bengkok menggunakan kabel yang terhubung ke pin 1/2 dan 3/6, sedangkan kabel lainnya tidak digunakan. Variasi lain dari kabel kategori 5 disebut kabel kategori 5e; itu meningkat dari kategori 5 dengan mengurangi jumlah pembicaraan silang yang dapat terjadi di antara pasangan.

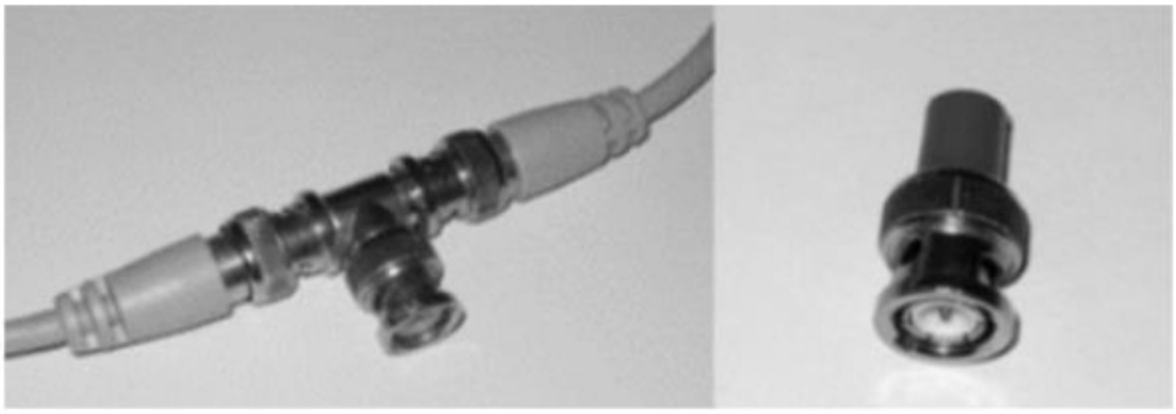
Mengenai kabel 10Base-T dan 100Base-T, ada dua format kabel berbeda yang dikenal sebagai kabel straight-through dan kabel crossover (jangan bingung dengan cross-talk yang disebutkan sebelumnya). Dengan kabel langsung, pin 1 dan 2 di satu ujung terhubung ke pin 1 dan 2 di ujung lainnya, sedangkan pin 3 dan 6 di satu ujung terhubung ke pin 3 dan 6 di ujung lainnya. Idenya adalah bahwa satu perangkat akan mengirimkan melalui 1 dan 2 dan menerima lebih dari 3 dan 6, sedangkan perangkat lain akan menerima lebih dari 1 dan 2 dan mengirimkan melalui 3 dan 6. Ini adalah kasus ketika komputer terhubung ke hub atau mengalihkan. Namun, ketika dua perangkat terhubung langsung bersama-sama, tanpa hub atau switch, maka kedua perangkat akan menggunakan pin 1 dan 2 untuk mengirim dan 3 dan 6 untuk menerima. Ini membutuhkan kabel crossover. Kami juga menggunakan pendekatan ini saat menghubungkan dua hub atau dua sakelar bersamaan, karena keduanya ingin mengirimkan lebih dari 3/6 dan menerima lebih dari 1/2. Kabel Ethernet gigabit yang lebih modern menggunakan keempat pasang bengkok untuk berkomunikasi, sehingga kabel crossover memetakan kabel 4 dan 5 ke 7 dan 8. Dengan kabel serat optik, dua serat digunakan berpasangan untuk mode dupleks penuh (dibahas dalam beberapa paragraf), terhubung ke perangkat yang disebut konektor dupleks, sehingga satu serat digunakan untuk mengirim dari satu ujung dan menerima di ujung lainnya dan serat kedua membalikkan ini. Selain itu, perangkat jaringan yang lebih baru dapat dihubungkan melalui kabel langsung dan menggunakan logika bawaan untuk mendeteksi apakah jenis perangkat yang berkomunikasi memerlukan kabel crossover, dan jika demikian, apakah kabel dapat dipetakan ulang secara dinamis.

Kabel kategori 6 menggunakan standar yang lebih baru yang memungkinkan lebih sedikit pembicaraan silang daripada kategori 5 dan 5e. Kabel kategori 6 juga menggunakan konektor yang berbeda; namun, kabel ditempatkan pada posisi yang sama, seperti yang ditunjukkan pada Gambar 4.2. Ada juga kategori 6e yang mirip dengan kategori 5e. Keempat jenis kabel ini merupakan bentuk unshielded twisted-wire pair (UTP). Pasangan kawat terpilin terlindung kurang umum digunakan untuk jaringan komputer karena biayanya yang lebih besar, sedangkan UTP dengan perlindungan lintas-bicara yang cukup sudah cukup.

Pada jaringan Ethernet awal, perangkat terhubung ke kabel dengan menggunakan konektor-T, seperti yang ditunjukkan di sisi kiri Gambar 4.3. Setiap perangkat ditambahkan ke

kabel jaringan tunggal. Ini menciptakan jaringan topologi bus. Saat perangkat ditambahkan, jaringan tumbuh. Setelah panjang tertentu, jaringan akan menjadi terlalu panjang untuk sebuah pesan berhasil melewati seluruh panjang kabel tembaga. Konektor-T bukanlah pengulang, dan oleh karena itu, seluruh panjang jaringan dibatasi, kecuali pengulang digunakan. Panjang jaringan ini dikenal sebagai segmen. Selain batasan panjang segmen, beberapa kabel juga hanya mengizinkan koneksi perangkat dalam jumlah terbatas. 10Base-2, misalnya, mengizinkan tidak lebih dari 30 perangkat per segmen, terlepas dari panjang segmen. Di akhir segmen, akan ada terminator, seperti yang ditunjukkan di sisi kanan Gambar 4.3, atau repeater.

Tugas pengulang Ethernet adalah menangani degradasi sinyal. Di Bab 1, kita melihat bahwa tugas hub adalah meneruskan pesan ke sumber daya lain di jaringan yang sama. Dengan demikian, ia mengulangi pesan ke semua komponen yang terhubung dengannya. Dalam jaringan Ethernet, pengulang (juga disebut sebagai ekstender Ethernet) diperlukan untuk meningkatkan sinyal, sehingga dapat terus berjalan melintasi jaringan.



Gambar 4.3 T-connector menghubungkan bus ke NIC dan terminator.

Panjang maksimal kabel yang dapat digunakan tanpa repeater diperkirakan 100 meter untuk 10Base-T dan hingga 500 meter untuk 10Base-5. Melalui penggunaan repeater, jaringan Ethernet dapat berkembang menjadi ukuran yang lebih besar. Saat ini, jarak maksimum antara repeater Ethernet diperkirakan sekitar 2000 meter.

LAN Ethernet gaya bus yang menggunakan repeater menerapkan aturan 5-4-3. Aturan ini menunjukkan bahwa jaringan tidak memiliki lebih dari lima segmen yang dihubungkan bersama dengan menggunakan tidak lebih dari empat pengulang (satu di antara setiap segmen). Selain itu, tidak boleh lebih dari tiga pengguna (perangkat) berkomunikasi setiap saat, masing-masing pada segmen terpisah. LAN Ethernet selanjutnya harus memiliki tidak lebih dari dua segmen jaringan, bergabung dalam topologi pohon, membentuk apa yang dikenal sebagai domain tabrakan tunggal. Beberapa menyebut ini sebagai aturan 5-4-3-2-1.

Meskipun repeater membantu memperluas ukuran fisik jaringan, semua perangkat dalam jaringan bus rentan terhadap benturan. Dengan demikian, semakin banyak sumber daya yang ditambahkan ke jaringan, semakin besar kemungkinan tabrakan. Ini membatasi jumlah sumber daya yang secara praktis dapat dihubungkan ke LAN Ethernet. Hub dipilih

sebagai repeater karena hub dapat menggabungkan beberapa jaringan bus individu secara bersamaan, misalnya membuat pohon, seperti yang disebutkan di paragraf sebelumnya. Namun, dimasukkannya saklar memungkinkan pergeseran dari topologi bus ke topologi bintang, yang selanjutnya mengurangi masalah tabrakan. Akibatnya, semua perangkat yang terhubung ke satu sakelar dapat berkomunikasi dengan sakelar itu secara bersamaan, tanpa khawatir terjadi benturan. Sakelar kemudian dapat berkomunikasi dengan sakelar (atau router) lain, tetapi hanya satu pesan yang dapat dikomunikasikan pada satu waktu di sepanjang jalur yang menghubungkannya. Sakelar, membentuk jaringan bintang, berarti bahwa tidak akan ada media bersama antara sumber daya yang terhubung ke sakelar melainkan koneksi point-to-point individual antara setiap perangkat dan sakelarnya.

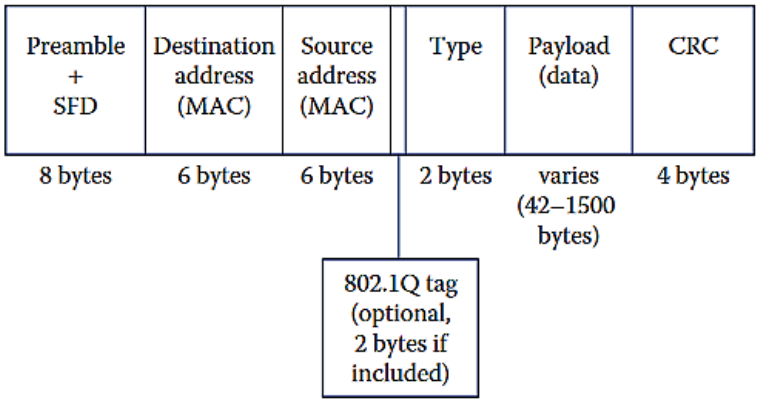
Sekarang, dengan tersedianya komunikasi point-to-point, jaringan juga dapat berpindah dari mode half-duplex ke mode full-duplex. Apa artinya ini? Dalam komunikasi, umumnya tiga mode dimungkinkan. Mode Simplex menyatakan bahwa dari dua perangkat, satu perangkat akan selalu mengirim dan yang lainnya akan selalu menerima. Ini mungkin terjadi, misalnya, ketika kita memiliki pemancar seperti mouse Bluetooth dan penerima seperti unit sistem komputer Anda atau fob kunci mobil dan mobil. Karena komputer dan sumber daya memerlukan komunikasi dua arah, mode simpleks tidak cocok untuk jaringan komputer.

Komunikasi dua arah ini dikenal dengan mode dupleks, dimana ada dua bentuk. Dalam mode half-duplex, sebuah perangkat dapat mengirimkan ke yang lain atau menerima pesan dari yang lain tetapi tidak pada waktu yang sama. Sebaliknya, satu perangkat adalah pemancar dan yang lainnya adalah penerima. Setelah pesan terkirim, kedua perangkat dapat membalikkan peran. Oleh karena itu, perangkat tidak dapat menjadi pemancar dan penerima sekaligus. Walkie-talkie akan menjadi contoh perangkat yang menggunakan mode setengah dupleks. Dalam mode full-duplex (atau hanya duplex), kedua belah pihak dapat mengirim dan menerima secara bersamaan, seperti telepon.

Implementasi topologi bus Ethernet mengharuskan perangkat hanya menggunakan mode setengah dupleks. Dengan topologi bintang, hub yang digunakan sebagai titik pusat bintang akan membatasi perangkat ke mode setengah dupleks. Namun, pada tahun 1989, sakelar Ethernet pertama tersedia untuk jaringan Ethernet. Dengan sakelar, Ethernet akan didasarkan pada topologi bintang, dan dalam hal ini, mode dupleks penuh dapat digunakan. Itu tidak berarti bahwa mode full-duplex akan digunakan.

Satu perubahan tambahan diperlukan saat berpindah dari satu bentuk teknologi Ethernet ke bentuk lainnya, apakah ini dari jenis kabel yang berbeda atau berpindah dari bus ke bintang. Perubahan ini dikenal sebagai autonegosiasi. Ketika perangkat memiliki kemampuan untuk berkomunikasi dengan kecepatan yang berbeda atau dengan mode dupleks yang berbeda, sebelum komunikasi dapat dimulai, kedua perangkat harus menetapkan kecepatan dan mode. Autonegotiation mensyaratkan hub/switch jaringan dan kartu antarmuka jaringan (NIC) komputer untuk dapat mengomunikasikan kemampuan mereka. Hub hanya dapat beroperasi dalam mode setengah dupleks, jadi hanya sakelar yang perlu menegosiasikan mode secara otomatis. Namun, hub dan switch berpotensi

berkomunikasi dengan kecepatan berbeda berdasarkan jenis kabel dan NIC di perangkat yang terhubung. Mungkin juga perlu negosiasi otomatis antara dua sakelar.



Gambar 4.4 format paket Ethernet.

Saat ini, kabel tipe 40GBase dan 100GBase menggunakan mode full-duplex, tidak memerlukan repeater, dan dicolokkan ke switch jaringan daripada hub. Selain itu, bentuk kabel ini tidak menggunakan CSMA/CD, karena diharapkan hanya digunakan dalam koneksi point-to-point antara komputer dan switch.

4.3 SPESIFIKASI LAPISAN ETHERNET DATA LINK

Kami sekarang fokus pada lapisan tautan data, seperti yang ditentukan oleh Ethernet. Di lapisan ini, Ethernet membagi pesan jaringan menjadi paket-paket (dikenal sebagai frame dalam terminologi Ethernet).

Bahwa bingkai Ethernet dimulai dengan pembukaan 8-byte yang terdiri dari bit 1 dan 0 bergantian yang diakhiri dengan 11 dengan total 8 byte. Pembukaan digunakan untuk menyinkronkan bingkai yang masuk dari jaringan dengan jam komputer. Dengan diakhiri dengan urutan 11, komputer akan mengetahui di mana sebagian besar paket dimulai.

Tiga bidang frame berikutnya menunjukkan header frame yang terdiri dari dua alamat kontrol akses media (MAC) (masing-masing 6 byte) dari perangkat tujuan dan perangkat sumber dan bidang tipe 2-byte. Bidang jenis menunjukkan jenis bingkai jika itu adalah bingkai non-default atau panjang bagian payload dalam byte, mulai dari 42 hingga 1500 byte. Nilai yang lebih besar dari 1536 menunjukkan jenis bingkai non-de-fault, yang dapat berupa bingkai Ethernet II, bingkai Novell, bingkai Protokol Akses Subjaringan, paket DECnet, atau bingkai Kontrol Tautan Logis. Antara alamat MAC dan kolom tipe adalah kolom 2-byte opsional. Ini, jika ditentukan, adalah tag 802.1Q yang digunakan untuk menunjukkan bahwa LAN adalah LAN virtual (VLAN) atau bahwa LAN berisi komponen VLAN. Jika LAN tidak memiliki komponen VLAN, bidang ini dihilangkan. Kami secara singkat menjelajahi VLAN dalam bacaan online yang menyertai bab ini di situs web Taylor & Francis Group/CRC.

Frame berlanjut dengan konten sebenarnya dari frame, data atau payload. Muatannya minimal 42 byte dan bisa sebesar 1500 byte; namun, bingkai khusus yang dikenal sebagai bingkai jumbo bisa sebesar 9000 byte. Mengikuti bingkai adalah cuplikan yang digunakan

untuk deteksi kesalahan. Ethernet merujuk ke bidang ini sebagai urutan pemeriksaan bingkai tetapi sebenarnya adalah nilai CRC 4-byte, yang dihitung oleh perangkat pengirim. Jika perangkat penerima mendeteksi kesalahan, frame dijatuhkan, dan perangkat penerima meminta agar itu ditransmisikan ulang. Mengikuti frame adalah celah interpacket minimal 96 bit. Seluruh frame Ethernet harus memiliki panjang minimal 64 byte, dan jika tidak demikian, padding bit ditambahkan ke dalamnya.

Meskipun kami memperkenalkan alamat MAC di Bab 1, mari kita lihat lebih dekat di sini. Ingat bahwa alamat MAC adalah nilai 48-bit, direpresentasikan sebagai 12 digit heksadesimal, ditentukan dua digit sekaligus, dengan setiap urutan dua digit sering dipisahkan oleh tanda hubung. Berikut ini menunjukkan tiga cara untuk melihat alamat MAC: dalam biner, dalam heksadesimal, dan dalam heksadesimal dengan tanda hubung. Ini semua mewakili alamat yang sama.

```
01001000 00101011 00111100 00101010 01100111 00001010
482B3C2A670A
48-2B-3C-2A-67-0A
```

Jelas, daftar dengan tanda penghubung adalah yang paling mudah dipahami. Dengan 48 bit, ada 248 atau hampir 300 triliun kombinasi alamat. Alamat MAC 48-bit tradisional didasarkan pada implementasi asli Ethernet Xerox PARC. Variasi format alamat MAC disebut Extended Unique Identifier (EUI)-48, di mana alamat dibagi menjadi dua bagian: nomor organisasi dan nomor perangkat, keduanya 24 bit. Setiap organisasi harus memiliki nomor uniknya sendiri, dan kemudian di dalam organisasi, setiap nomor perangkat harus unik. Perbedaannya kemudian adalah apakah pabrikan atau organisasi bertanggung jawab untuk menetapkan keunikan alamat. Pada perangkat keras saat ini, alamat MAC juga dapat diberikan melalui alamat yang dibuat secara acak oleh perangkat lunak (seperti sistem operasi komputer).

Standar yang lebih baru telah ditetapkan, yang dikenal sebagai EUI-64, di mana alamat MAC diperluas hingga 64 bit dengan memasukkan dua byte FF-FE di tengah alamat MAC 48-bit sebelumnya. Institute of Electrical and Electronics Engineers mendorong penerapan standar EUI-64 saat alamat MAC baru ditetapkan. Meskipun alamat MAC 48-bit digunakan dalam kartu Ethernet (NIC), mereka juga ditemukan di perangkat jaringan nirkabel, termasuk perangkat Bluetooth, ponsel pintar, dan tablet, serta perangkat jaringan kabel lainnya seperti Token Ring dan kartu jaringan ATM. Di sisi lain, EUI-64 digunakan di FireWire dan Internet Protocol versi 6 (IPv6).

Untuk menunjukkan apakah alamat tersebut unik secara universal (diberikan oleh pabrikan) atau unik secara organisasional, bit kedua hingga terakhir dari byte pertama alamat digunakan. A 0 untuk bit ini menunjukkan alamat yang dikelola secara universal (atau unik global), dan 1 menunjukkan alamat yang dikelola secara lokal. Alamat di atas (dimulai dengan 48) digunakan secara universal, karena 48 adalah 0100 1000, jadi bit kedua hingga terakhir adalah 0.

Selain alamat MAC untuk perangkat, dua jenis alamat khusus digunakan di Ethernet. Alamat pertama adalah FF-FF-FF-FF-FF-FF, yang merupakan alamat broadcast Ethernet. Jika bingkai Ethernet berisi alamat ini sebagai tujuannya, bingkai dikirim ke semua perangkat di jaringan. Saat menyebutkan siaran Ethernet, perlu dicatat bahwa sakelar di jaringan Ethernet

juga dapat menyiarkan pesan dalam keadaan lain. Jika sebuah frame tiba di sebuah switch yang alamat MAC tujuannya tidak diketahui (tidak disimpan dalam tabel alamatnya), maka switch akan menyiarkan pesan ini ke semua perangkat yang terhubung dengannya dengan harapan seseorang akan mengenali alamat tersebut. Ini dikenal sebagai banjir. Alamat khusus lain yang digunakan dalam Ethernet adalah alamat multicast. Berbeda dengan alamat broadcast, alamat multicast mengambil bentuk yang berbeda. Bit terakhir dari byte pertama digunakan untuk menunjukkan apakah alamat mewakili lokasi unicast atau multicast. Jika alamatnya adalah unicast, maka alamat tersebut hanya ditetapkan ke satu perangkat. Jika itu adalah alamat multicast, meskipun alamatnya masih unik, itu tidak ditetapkan secara unik. Sebagai gantinya, itu dapat ditugaskan ke beberapa perangkat. Saat menerima bingkai dengan alamat MAC yang byte pertamanya diakhiri dengan 1 bit, bingkai diteruskan ke semua perangkat yang berbagi alamat multicast yang sama. Misalnya, alamat 01:80:C2:00:00:0E adalah alamat multicast (byte pertama adalah 00000001). Alamat khusus ini sebenarnya dicadangkan untuk Precision Time Protocol (PTP).

Switch mempelajari alamat MAC perangkat yang berkomunikasi dengannya. Sakelar semacam itu disebut sebagai sakelar sadar lapisan 2. Sakelar akan dimulai dengan tabel alamat MAC kosong. Saat perangkat berkomunikasi melalui sakelar, sakelar mengingat alamat MAC perangkat dan port yang terhubung dengannya. Ini adalah alamat MAC sumber dari frame yang masuk.

Pertimbangkan kasus di mana perangkat A mengirim ke perangkat B dan tidak ada alamat MAC perangkat yang saat ini tercatat di tabel sakelar. Saat menerima pesan dari A, saklar merekam alamat MAC A. Namun, ia belum mendengar kabar dari B, jadi bagaimana ia tahu port mana yang akan digunakan untuk meneruskan pesan? Itu tidak. Dalam hal ini, sakelar membanjiri semua perangkat dengan pesan tersebut. Perangkat B, saat menerima pesan dan mengetahui bahwa pesan itu ditujukan untuk dirinya sendiri, mengirimkan tanggapan kembali ke perangkat A. Tanggapan ini akan menyertakan alamat MAC B. Sekarang, switch dapat mengubah tabel alamat MAC sehingga memiliki entri yang diperlukan untuk perangkat B.

Sakelar akan mempertahankan tabel alamat MAC hingga sakelar diperintahkan untuk membersihkan mejanya atau hingga perangkat tidak berkomunikasi dengan sakelar selama beberapa waktu. Dalam keadaan terakhir ini, sakelar mengasumsikan bahwa perangkat tidak lagi terpasang. Sakelar juga dapat menghapus entri yang dipilih jika tabel menjadi penuh. Dalam kasus seperti itu, sakelar menggunakan strategi yang dikenal sebagai yang paling jarang digunakan, untuk memutuskan entri atau entri mana yang akan dihapus. Entri yang terakhir digunakan adalah entri yang paling lama tidak digunakan, sebagaimana ditentukan dengan menempatkan stempel waktu yang diperbarui di tabel sakelar untuk perangkat tempat pesan diterima.

Penggunaan Ethernet tidak terbatas pada LAN sumber daya komputer. Saat ini, teknologi Ethernet digunakan untuk membangun jaringan regional yang lebih besar dari LAN. Apa yang disebut Metro Ethernet adalah salah satu di mana berbagai bentuk kabel digunakan untuk menghubungkan berbagai komponen jaringan area metropolitan (MAN) bersama-sama. Dalam banyak kasus, MAN dibangun dengan menghubungkan LAN menggunakan

Synchronous Optical Networking (SONET) dan *Synchronous Digital Hierarchy* (SDH). Penggunaan teknologi Ethernet untuk membangun MAN, karena sebagian besar LAN individual sudah diimplementasikan menggunakan Ethernet, lebih murah daripada penggunaan SONET sementara, dengan penggunaan kabel berbasis serat optik, MAN masih dapat bersaing dengan SONET dalam hal bandwidth. Juga lebih mudah untuk menghubungkan LAN individual ke MAN berbasis Ethernet daripada ke MAN berbasis SONET. Kelemahan utama dari MAN berbasis Ethernet kurang reliabilitas dan skalabilitas terbatas. Oleh karena itu, pilihan lain adalah menggunakan model hybrid seperti Ethernet melalui SONET, dalam hal ini protokol Ethernet digunakan di seluruh MAN, sedangkan koneksi fisik antar LAN dibuat oleh teknologi SONET. Kami tidak akan membahas SONET lebih detail.

Membangun Jaringan Wilayah Lokal Ethernet

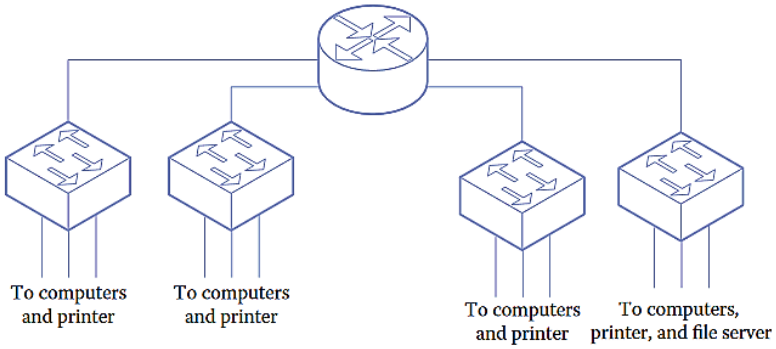
Sekarang setelah kita memahami teknologi Ethernet, mari kita pertimbangkan apa yang diperlukan untuk merancang dan mengimplementasikan jaringan kabel berbasis Ethernet. Pembahasan ini akan dibatasi hanya pada Ethernet dan oleh karena itu pada dua lapisan terendah dari OSI. Masalah lain seperti alamat jaringan (misalnya alamat IP), keamanan, dan sebagainya tidak akan dibahas di sini. Kami akan melihat langkah-langkah keamanan Ethernet nanti di bab ini.

Langkah pertama kami adalah mengidentifikasi kebutuhan organisasi kami (atau rumah tangga) dan menerjemahkan kebutuhan tersebut ke dalam sumber daya fisik yang akan kami butuhkan. Mari kita asumsikan bahwa kita sedang mengembangkan jaringan untuk bisnis menengah dengan 40 karyawan, yang masing-masing membutuhkan komputer desktop sendiri. Karyawan perlu mencetak materi, dan kami memutuskan bahwa empat printer sudah cukup. Kami juga memutuskan bahwa kami akan menggunakan server file terpusat untuk dokumen bersama. Kami akan menggunakan server file tunggal untuk tujuan ini. Kita perlu menghubungkan semua sumber daya ini bersama-sama ke dalam jaringan dan mengizinkan akses Internet. Printer akan ditempatkan secara terpusat dan dibagi di antara 10 karyawan, sedangkan server file dapat ditempatkan di lokasi lain, mungkin di ruangannya sendiri. 40 karyawan menempati 2 lantai sebuah gedung, dengan kantor yang terletak setiap 20 kaki (kurang-lebih).

Sekarang setelah kita mengetahui sumber daya kita, kita dapat melihat standar Ethernet untuk menentukan perangkat keras pendukung jaringan apa yang kita perlukan. Organisasi memutuskan untuk efektivitas biaya untuk menggunakan 10GBase-T, yaitu kabel pasangan kabel berpilin 10 Gb. Karena pemilihan kabel tembaga, jarak antar sumber daya berperan. Kami dapat menempatkan sumber daya hingga jarak 100 meter, tanpa harus menggunakan pengulang.

Kami akan menghubungkan sumber daya kami bersama dengan sakelar jaringan, membentuk topologi bintang. Sebuah switch jaringan tipikal mungkin memiliki 16 port yang tersedia. Diberikan sakelar 16 port, kita dapat menyambungkannya dengan 16 perangkat. Kami akan menempatkan sakelar kami secara terpusat di lantai antara 10 kantor. Artinya, kami akan menghubungkan 10 komputer karyawan ke sebuah saklar. Kami juga akan menghubungkan printer ke setiap sakelar dan server file ke salah satu dari empat sakelar. Dengan 40 komputer, 4 printer, dan server file, kita membutuhkan 4 sakelar. Jumlah

perangkat yang terhubung ke setiap sakelar akan menjadi 11 (10 komputer plus printer) atau 12 (perangkat tambahan adalah server file). Untuk menghubungkan sakelar bersama, kami menggunakan router. Pengaturan ini ditunjukkan pada Gambar 4.5



Gambar 4.5 Contoh pengaturan LAN dari empat LAN, dengan sakelar yang terhubung ke satu router.

Dengan kantor karyawan berjarak 20 kaki, jika ada 11 item dengan jarak yang sama yang terhubung ke sakelar, sakelar dapat diposisikan tidak lebih dari sekitar 220 kaki dari perangkat terjauh (yaitu, dalam jarak 75 meter). Faktanya, jika sakelar terletak di pusat di antara 10 komputer, jarak terjauh sumber daya apa pun dari sakelar harus sekitar setengahnya atau, mungkin, 40 meter. Oleh karena itu, kami dapat menggunakan kabel 10GBase-T tanpa repeater.

Saat karyawan bekerja di dua lantai, dua sakelar berada di satu lantai dan dua sakelar lainnya berada di lantai lainnya. Kami akan memposisikan router berdekatan dengan salah satu dari empat sakelar. Untuk menghubungkan switch ke router, kita mungkin membutuhkan lebih dari 100 meter kabel, tergantung bagaimana kita menata kabelnya. Jika kita membuat lubang di lantai/langit-langit, kita mungkin bisa menghubungkannya dengan kabel kurang dari 100 meter. Jika mereka harus berjalan di sepanjang langit-langit atau lantai untuk jarak yang sangat jauh, maka kabelnya mungkin perlu lebih panjang. Daripada menggunakan repeater, kita akan menghubungkan keempat switch ke router dengan menggunakan kabel fiber optic. Karena kami hanya menggunakan kabel serat optik untuk empat koneksi, ini akan membantu menekan biaya. Keempat koneksi ini juga akan menerima banyak lalu lintas, sehingga pilihan menggunakan kabel serat optik di sini membantu meningkatkan kinerja jaringan secara keseluruhan.

Perhatikan bahwa Gambar 4.5 mungkin merupakan pendekatan ekstrem untuk membangun jaringan kita. Kita tidak perlu menghubungkan setiap switch ke router. Alih-alih, sakelar dapat dirangkai bersama. Idenya adalah beberapa perangkat terhubung ke sakelar 1 dan sakelar 1 kemudian terhubung ke sakelar 2, yang juga terhubung ke perangkat lain. Sakelar 2 terhubung ke sakelar 3, yang juga menyambung ke perangkat lain. Untuk LAN kecil kami dalam contoh ini, kami dapat langsung menghubungkan sakelar ke satu router, tetapi jika bangunannya berbentuk berbeda atau kami memiliki lebih banyak sakelar, rantai daisy dapat mengurangi biaya potensial, karena kami dapat membatasi jumlahnya. dari router yang digunakan.

Untuk kabel kawat tembaga 10GBase-T, kita perlu memilih antara kabel kategori 6, kategori 6a, dan kategori 7. Kategori 6a dan 7 menggunakan kawat berpelindung daripada UTP dan lebih mahal. Kami memutuskan kabel kategori 6. Kami menggunakan kabel langsung untuk menghubungkan sumber daya kami ke sakelar dan kabel crossover untuk menghubungkan sakelar ke router kami.

Kami harus memutuskan bagaimana kami akan meletakkan kabel antara komputer di kantor dan sakelar dan antara sakelar dan router. Kita bisa membuat lubang di dinding dan memasukkan saluran tempat kabel bisa diletakkan. Jika kami melakukannya, kami mungkin ingin menempatkan saluran di dalam dinding yang dibagi antara dua kantor. Namun, pendekatan yang lebih murah adalah memasang baki di dekat langit-langit di lorong. Dengan baki seperti itu, kita dapat meletakkan kabel sehingga mengalir dari komputer, di sepanjang dinding, di dekat langit-langit, dan keluar dari kantor ke dalam baki. Kami harus membuat lubang di dekat langit-langit atau, jika mungkin, melewatkan kabel melalui langit-langit (misalnya, jika langit-langit memiliki ubin). Keuntungan dari baki adalah area kabel mudah diakses untuk pemeliharaan. Baki juga harus menjadi pendekatan yang lebih murah karena kita tidak perlu khawatir memasukkan saluran di dalam dinding.

Setiap sumber daya kami akan memerlukan NIC. Kami memilih NIC yang sesuai yang akan cocok dengan kabel yang telah kami pilih. NIC mungkin memiliki alamat MAC yang sudah terinstal. Jika tidak, kami akan menetapkan alamat MAC kami sendiri dengan menggunakan EUI-48 atau EUI-64. Meskipun semua sumber daya kami tampaknya akan berkomunikasi pada kecepatan bit yang sama, kami ingin memastikan bahwa semua NIC dan sakelar kami memiliki kemampuan negosiasi otomatis.

Sekarang yang tersisa hanyalah menginstal jaringan. Pertama, kami memasang kartu NIC ke sumber daya kami. Selanjutnya, kita harus menghubungkan sumber daya ke sakelar dengan kabel. Kami mungkin membeli kabel kategori 6 atau memotong kabel dan memasang jack kami sendiri jika kami membeli seikat besar kabel tembaga. Kami juga harus memasang baki di sepanjang lorong, sehingga kami dapat menempatkan kabel di baki tersebut. Kami harus meletakkan kabel dari setiap komputer atau printer ke lorong dan ke baki (sekali lagi, ini mungkin memerlukan pemotongan lubang di dinding). Dari sana, kabel secara kolektif masuk ke ruangan sakelar terdekat. Kami mungkin ingin menggabungkan semua kabel menjadi satu dengan menggunakan pengikat kabel. Setelah kabel mencapai sakelar, kami menyambungkan kabel ke port yang tersedia di sakelar. Kami juga menghubungkan sakelar ke router. Kita harus mengonfigurasi sakelar dan router kita (dan menetapkan alamat MAC jika kita tidak ingin menggunakan alamat default).

Menyelesaikan jaringan kami membutuhkan lapisan protokol yang lebih tinggi daripada dua lapisan yang ditawarkan oleh Ethernet. Misalnya, router beroperasi pada alamat jaringan, bukan pada alamat MAC. Baik perute maupun koneksi ke Internet bukanlah masalah yang menjadi perhatian Ethernet. Karena kami membahas alamat jaringan dengan TCP/IP di Bab 3, kami menghilangkannya dari diskusi kami di sini.

Kami telah membangun jaringan kami yang menggabungkan semua sumber daya komputasi kami. Sudahkah kita melakukan pekerjaan yang cukup dalam desainnya? Iya dan tidak. Kami telah memastikan bahwa setiap sakelar memiliki cukup sedikit perangkat yang

terhubung dengannya dan jarak yang dilalui kabel dapat dilakukan tanpa pengulang. Namun, kami belum mempertimbangkan keandalan sebagai faktor jaringan kami. Pertimbangkan apa yang akan terjadi jika kabel rusak. Sumber daya yang dihubungkan kabel ke sakelar tidak lagi dapat mengakses jaringan. Ini sendiri bukan masalah khusus karena kita harus memiliki kabel tambahan untuk memastikan kabel pengganti siap sesuai kebutuhan. Kurangnya akses jaringan hanya akan mempengaruhi satu orang. Namun, bagaimana jika kabel yang buruk menghubungkan sakelar ke router? Sekarang, seluruh bagian jaringan tidak dapat berkomunikasi dengan bagian jaringan lainnya.

Solusi kami untuk masalah ini adalah dengan menerimanya dan mengganti kabel sesuai kebutuhan. Sebagai alternatif, kami dapat menyediakan redundansi. Kita dapat melakukannya dengan menyambungkan setiap sakelar ke router melalui dua kabel terpisah. Kami juga dapat menggunakan dua router dan menghubungkan setiap sakelar ke kedua router. Salah satu solusi lebih mahal daripada versi jaringan kami yang kurang toleran terhadap kesalahan. Jika biaya menjadi perhatian yang lebih besar daripada keandalan, kami tidak akan memilih salah satu solusi. Membeli perute kedua adalah kemungkinan solusi kami yang paling mahal, tetapi tanggapan praktis karena membantu mencegah tidak hanya kegagalan kabel tetapi juga kegagalan perute itu sendiri. Terlepas dari biayanya, memiliki router tambahan yang dirangkai bersama-sama dapat memberikan redundansi yang lebih besar.

4.4 JARINGAN AREA NIRKABEL LOKAL

Sebagian besar LAN menggunakan kabel, menggunakan Ethernet, hingga sekitar tahun 2000. Dengan proliferasi perangkat nirkabel seperti laptop, ponsel pintar, dan tablet serta peningkatan kinerja NIC nirkabel, banyak LAN kabel eksklusif sekarang merupakan kombinasi kabel dan nirkabel. Dalam beberapa kasus, seperti di LAN rumah (jaringan area pribadi [PAN]), seluruh LAN mungkin nirkabel.

Pertimbangkan, misalnya, kedai kopi yang menawarkan akses Internet nirkabel. Masih ada komponen kabel yang membentuk sebagian dari jaringan kedai kopi. Kemungkinan besar akan ada komputer kantor, printer, dan mungkin server web terpisah. Komponen-komponen ini biasanya akan dihubungkan dengan kabel LAN. Selain itu, LAN berkabel akan memiliki router yang tidak hanya menghubungkan perangkat ini bersama-sama tetapi juga menghubungkannya ke penyedia layanan Internet (ISP) kedai kopi. Kemudian, ada LAN nirkabel yang digunakan pelanggan untuk terhubung ke Internet. Daripada membangun dua jaringan yang sepenuhnya terpisah, jaringan nirkabel akan terhubung ke jaringan kabel. Ini akan dicapai dengan menggunakan satu atau lebih titik akses nirkabel dan menghubungkannya ke router. Perute ini mungkin merupakan perute yang sama seperti yang digunakan untuk komponen jaringan berkabel, atau mungkin merupakan perute terpisah yang menghubungkan dirinya sendiri ke perute LAN berkabel. Melalui titik akses nirkabel pelanggan mendapatkan akses ke router dan dengan demikian bagian kabel dari jaringan, sehingga mereka dapat mengakses Internet. Pada bagian ini, kita fokus pada topologi, protokol, standar, dan komponen yang mengizinkan komunikasi nirkabel, apakah komponen nirkabel

merupakan bagian dari LAN nirkabel sepenuhnya atau komponen LAN yang juga berisi bagian berkabel.

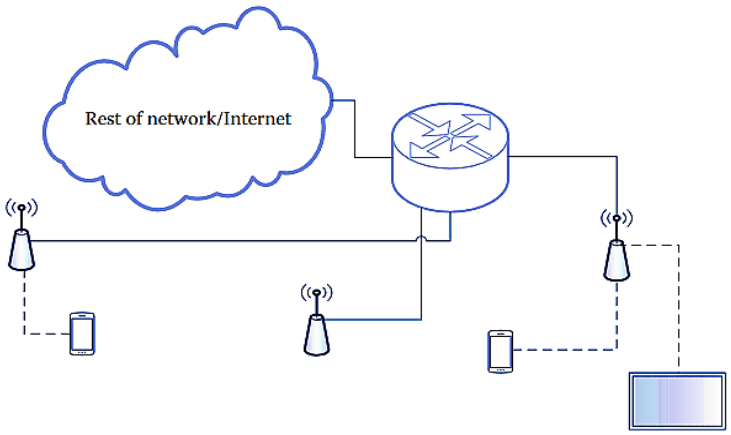
Untuk menyediakan akses nirkabel ke LAN kabel, kami memodifikasi tepi LAN. Tepi LAN (atau lapisan akses LAN) terdiri dari koneksi perangkat ke bagian LAN lainnya. Di LAN kabel, kami menghubungkan perangkat ke hub, switch, atau router. Dalam LAN nirkabel, perangkat nirkabel kami berkomunikasi dengan titik akses nirkabel (WAP), yang dengan sendirinya terhubung ke hub, sakelar, atau router. Perhatikan bahwa kami mengacu pada komunikasi LAN nirkabel secara umum sebagai Wi-Fi. Ini membedakan bentuk komunikasi dari bentuk komunikasi nirkabel lainnya seperti Bluetooth (yang akan kita bahas secara singkat di akhir bagian ini).

Saat kami menelusuri bagian ini, kami akan menjelaskan banyak standar nirkabel. Ini dilarang di bawah IEEE 802.11, dengan pengecualian PAN dan MAN nirkabel, yang masing-masing berada di bawah IEEE 802.15 dan IEEE 802.16. Kami juga akan membahas struktur komunikasi jaringan nirkabel. Perangkat berkomunikasi melalui paket yang disebut frame. Pada subbab selanjutnya, kami akan menyebutkan beberapa jenis bingkai. Kami akan mengeksplorasi jenis ini secara rinci di sub-bagian selanjutnya.

Topologi Dan Asosiasi Jaringan Area Lokal Nirkabel

Perubahan utama dari LAN kabel ke LAN nirkabel (WLAN), atau yang memungkinkan keduanya, adalah penyertaan WAP. Kami juga harus menyesuaikan NIC kami dari dalam perangkat kami menjadi NIC nirkabel (WNIC). Baik NIC dan WNIC adalah jenis perangkat yang serupa, tetapi WNIC berkomunikasi melalui gelombang radio frekuensi tinggi daripada melalui kabel.

WAP menyediakan akses ke LAN dengan menawarkan apa yang disebut hotspot. WAP menerima sinyal radio dari perangkat nirkabel terdekat dan meneruskan sinyal tersebut ke router. WAP dan router dihubungkan bersama oleh beberapa bentuk kabel (biasanya). Ketika pesan ditujukan untuk perangkat nirkabel, prosesnya dibalik di mana router meneruskan pesan ke WAP yang sesuai, yang kemudian menyiarkan pesan sebagai gelombang radio untuk diterima perangkat. Gambar 2.6 mengilustrasikan gagasan tiga perangkat nirkabel sebagai bagian dari jaringan nirkabel. Dalam hal ini, ada beberapa WAP yang terhubung ke satu router, yang dengan sendirinya terhubung ke seluruh jaringan. Tiga perangkat nirkabel berkomunikasi ke jaringan melalui WAP terdekat yang tersedia. Perhatikan bahwa meskipun gambar ini dan selanjutnya mengilustrasikan WAP yang terhubung ke router, untuk jaringan yang lebih besar, WAP mungkin terhubung ke sakelar terdekat.



Gambar 4.6 WAP terhubung ke router untuk menawarkan akses jaringan ke perangkat nirkabel.

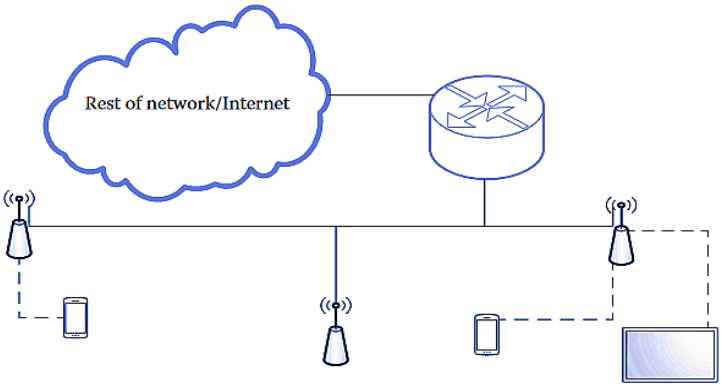
WAP memiliki bandwidth bersama yang terbatas. Artinya, bandwidth WAP adalah jumlah yang tetap. Semakin besar jumlah perangkat yang berbagi satu WAP, semakin sedikit bandwidth yang dapat digunakan oleh setiap perangkat. Oleh karena itu, kinerja dapat menurun karena lebih banyak perangkat berbagi WAP yang sama. Solusinya adalah dengan menambah jumlah WAP; namun, hal ini meningkatkan biaya dan menimbulkan masalah interferensi tersendiri, yang akan dibahas nanti.

Jumlah WAP dan cara penggunaannya menentukan topologi WLAN. Ada tiga bentuk topologi WLAN. Dua topologi pertama memanfaatkan WAP, membentuk apa yang terkadang disebut sebagai mode infrastruktur. Idenya adalah bahwa akses ke jaringan dilakukan dengan memiliki infrastruktur yang telah ditetapkan sebelumnya. Topologi ketiga tidak menggunakan WAP dan disebut sebagai mode ad hoc. Mari kita jelajahi topologi ini secara detail.

WLAN kecil, seperti yang ditemukan di rumah atau kantor kecil (dikenal sebagai jaringan SOHO) mungkin memiliki satu WAP. WLAN seperti itu diklasifikasikan sebagai topologi Basic Service Set (BSS) di mana semua perangkat nirkabel berkomunikasi dengan satu WAP dan WAP mencapai jaringan kabel melalui satu router. Dengan memanfaatkan satu WAP, kinerja dapat menurun jika ukuran WLAN (jumlah sumber daya yang menggunakan WAP) terlalu besar. Karena itu, kami mungkin ingin membangun jaringan yang lebih besar, jadi, kami beralih ke topologi berikutnya.

Jaringan dengan lebih dari satu WAP disebut sebagai topologi *Extended Service Set* (ESS). Dalam topologi ini, setiap WAP dapat terhubung ke routernya sendiri atau semua WAP dapat terhubung ke satu router. Dalam kasus sebelumnya, masing-masing router terhubung satu sama lain untuk membentuk LAN kabel. Jika beberapa WAP berbagi satu router, mereka dapat membuat koneksi ini melalui salah satu topologi jaringan (misalnya bus, ring, dan bintang). Kumpulan WAP ini dan konektivitasnya dalam ESS yang sama dikenal sebagai sistem distribusi. Gambar 4.7 mengilustrasikan satu kemungkinan ESS di mana tiga WAP membentuk sistem distribusi dan terhubung ke satu router. Dalam hal ini, sistem distribusi digabungkan melalui topologi bus. Perhatikan bahwa jika perangkat nirkabel berkomunikasi satu sama lain, mereka tidak perlu menggunakan bagian mana pun dari tulang punggung berkabel (router,

seluruh jaringan, dan Internet). Dengan demikian, sistem distribusi adalah kumpulan WAP yang mengizinkan komunikasi nirkabel tanpa backbone kabel.



Gambar 4.7 ESS dengan sistem distribusi.

Topologi ketiga adalah wireless ad hoc LAN. Jenis WLAN ini tidak memiliki WAP, dan sebagai gantinya, perangkat berkomunikasi langsung satu sama lain dalam format peer-to-peer. Bentuk topologi ini terkadang disebut sebagai Independent Basic Service Set (IBSS). Ada banyak variasi jaringan ad hoc nirkabel (WANET), jadi, kami akan membedakannya secara singkat di akhir bagian tentang WLAN ini.

Ketiga topologi ini berbeda dalam cara perangkat berkomunikasi satu sama lain. WLAN ad hoc tidak memiliki WAP, jadi, semua komunikasi dilakukan dari perangkat ke perangkat, tanpa WAP perantara. BSS memiliki satu WAP dimana semua pesan menggunakan satu WAP sebagai perantara. ESS memiliki banyak WAP, sehingga pesan diteruskan melalui satu atau lebih WAP. Pesan mungkin berpindah dari perangkat ke WAP ke perangkat atau dari perangkat ke WAP ke WAP lain (sistem distribusi) ke perangkat. Topologi ESS mengizinkan roaming, tetapi jaringan BSS dan ad hoc tidak mengizinkannya.

Agar pesan melintasi jaringan distribusi atau jaringan kabel untuk mencapai WLAN, perangkat harus dapat mengidentifikasi WLAN yang ingin digunakan untuk berkomunikasi. Identifikasi ditangani melalui nilai unik yang diberikan ke WLAN tersebut, yang dikenal sebagai Service Set ID, atau SSID. BSS memiliki SSID tunggal, ditugaskan ke WAP tunggal di BSS. Namun, ESS memiliki dua jenis SSID. Pertama, ada SSID untuk seluruh ESS. Ini terkadang disebut sebagai Extended Service Set ID (ESSID). Di dalam ESS, setiap WAP memiliki SSID-nya sendiri, yang disebut sebagai Basic Service Set ID (BSSID) WAP. Dalam kasus jaringan nirkabel ad hoc, perangkat harus dapat mengidentifikasi perangkat yang diizinkan untuk berkomunikasi, dan mereka melakukannya dengan menggunakan SSID. Dalam hal ini, SSID bukanlah nama permanen tetapi dibuat untuk membentuk jaringan ad hoc.

Perangkat nirkabel dan WAP menggunakan SSID untuk memastikan perutean pesan yang benar. Misalnya, router yang terhubung ke beberapa WAP di ESS akan menggunakan SSID pesan untuk menunjukkan WAP mana yang harus menerima pesan tersebut.

Biasanya, SSID adalah nama (karakter alfanumerik) yang diberikan oleh manusia untuk mendefinisikan jaringan. Panjang nama ini dibatasi hingga 32 karakter dan harus mengidentifikasi jaringan secara unik. Dua WLAN yang berdekatan tidak dapat menggunakan

nama yang sama (walaupun dua WLAN yang berlokasi di kota yang berbeda dapat memiliki nama yang sama). BSSID biasanya adalah alamat MAC dari WAP yang bersangkutan. Perhatikan bahwa meskipun WAP memiliki SSID tunggal (BSSID), ESS sebenarnya dapat memiliki banyak SSID (yaitu, beberapa ESSID). Alasannya adalah karena ESS itu sendiri dapat mendukung banyak WLAN. Karena ESS adalah kumpulan WAP dan sistem distribusi, perangkat keras tersebut dapat menawarkan layanan kepada kelompok pengguna yang berbeda. Suatu organisasi dapat menyediakan akses WiFi kepada karyawan, pelanggan, dan masyarakat umum. Karena karyawan mungkin memerlukan akses ke server file, dan pelanggan serta karyawan harus memiliki akses yang aman, kami dapat menyediakan tiga jaringan menggunakan perangkat keras yang sama. Dengan demikian, ESS mendukung tiga jaringan dengan SSID: karyawan, aman, dan publik.

Sekarang, mari kita pertimbangkan bagaimana perangkat nirkabel melakukan kontak dengan WAP. Pertama, kita memerlukan beberapa mekanisme dimana perangkat nirkabel dapat mengidentifikasi BSS/ESS di areanya dan, lebih khusus lagi, mengetahui SSID dari WAP yang ingin dikomunikasikan. Proses menghubungkan perangkat nirkabel ke WAP dikenal sebagai asosiasi.

Agar perangkat nirkabel dapat diasosiasikan dengan WAP, jabat tangan harus dilakukan. Prosesnya bervariasi tergantung pada apakah perangkat nirkabel berkomunikasi dalam BSS atau ESS. Dalam kedua kasus tersebut, komunikasi dimulai dengan WAP mengirimkan bingkai suar untuk mengumumkan kehadirannya. Sebuah WAP akan mengirimkan bingkai tersebut pada frekuensi radio yang telah ditetapkan dan pada interval tertentu, biasanya sekali setiap 100 satuan waktu, di mana satuan waktu, secara default, adalah 1024 mikrodetik. Mikrodetik adalah sepersejuta detik, jadi 100 * 1024 sepersejuta detik kira-kira sepersepuluh detik.

Bingkai suar menyertakan SSID untuk WAP (dan SSID untuk seluruh jaringan jika itu adalah ESS). Perangkat nirkabel memindai rentang frekuensi radio untuk bingkai suar. Saat menerima satu atau lebih bingkai suar, perangkat nirkabel dapat mencoba menyambung ke salah satu WAP yang sesuai. Sistem operasi perangkat nirkabel memiliki dua opsi. Pertama, jika menerima bingkai suar dari WAP pilihannya, ia menghubungi WAP tersebut untuk membuat sambungan. Jika tidak, daftar semua WAP yang tersedia yang telah ditemukan dan menunggu pengguna untuk memilih salah satu. Jelas, dalam BSS, daftar hanya akan berisi satu pilihan.

Dalam kasus di mana tidak ada pilihan yang lebih disukai di antara banyak WAP yang tersedia, diperlukan langkah-langkah tambahan. Pertama, klien mengirimkan bingkai permintaan penyelidikan ke semua WAP yang tersedia dengan menggunakan SSID WAP, seperti yang diterima dari bingkai beacon. Setiap WAP, saat menerima probe, akan merespons. Tanggapan akan mencakup informasi seperti apakah otentikasi diperlukan dan kekuatan sinyal relatif dari sinyal yang diterima. Dengan informasi ini, pengguna dapat membuat pilihan (umumnya, pengguna akan memilih WAP yang memberikan kekuatan sinyal terkuat, tetapi jika otentikasi diperlukan dan pengguna tidak mengetahui kata sandinya, pengguna dapat memilih yang lain. WAP). Pilihan ini menghasilkan siaran bingkai permintaan ke semua WAP, tetapi akan menyertakan SSID dari WAP yang dipilih. WAP yang dipilih

kemudian merespons. Jika autentikasi diperlukan, maka WAP akan merespons dengan bingkai permintaan autentikasi. Kemudian, pengguna harus mengirim kata sandi sebagai tanggapan. Jika autentikasi tidak diperlukan, atau pengguna berhasil mengotentikasi, maka perangkat nirkabel akan dikaitkan dengan WAP tersebut. WAP menyimpan tabel alamat MAC dari semua perangkat yang berkomunikasi dengannya, dan perangkat nirkabel menyimpan SSID dari WAP yang berkomunikasi dengannya.

Setiap WAP memiliki area jangkauan. Cakupan area ini menunjukkan jarak dari WAP dimana komunikasi yang efektif dimungkinkan. Pikirkan area ini sebagai rangkaian lingkaran konsentris. Saat seseorang bergerak lebih dekat ke pusat, kekuatan sinyal akan lebih kuat, dan oleh karena itu, kecepatan bit maksimum untuk komunikasi akan lebih tinggi. Kita dapat menemukan, misalnya, bahwa dalam jarak 30 meter, kecepatan transmisi adalah 54 Mbps, sedangkan pada jarak 60 meter, turun menjadi 20 Mbps, dan pada jarak 90 meter, turun menjadi 10 Mbps. Gagasan bahwa sinyal radio melemah pada jarak yang lebih jauh dikenal sebagai pelemahan. Perhatikan bahwa area cakupan mungkin tidak benar-benar melingkar. Degradasi sinyal dapat timbul karena hambatan seperti dinding, ubin langit-langit, pohon, jendela, dan perabot berukuran besar, serta karena kondisi atmosfer. Perangkat menampilkan kekuatan sinyal melalui batang, di mana 5 batang mewakili sinyal terkuat. Anda pasti pernah melihat ini di ponsel atau komputer laptop Anda.

Salah satu keunggulan komunikasi nirkabel adalah kemampuannya untuk bergerak saat mengakses jaringan. Ini dikenal sebagai jelajah. Jika perangkat menjelajah terlalu jauh dari WAP, konektivitas akan terputus. Lalu, bagaimana kita mendukung roaming?

Jika perangkat nirkabel menjelajah di area umum yang sama dengan WAP terkait, tidak akan terjadi apa-apa selama kekuatan sinyal masih dapat diterima. Namun, begitu kekuatan sinyal turun di bawah level yang dapat diandalkan, proses asosiasi dimulai lagi. Sistem operasi perangkat nirkabel akan mengumpulkan bingkai suar, dan setelah koneksi terputus atau berkurang ke tingkat yang tidak dapat diterima, sistem operasi memilih WAP baru untuk diasosiasikan. Membuat WAP baru dilakukan secara otomatis selama WAP milik ESS yang sama. Kami akan menganggap ini sebagai kasus di sini.

Apa yang terjadi pada setiap pesan yang telah Anda kirim, jika dikaitkan dengan satu WAP? Akankah tanggapan mencapai Anda? Karena setiap pesan yang dikirim dari perangkat Anda menyertakan alamat IP balasan Anda, respons akan dikirim melalui jaringan kabel (atau Internet) ke alamat tersebut. Bagaimana LAN Anda menemukan perangkat nirkabel Anda, karena Anda telah menjelajah dari satu WAP ke WAP lainnya? Alamat pengirim dirutekan ke router yang terhubung ke ESS Anda. Dari sana, router menyiarkan pesan ke semua WAP. Setiap WAP menyimpan daftar perangkat nirkabel yang terkait dengannya. Daftar ini mencakup alamat MAC dan alamat IP perangkat. Ketika WAP menerima respons siaran yang alamat IP tujuannya cocok dengan perangkat nirkabel yang terkait dengannya, WAP akan menyiarkan respons tersebut dengan menggunakan alamat MAC perangkat tersebut. Semua perangkat di area tersebut mungkin mendengar siaran tersebut, tetapi hanya perangkat Anda yang akan menerimanya. Oleh karena itu, saat Anda menjelajah, WAP yang baru terasosiasi mengetahui untuk menghubungi Anda melalui alamat IP, sedangkan WAP yang sebelumnya terasosiasi menghapus alamat IP Anda. Dengan cara ini, jaringan kabel dan router tidak

menyadari pergerakan Anda, dan hanya WAP di dalam ESS yang mengetahui pergerakan Anda. Kami mengeksplorasi handoff WAP ke WAP secara lebih rinci nanti di bagian ini.

Standar Jaringan Area Nirkabel Lokal

Standar LAN nirkabel telah diproduksi. Standar ini menentukan kecepatan transfer data, daya, frekuensi, jarak, dan modulasi perangkat nirkabel. Semua standar ini diterbitkan di bawah 802.11 dan termasuk, misalnya, 802.11a, 802.11b, 802.11g, dan 802.11n (ini termasuk standar yang paling signifikan). Standar 802.11 paling awal, diterbitkan pada tahun 1997, dianggap usang. Standar yang lebih baru telah meningkat dari yang asli dengan membutuhkan jarak yang lebih jauh, frekuensi jangkauan yang lebih luas, dan kecepatan transfer yang lebih tinggi. Misalnya, standar terbaru yang melarang kecepatan transfer adalah 802.11g, yang menentukan kecepatan transfer 54 Mbit per detik dan juga meminta kompatibilitas mundur kecepatan serendah 2 Mbit per detik.

Standar nirkabel asli menyerukan frekuensi radio 2,4 GHz (rentang frekuensi tinggi ini sebenarnya diklasifikasikan sebagai sinyal gelombang mikro). Frekuensi 2,4 GHz terus digunakan dalam standar saat ini. Sayangnya, banyak perangkat (termasuk perangkat Bluetooth, telepon nirkabel, dan oven microwave) dapat beroperasi dalam rentang yang sama ini, yang dapat menimbulkan interferensi. Standar 802.11n memperluas frekuensi radio untuk menyertakan 5 GHz, sedangkan standar 802.11ad mencakup 60 GHz, dan standar 802.11ah mengizinkan operasi WLAN kurang dari 1 GHz (termasuk pita tempat televisi beroperasi). Standar 802.11a juga menetapkan 12 saluran yang tumpang tindih, atau pita, sehingga beberapa perangkat terdekat dapat berkomunikasi pada saluran yang berbeda, tanpa sinyalnya saling mengganggu. Saluran yang tumpang tindih menyediakan cara untuk sedikit memvariasikan frekuensi. Misalnya, jika perangkat menggunakan 2,4 GHz, perangkat tetangga mungkin menggunakan frekuensi yang sedikit diubah seperti 2,422 GHz. Dalam spesifikasi asli 802.11, 11 saluran disarankan, masing-masing terpisah 22 MHz. Spesifikasi selanjutnya telah memvariasikan jarak saluran ini menjadi sesedikit 20 MHz dan sebanyak 160 MHz. Namun sebagian besar, saluran berdiri terpisah 20 MHz.

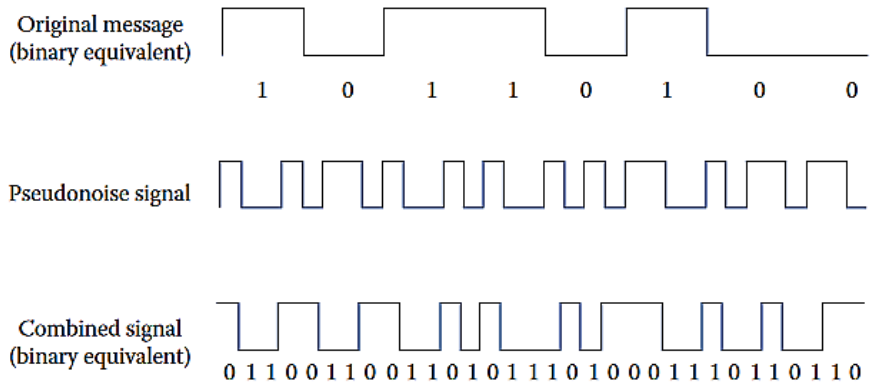
Perhatikan bahwa Anda tidak boleh bingung dengan terminologi yang digunakan di atas. Kecepatan bit transmisi dan frekuensi transmisi sering disebut dengan menggunakan istilah bandwidth. Secara harfiah, istilah lebar pita digunakan untuk frekuensi, tetapi kita sering juga menggunakannya untuk menjelaskan laju transmisi. Di sini, kami mengacu pada dua fitur secara lebih spesifik untuk menghindari kebingungan.

Standar asli 802.11 menyerukan jangkauan dalam ruangan 20 meter dan jangkauan luar ruangan 100 meter untuk komunikasi nirkabel. Nilai-nilai ini perlahan-lahan ditingkatkan, dengan 802.11n membutuhkan 70 meter di dalam ruangan dan 250 meter di luar ruangan. Menariknya, 802.ad yang lebih baru menyerukan jarak 60 dan 100 meter untuk komunikasi dalam ruangan dan komunikasi luar ruangan. Diharapkan standar 802.ay (jatuh tempo pada 2017) masing-masing akan membutuhkan rentang 60 dan 1000 meter.

Standar mendefinisikan bentuk modulasi, istilah yang diperkenalkan pada Bab 1 sehubungan dengan MODEM. Modulasi secara harfiah didefinisikan sebagai variasi yang ditemukan dalam bentuk gelombang. Dengan komunikasi nirkabel, idenya adalah untuk

mengungkapkan bagaimana gelombang radio akan bervariasi selama transmisi. Empat bentuk modulasi yang berbeda telah diusulkan untuk komunikasi nirkabel.

Bentuk modulasi pertama yang ditentukan adalah *Direct-Sequence Spread Spectrum* (DSSS). Dalam bentuk modulasi ini, sinyal (data) ditempatkan di dalam sinyal lain. Sinyal data berada pada frekuensi standar (misalnya, 2,4 GHz), sedangkan sinyal lainnya dihasilkan pada frekuensi yang jauh lebih tinggi. Bit dari sinyal lain ini kemudian ditempatkan dengan lebih kompak. Sinyal lainnya dikenal sebagai pseudonoise; itu merupakan serangkaian bit, yang secara kolektif disebut chip. Sinyal yang ditransmisikan kemudian terdiri dari pesan sebenarnya, pada frekuensi yang lebih rendah, dan chip, pada frekuensi yang lebih tinggi. Karena chip dikemas lebih rapat dalam pesan, ruang yang diperlukan untuk mengkodekan 1 bit dalam chip jauh lebih sedikit daripada ruang yang diperlukan untuk mengkodekan 1 bit dalam pesan yang sebenarnya. Gambar 4.8 mengilustrasikan bagaimana pesan digabungkan dengan chip untuk membentuk pesan yang sedang dikirim. Perhatikan bagaimana potongan pesan jauh lebih tersebar daripada chip. Penerima, yang mengetahui struktur chip, akan memisahkan kedua pesan tersebut, membuang suara semu. Anda mungkin bertanya-tanya mengapa kita perlu menyandikan pesan dengan menggunakan bentuk modulasi ini. Jawabannya ada hubungannya dengan mengurangi jumlah kebisingan statis atau putih yang dapat ditemukan dalam sinyal.



Gambar 4.8 Sinyal dimodulasi menggunakan DSSS.

Bentuk modulasi selanjutnya adalah frequency-hopping spread spectrum (FHSS). Dalam FHSS, beberapa frekuensi digunakan untuk mengirim pesan dengan memodulasi frekuensi selama transmisi. Perubahan frekuensi berada dalam satu frekuensi yang terkenal tetapi disesuaikan dalam pita yang diizinkan. Perubahan subfrekuensi ini mengikuti urutan yang diketahui oleh pemancar dan penerima. Idenya adalah bahwa gangguan apa pun yang muncul dalam satu pita diminimalkan karena sinyal tetap berada di pita itu hanya untuk durasi yang singkat. Selain itu, perangkat yang mencoba menguping komunikasi mungkin tidak dapat mengambil seluruh pesan. Jika band awal tidak diketahui, maka penyadap tidak akan mengikuti pola band yang sama, atau pergeseran frekuensi, untuk mengambil seluruh pesan, membuat komunikasi lebih sulit untuk dicegat. Penggunaan FHSS juga memungkinkan perangkat terdekat untuk berkomunikasi dengan perangkat lain tanpa gangguan, selama tidak ada pita yang tumpang tindih selama komunikasi. Meskipun spesifikasi asli 802.11

menyerukan penggunaan DSSS dan FHSS, dalam praktiknya, FHSS tidak digunakan di perangkat nirkabel (namun, variasi FHSS telah menemukan aplikasi di perangkat Bluetooth).

Bentuk modulasi ketiga dikenal sebagai *orthogonal frequency-division multiplexing* (OFDM). Seperti semua bentuk multiplexing, OFDM membagi pesan menjadi beberapa bagian yang masing-masing disiarkan secara bersamaan dengan menggunakan frekuensi yang berbeda. Multiplexing pembagian frekuensi tidak jarang dalam bentuk telekomunikasi. Apa yang unik tentang OFDM adalah bahwa sinyal independen saling orthogonal, dengan masing-masing sinyal berjarak sama di band tetangga. Keuntungannya di sini adalah dengan memberikan sinyal melintasi rentang pita, interferensi diminimalkan karena pita tersebut ditempati. Karena penggunaan multiplexing, pesan dapat dikirim lebih cepat, karena sebagian dikirim pada waktu yang bersamaan.

Faktanya, OFDM telah menemukan penggunaan luas dalam komunikasi nirkabel dan merupakan metode pilihan untuk standar 802.11a dan 802.11g, dan varian OFDM adalah metode pilihan dalam standar 802.11n dan 802.11.ac. Hanya 802.11b yang menggunakan format yang berbeda, DSSS. Namun, ada pendekatan modulasi lain yang juga umum digunakan. Ini adalah varian OFDM yang disebut OFDM multiple-input, multiple-output (MIMO-OFDM). Kemampuan untuk menyediakan banyak input dan banyak output sekaligus ditangani dengan memanfaatkan beberapa antena. Dengan OFDM dan MIMO, banyak sinyal dikirim melalui beberapa antena dan dimultipleks melalui band yang berbeda. Ini memberikan jumlah transfer data terbesar. Faktanya, standar 802.11ac meminta MIMO-OFDM untuk dapat membawa sebanyak delapan aliran simultan, masing-masing dimultipleks untuk membagi sinyal menjadi beberapa frekuensi.

Standar lain telah diperkenalkan yang mencakup fitur yang tidak dibahas di sini. Ini termasuk, misalnya, keamanan, ekstensi untuk negara tertentu (misalnya, Jepang dan China), perluasan atau protokol baru, dan bentuk komunikasi nirkabel lainnya, termasuk mobil pintar dan televisi pintar. Karena kurang berpengaruh pada diskusi kita tentang LAN nirkabel dan perangkat nirkabel, kita tidak akan membahasnya lebih jauh (selain keamanan, yang akan kita lihat nanti di bab ini).

Perangkat Keras Nirkabel

WLAN terdiri dari setidaknya satu jenis komponen dan biasanya dua atau tiga. Pertama, kita membutuhkan perangkat nirkabel (misalnya, laptop dan ponsel pintar). Perangkat ini memerlukan NIC nirkabel. Hanya dengan perangkat nirkabel, kita dapat membangun jaringan ad hoc tetapi bukan WLAN yang sebenarnya. Untuk WLAN, kita membutuhkan satu atau lebih WAP. Komponen ketiga adalah router. Peran router adalah menghubungkan WLAN kita ke LAN kabel kita (dan kemungkinan besar, Internet). Komponen terakhir ini opsional karena kita dapat membangun WLAN tanpa koneksi ke jaringan kabel; namun, hal ini secara drastis membatasi kegunaan WLAN.

Mari kita jelajahi komponen ini dari sudut pandang fisik, dimulai dengan WNIC, kecuali bahwa ia harus dapat mengirimkan dan menerima data melalui sinyal radio. Karena berhubungan dengan komunikasi radio, maka WNIC memiliki antena. Sebagian besar WNIC berada di dalam unit sistem komputer, sehingga antena sebenarnya berada di dalam komputer, bukan antena eksternal. Pengecualian untuk ini adalah ketika WNIC terhubung ke

perangkat melalui port *Universal Serial Bus* (USB) eksternal. Namun, meskipun demikian, antena biasanya berada di dalam perangkat antarmuka. Gambar 4.9 membandingkan perangkat antarmuka nirkabel USB dengan antena internal dengan WNIC dengan antena eksternal. Selain antena dan kemampuan berkomunikasi melalui gelombang radio, NIC sangat mirip dengan NIC kabel, hanya saja kecepatan transfernya lebih rendah.

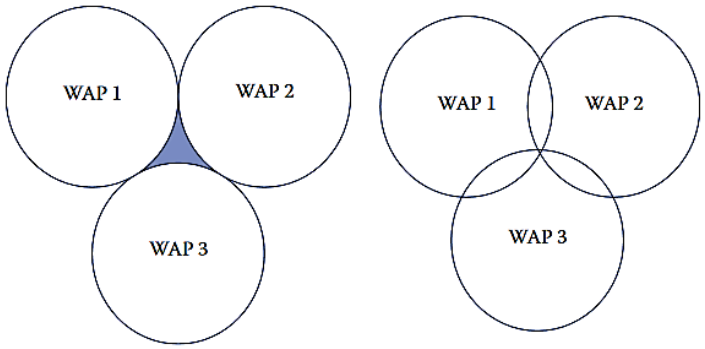
Gambar 4.9 Antarmuka USB nirkabel dan NIC nirkabel.

Ingatlah bahwa standar 802.3 menyertakan spesifikasi lengkap Ethernet. Saat merancang standar untuk 802.11, dirasakan bahwa perangkat nirkabel harus dibangun di atas standar Ethernet. Dengan demikian, setiap WNIC harus dapat menangani OSI layer 2 dan khususnya sublayer MAC. Ini dikenal sebagai MAC Sublayer Management Entity (MLME). Kemampuan WNIC untuk mengelola sublapisan MAC dapat diimplementasikan baik secara langsung di perangkat keras atau oleh sistem operasi perangkat atau perangkat lunak lainnya. Versi yang diimplementasikan perangkat keras disebut sebagai perangkat FullMAC (atau perangkat HardMAC), sedangkan versi yang diimplementasikan dalam perangkat lunak terpisah disebut perangkat SoftMAC. Untuk mengimplementasikan WNIC FullMAC, biasanya mekanisme yang tepat untuk menangani sublapisan MAC diimplementasikan secara langsung dalam chip yang ditempatkan pada WNIC. Ini mungkin sedikit meningkatkan biaya WNIC, tetapi saat ini, peningkatan biaya seperti itu minimal. Baik Linux maupun FreeBSD Unix mengimplementasikan MLME, sehingga pengguna salah satu sistem operasi ini tidak perlu membeli perangkat FullMAC.

WLAN hanya dapat terdiri dari perangkat nirkabel yang diatur sebagai jaringan ad hoc. Dalam kasus seperti itu, perangkat nirkabel berkomunikasi secara langsung satu sama lain daripada melalui WAP dan sistem distribusi. Keuntungan dari jaringan ad hoc adalah lebih murah dan dapat dikonfigurasi sendiri karena bentuk dan ukurannya berubah saat Anda menambah atau menghapus perangkat nirkabel. Sayangnya, kekurangan jaringan ad hoc membuatnya kurang diminati. Kurangnya perangkat terpusat, jaringan ad hoc lebih sulit untuk terhubung ke LAN kabel dan ke Internet (dan, pada kenyataannya, mungkin tidak memiliki sarana untuk terhubung ke Internet). Jaringan ad hoc juga menempatkan beban yang lebih besar pada perangkat nirkabel. Tanpa WAP, perangkat nirkabellah yang harus melakukan pendistribusian pesan dari satu bagian jaringan ke bagian lainnya. Pertimbangkan, misalnya, jaringan ad hoc di mana dua perangkat terjauh berkomunikasi satu sama lain. Perangkat

nirkabel lain mungkin perlu dilibatkan untuk menjembatani dua jarak antara kedua perangkat ini.

Jadi, kami ingin memanfaatkan WAP, jika memungkinkan. Ini mengarah ke topologi BSS dan ESS. WAP khas akan berisi beberapa bentuk antena omnidirectional, sehingga komunikasi tidak terbatas saling berhadapan. Di sisi lain, beberapa perencanaan penempatan setiap WAP berguna. Kami biasanya ingin memposisikan WAP sesentral mungkin ke area tersebut. Kami juga ingin memastikan bahwa area jangkauan bebas dari halangan yang dapat memblokir sinyal radio, seperti dinding tebal.



Gambar 4.10 Menempatkan WAP untuk jangkauan maksimum.

Dengan ESS, kami akan memiliki banyak WAP untuk mencakup area WLAN kami. Bagaimana kita memposisikan WAP tersebut? Misalnya, haruskah WAP ditempatkan cukup jauh untuk memaksimalkan total area yang dicakup, atau apakah kita ingin menempatkannya lebih dekat sehingga tidak ada celah di area cakupan? Pada Gambar 4.10, kita melihat dua kemungkinan konfigurasi dari tiga WAP. Di sebelah kiri, ada area kecil yang berada di luar area jangkauan WAP, sedangkan di sebelah kanan, karena tumpang tindih antara ketiga WAP, cakupannya lengkap, total area yang dicakup lebih sedikit.

Cakupan area, seperti yang ditunjukkan pada bagian kiri Gambar 4.10, mungkin tidak akurat. Cakupan berkurang saat Anda menjauh dari WAP, tetapi sinyal tidak tiba-tiba terputus. Alih-alih, area abu-abu pada gambar mungkin sebenarnya tercakup, tetapi dengan kekuatan sinyal yang lebih rendah, dan karena itu menawarkan tingkat transmisi yang lebih rendah. Dengan demikian, posisi WAP seperti itu mungkin tidak diinginkan karena total area yang dicakup dimaksimalkan. Pilihan lainnya adalah menempatkan WAP di lokasi sedemikian rupa sehingga area yang tidak tertutup tidak dapat diakses oleh manusia (seperti tembok). Jika area abu-abu dari gambar ini sebenarnya adalah lemari, maka mungkin dapat diterima jika area seperti itu menerima pengurangan cakupan. Namun, WAP tidak terlalu mahal, sehingga harus membeli lebih banyak WAP untuk mencakup seluruh area biasanya tidak akan merugikan organisasi yang membangun WLAN.

Dengan demikian, terlalu banyak WAP di satu tempat dapat menimbulkan masalah yang dikenal sebagai interferensi co-channel. Jika WAP menggunakan frekuensi radio yang sama, maka sinyalnya dari WAP tetangga dapat mengganggu satu sama lain. Ini adalah bentuk

crosstalk nirkabel, mirip dengan masalah yang dapat terjadi pada kabel UTP jika tidak diproduksi atau dipasang kabel dengan benar.

Hal lain yang perlu diingat adalah bahwa WAP menyediakan bandwidth bersama di antara perangkat nirkabel yang berkomunikasi di BSS atau ESS. Tidak seperti jaringan bus berkabel, di mana lebar pita bersama tidak lagi diperlukan (karena penggunaan topologi bintang), WLAN mungkin masih memerlukan bentuk penanganan tabrakan. Ini karena WAP mungkin diminta untuk menangani banyak pesan kapan saja. WAP menggunakan bentuk CSMA/CD yang dikenal sebagai *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). Bentuk utama penerapan CSMA/CA adalah melalui permintaan terpisah untuk mengirim dan izin untuk mengirim pesan.

Titik akses nirkabel dapat dikonfigurasi. Ada banyak pengaturan yang dapat Anda buat. Di antara yang paling signifikan adalah SSID WAP dan kata sandi enkripsi WAP. Di luar ini, Anda dapat menetapkan frekuensi radio yang akan digunakan WAP (rentang tersedia berdasarkan negara penggunaan, seperti Amerika Serikat vs Eropa vs Jepang), teknik modulasi, kecepatan data yang harus diterapkan (beberapa kecepatan dapat dipilih, di mana WAP menggunakan kecepatan yang paling sesuai dengan perangkat nirkabel tertentu), pengaturan daya, dan sensitivitas penerima.

Titik akses nirkabel sering berada di desktop atau dipasang di langit-langit. Antena mungkin eksternal atau internal ke perangkat. Titik akses nirkabel ringan dan panjang dan lebarnya kurang dari satu kaki. Selain dari pilihan-pilihan ini, ada sejumlah faktor yang dapat mempengaruhi kinerja WAP. Ini termasuk beberapa pengaturan seperti dijelaskan di atas, interferensi dari sumber lain, dan, sebagaimana disebutkan, penempatan WAP. Selain itu, kinerja akan terpengaruh karena semakin banyak perangkat yang berkomunikasi dengan (atau melalui) WAP. Saat ini sudah umum bagi WAP untuk dapat menangani sebanyak 30 komunikasi yang berbeda sekaligus.

Perangkat terakhir untuk WLAN kami, yang bersifat opsional, adalah router. Kami biasanya menghubungkan WAP kami ke router melalui kabel. Namun, tidak seperti switch dan komputer, WAP tidak perlu diberi alamat IP, karena WAP akan menggunakan BSSID untuk menunjukkan dirinya sendiri. WAP mungkin memiliki alamat IP tergantung pada perannya dalam jaringan. Misalnya, lihat paragraf berikutnya.

Banyak WAP saat ini sebenarnya adalah WAP dan router. Bentuk WAP ini digunakan di banyak jaringan SOHO untuk membatasi perangkat keras yang diperlukan untuk menyiapkan LAN berkabel dan nirkabel. Kombinasi WAP/router seperti itu memiliki beberapa port untuk koneksi kabel, tetapi WAP/router terutama ditujukan untuk mendukung komunikasi perangkat nirkabel. Selain itu, beberapa WAP ditugaskan untuk mengalokasikan alamat IP secara dinamis ke perangkat nirkabel yang terhubung dengannya. WAP semacam itu membutuhkan alamat IP-nya sendiri. Kami mengeksplorasi alokasi dinamis alamat IP di Bab 5 dan 6.

Kami mungkin mempertimbangkan perangkat opsional lain untuk WLAN kami, pengontrol nirkabel. Setiap WAP bekerja secara independen dari WAP lainnya di ESS. Ada contoh di mana upaya koordinasi dapat meningkatkan kinerja. Selain itu, pengiriman sinyal roaming dari satu WAP ke WAP lainnya biasanya diprakarsai oleh perangkat nirkabel. Peran

pengontrol nirkabel adalah untuk mengawasi semua komunikasi nirkabel dalam upaya meningkatkan komunikasi. Ada beberapa cara di mana pengontrol nirkabel dapat dilibatkan.

Pertama, pengalihan dari satu WAP ke WAP lainnya dapat dipicu oleh pengontrol nirkabel. Akibatnya, pengontrol nirkabel memaksa perangkat untuk melepaskan hubungannya dengan satu WAP untuk meningkatkan komunikasi karena WAP lain menawarkan kekuatan sinyal yang lebih besar atau kecepatan transfer yang lebih baik. Ini mungkin karena roaming keluar dari satu area jangkauan, tetapi mungkin juga karena WAP saat ini sedang sibuk dengan beberapa perangkat lain, sehingga bandwidth bersama yang tersedia kurang dari WAP terdekat lainnya.

Kedua, pengontrol nirkabel dapat melacak band mana yang sedang digunakan di sekitar area tersebut. Ini dapat membantu WAP memutuskan apakah akan mengganti band atau mempertahankan komunikasi dengan band saat ini. Sebagai contoh, jika dua WAP berdekatan dan satu WAP menggunakan pita 10 MHz lebih tinggi dari yang lain, pengontrol dapat merekomendasikan kepada WAP lain untuk tidak menaikkan pitanya sebesar 10 MHz, meskipun ada interferensi pada pita saat ini.

Ketiga, pengontrol nirkabel dapat memantau berbagai bentuk interferensi dan menginformasikan WAP untuk menyesuaikan frekuensinya. Interferensi dapat berasal dari WAP itu sendiri tetapi juga dapat dikaitkan dengan perubahan lingkungan (seperti penggunaan oven microwave di dekat situ) atau masuknya perangkat nirkabel yang tidak terduga.

Biasanya, pengontrol nirkabel akan berkomunikasi dengan semua WAP untuk membentuk model keadaan jaringan nirkabel. Pengontrol ini menggunakan protokol mereka sendiri yang dikenal sebagai Protokol Titik Akses Ringan, yang mereka gunakan untuk berkomunikasi dengan WAP.

Frame Jaringan Area Nirkabel Lokal

Melihat bahwa komunikasi antara perangkat nirkabel dan WAP ditangani melalui bingkai. Mari kita jelajahi bingkai ini secara mendetail. Bingkai ditentukan oleh standar 802.11 sebagai implementasi OSI layer 2 (data link layer). Setiap bingkai komunikasi nirkabel terdiri dari header MAC dan trailer (urutan pemeriksaan bingkai atau checksum). Selain itu, rangka nirkabel memiliki muatan opsional, tergantung pada jenis rangka. Header MAC panjangnya 30-byte dan dijelaskan pada Tabel 4.2.

Keempat alamat 6-byte semuanya adalah alamat MAC dari berbagai perangkat yang terlibat dalam komunikasi. Mereka akan terdiri dari empat perangkat: perangkat sumber, perangkat tujuan, SSID BSS, dan perangkat transmisi. Perangkat terakhir ini adalah item yang harus diakui saat pesan diterima. Kami akan segera menjelajahi keempat alamat ini secara lebih rinci. Pertama-tama mari kita tentukan byte kontrol bingkai, yang perlu kita ketahui sebelum kita dapat menentukan empat alamat spesifik yang digunakan.

Tabel 4.2 Header MAC Bingkai Nirkabel

Bidang	Ukuran (dalam byte)	Penggunaan
Kontrol bingkai	2	Mendefinisikan protokol, tipe, dan informasi lainnya; lihat pembahasan di bawah dan Tabel 4.3

Durasi/ID	2	Menunjukkan baik durasi yang dicadangkan oleh WAP untuk sinyal radio dari perangkat nirkabel (yaitu, periode bebas pertengkaran yang dicadangkan oleh WAP) atau ID asosiasi, yang akan berupa SSID, bergantung pada jenis bingkai, sebagaimana ditentukan dalam byte kontrol bingkai
Alamat 1	6	Lihat Tabel 4.4
Alamat 2	6	Lihat Tabel 4.4
Alamat 3	6	Lihat Tabel 4.4
Kontrol urutan	2	Menentukan nomor fragmen (4 bit) dan nomor urut (12 bit) untuk pengurutan saat pesan dibagi menjadi beberapa bingkai; juga memungkinkan frame duplikat untuk dijatuhkan
Alamat 4	6	Lihat Tabel 4.4

Bidang kontrol bingkai adalah 2 byte, terdiri dari 11 subbidang masing-masing 1–4 bit. Bidang ini didefinisikan dalam Tabel 4.3. Tipe dan subtipe digabungkan untuk menentukan tipe bingkai yang tepat. Bidang Ke DS dan Dari DS digabungkan menentukan penggunaan empat alamat MAC.

Tiga jenis frame (manajemen, kontrol, dan data) dibagi lagi. Bingkai manajemen termasuk otentikasi, permintaan asosiasi, respon asosiasi, suar, deauthentication, disasosiasi, permintaan probe, respon probe, permintaan reasosiasi, dan respon reasosiasi.

Sebuah bingkai deauthentication, seperti namanya, menunjukkan bahwa pengirim ingin menghentikan komunikasi aman tetapi terus berhubungan dengan WAP. Bingkai disasosiasi menunjukkan bahwa pengirim ingin memutuskan hubungan dengan WAP. Reasosiasi adalah situasi dimana klien roaming berpindah dari satu WAP ke WAP lainnya. Dalam permintaan reasosiasi, perangkat nirkabel meminta WAP baru untuk berkoordinasi dengan WAP lama untuk meneruskan data yang diminta dengan benar saat klien dikaitkan dengan WAP lama. Sebuah kerangka respons reasosiasi dikirim oleh WAP baru untuk menunjukkan penerimaan atau penolakan permintaan.

Tabel 4.3 Bidang Kontrol Frame

Bidang	Ukuran (dalam bit)	Penggunaan
Versi protokol	2	Setel ke 00; pola lain dicadangkan untuk penggunaan di masa mendatang
Jenis	2	Menunjukkan manajemen, kontrol, atau bingkai data
Subtipe	4	Dibahas dalam teks
Ke DS	1	Menunjukkan arah pesan, ke atau dari sistem distribusi (DS)
Dari DS	1	komponen; kedua nilai adalah 0 jika sebuah pesan dikirimkan antara dua peer dalam jaringan ad hoc (tanpa WAP)
Lebih banyak fragmen	1	Setel jika pesan terdiri dari beberapa bingkai dan lebih banyak bingkai akan datang
Mencoba kembali	1	Setel jika bingkai ini dikirim ulang karena pengiriman yang gagal atau salah

Manajemen daya	1	Menunjukkan status manajemen daya pengirim (manajemen daya hidup atau mati)
Lebih banyak data	1	Jika disetel, menunjukkan bahwa bingkai harus disangga dan dikirim bersama saat penerima tersedia
WEP	1	Jika disetel, menunjukkan bahwa enkripsi sedang digunakan
Memesan	1	Jika disetel, menunjukkan bahwa pengurutan yang ketat untuk fragmen yang dikirim harus digunakan

Frame kontrol terdiri dari power save polling, request to send, clear to send, acknowledgment, dan beberapa jenis frame bebas contention. Bingkai jajak pendapat hemat daya digunakan untuk menunjukkan bahwa perangkat telah bangun dari mode hemat daya dan ingin mengetahui apakah ada bingkai buffer yang menunggu. Dengan cara ini, WAP akan bertugas mengumpulkan bingkai data untuk perangkat nirkabel ketika mode hemat dayanya tidak tersedia.

Permintaan untuk mengirim frame bersifat opsional tetapi dapat digunakan untuk mengurangi pertentangan pesan dalam situasi di mana beberapa perangkat mencoba berkomunikasi pada waktu yang sama melalui satu WAP. Permintaan meminta izin untuk melanjutkan dengan bingkai data, sedangkan bingkai izin untuk mengirim adalah pengakuan dari WAP bahwa perangkat nirkabel yang meminta dapat melanjutkan. Frame clear to send akan mencakup durasi berapa lama jendela terbuka untuk pemohon, sementara perangkat lain diminta untuk menunggu selama jangka waktu tersebut. Selama interval jendela terbuka ini, juga dikenal sebagai periode bebas pertengkaran, tersedia bentuk bingkai kontrol lainnya. Ini termasuk pengakuan bebas pertengkaran, jajak pendapat bebas perselisihan, kombinasi keduanya, dan akhir bebas perselisihan.

Bingkai data juga diketik. Yang paling umum adalah kerangka data yang ketat. Namun, ada varian untuk kerangka data yang dapat digabungkan dengan proses kontrol seperti data dan pengakuan bebas pertentangan, data dan polling bebas pertentangan, dan data yang digabungkan dengan pengakuan dan pemungutan suara. Bentuk lain dari bingkai data dikenal sebagai fungsi null, yang dikirimkan oleh perangkat nirkabel untuk menunjukkan perubahan manajemen daya ke WAP. Frame null tidak membawa data, meskipun dianggap sebagai frame data.

Bingkai data merangkum data yang sedang dikirim. Karena data ini adalah bagian dari beberapa komunikasi yang lebih besar (misalnya, pesan email dan permintaan atau tanggapan HTTP), kerangka data berisi headernya sendiri berdasarkan protokol pesan. Dengan demikian, kerangka data bukan sekadar data tetapi juga informasi header yang digunakan di seluruh protokol OSI yang diperlukan untuk mengidentifikasi informasi spesifikasi protokol, alamat IP, dan sebagainya. Header dan trailer MAC hanya menyandikan komponen layer 2 (data link layer) dari frame, yang diperlukan untuk bagian nirkabel dari komunikasi.

Di antara header dan trailer MAC adalah payload. Untuk frame data, payload adalah data yang menyusun pesan (atau bagian dari pesan yang sesuai dengan frame ini). Dengan bingkai kontrol dan manajemen, payload bersifat opsional, tergantung pada jenisnya. Bingkai suar, misalnya, akan menyertakan informasi berikut:

- Stempel waktu berdasarkan jam WAP, digunakan untuk sinkronisasi dengan perangkat nirkabel
- Interval antara bingkai suar yang dikirim oleh WAP ini
- Informasi kemampuan, termasuk jenis jaringan (BSS atau ESS), apakah enkripsi didukung, apakah polling didukung, dan sebagainya
- SSID WAP
- Kecepatan transfer yang didukung
- Berbagai set parameter seperti jika frame bebas contention didukung

Ingatlah bahwa header MAC mencakup hingga empat alamat. Ini didefinisikan sebagai alamat penerima (perangkat yang akan menerima frame), alamat pemancar (perangkat yang telah mengirimkan frame), alamat sumber, dan alamat tujuan. Apa perbedaan antara penerima dan tujuan, serta pemancar dan sumber? Bahkan, mereka mungkin sama (yaitu, penerima dan tujuan, dan sumber dan pemancar), tetapi itu tergantung dari mana pesan dikirim dari dan ke mana.

Sumber dan tujuan adalah, seperti yang kita lihat dengan model OSI, pencetus pesan dan tujuan akhir pesan. Jika server web mengembalikan halaman web ke perangkat nirkabel, maka server web adalah sumber dan perangkat nirkabel adalah tujuannya. Pemancar dan penerima adalah dua perangkat yang saat ini terlibat dalam tindakan pengiriman dan penerimaan pesan. Ini mungkin perangkat perantara, tergantung di mana pesan berada dan jenis WLAN. Melanjutkan contoh kami, halaman web diterima oleh router di LAN kabel kami dan diteruskan ke sistem distribusi WLAN kami. WAP yang terkait dengan alamat IP pesan akan menyiarkan pesan tersebut. Jadi, WAP adalah pemancar dan perangkat nirkabel adalah penerima. Jadi, dalam hal ini, perangkat nirkabel adalah penerima sekaligus tujuannya, sedangkan WAP adalah pemancarnya.

Mari kita pertimbangkan variasi. WLAN kami mencakup area yang lebih luas sehingga WAP yang terhubung ke router kami menyiarkan pesannya ke WAP yang ditempatkan lebih jauh. Biasanya WAP itulah yang menerima pesan dan menyiarkan ulang ke WAP lain atau ke perangkat nirkabel. Intinya, WAP lain berfungsi sebagai repeater, seperti di jaringan Ethernet awal. Dari contoh kita, router telah meneruskan pesan ke WAP, yang menyiarkan pesan tersebut. WAP yang diasosiasikan dengan perangkat nirkabel kami adalah satu-satunya yang tertarik dengan pesan tersebut karena menemukan kecocokan dengan alamat MAC tujuan dalam daftar tersimpannya. Jadi di sini, WAP penyiaran adalah pemancar dan WAP yang terkait dengan perangkat kita adalah penerima.

Tabel 4.4 menyajikan empat kombinasi alamat MAC berdasarkan nilai bidang Ke DS dan Dari DS. Kami melihat bahwa ketika bidang Ke DS dan Dari DS adalah 1, ada empat alamat unik. Alasan untuk hal ini adalah baik pengirim maupun penerima adalah WAP, sehingga keduanya tidak akan menjadi sumber atau tujuan. Dalam kebanyakan kasus lain, tiga perangkat terpisah terlibat: sumber, tujuan, dan WAP. Pengecualiannya adalah pada jaringan ad hoc, tidak ada WAP. Jaringan seperti itu akan memiliki Ke DS dan Dari DS terdaftar sebagai 0, dan alamat ketiga sebenarnya akan menjadi salinan dari alamat sumber. Perhatikan bahwa bingkai kontrol dan manajemen juga akan memiliki To DS dan From DS sebagai 0.

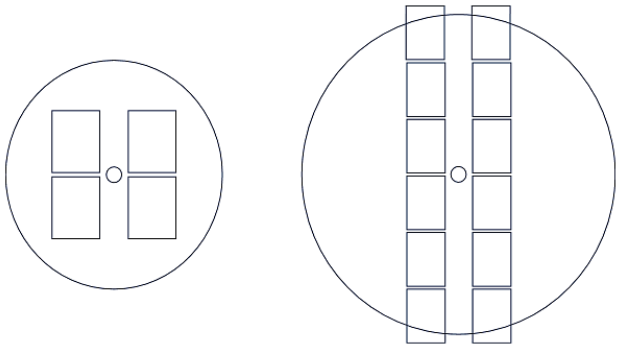
Tabel 4.4 Alamat MAC Digunakan dalam Bingkai Berdasarkan Nilai Ke DS dan Dari DS

Ke DS	Dari DS	Alamat 1	Alamat 2	Alamat 3	Alamat 4
0	0	Tujuan	Sumber	SSID WAP	Tidak ada
0	1	Tujuan	SSID WAP	Sumber	Tidak ada
1	0	SSID WAP	Sumber	Tujuan	Tidak ada
1	1	Penerima	Pemancar	Tujuan	Sumber

Menyiapkan jaringan wireless local area

Seperti yang kita lakukan dengan jaringan Ethernet, kita akan melihat pengaturan LAN nirkabel. Dalam hal ini, kami akan menguraikan contoh kami yang tercakup dalam Bagian sebelumnya, dengan perusahaan kami yang terdiri dari 40 karyawan, empat printer, satu server file, empat sakelar, dan satu router. Perusahaan telah memutuskan untuk menyediakan akses ke jaringan kabelnya melalui WiFi. Kami akan melihat langkah-langkah untuk mengatur ini.

Pertama, ingat dari contoh bahwa kantor karyawan kita meliputi dua lantai sebuah gedung. Kantor-kantor diberi jarak sedemikian rupa sehingga jarak antar pintu kira-kira 20 kaki. Hal pertama yang perlu kita perhatikan adalah topologi WLAN. Agar hemat biaya, kami tidak ingin membuat jaringan BSS individual, satu per kantor. Alasan pembatasan ini adalah karena memerlukan terlalu banyak WAP dan setiap WAP harus terhubung langsung ke jaringan kabel. Sebagai gantinya, kami akan membuat ESS yang terdiri dari banyak WAP. Kami akan memposisikan WAP untuk mencakup beberapa kantor. Meskipun kemungkinan pintu tertutup, kekuatan sinyal harus cukup untuk menutupi kantor terdekat. Kantor kami diatur di setiap sisi lorong, sehingga WAP yang ditempatkan di langit-langit di tengah lorong langsung di antara dua kantor yang berdekatan harus mencakup empat kantor yang berdekatan. Lihat sisi kiri Gambar 4.11. Dengan menempatkan WAP di lorong, akses di dalam kantor dengan pintu tertutup mungkin terganggu, tetapi harus tetap memiliki akses. Ini karena sinyal nirkabel dapat disiarkan melalui pintu atau dinding dalam bentuk yang terdegradasi. Jumlah degradasi bergantung pada bahan yang harus dilalui oleh sinyal (misalnya, kaca dan logam dapat memantulkan sebagian besar sinyal kembali ke kantor).



Gambar 4.11 Cakupan area untuk satu WAP.

Dengan 40 karyawan dan 40 kantor, ini akan membutuhkan total 10 WAP. Bisakah kita memotong ini? Ya, kita harus bisa. Dengan 40 kantor, setiap lantai ada 20 kantor, dan jika

kantor berada di kedua sisi lantai, maka terdapat 10 kantor di setiap sisi lorong setiap lantai. WAP tunggal yang terletak di pusat akan mencakup jarak melingkar 100 meter (300 kaki). Panjang ini akan mencakup 15 kantor; namun, karena liputannya melingkar, kantor terakhir di sepanjang lorong mungkin tidak menerima liputan penuh. Lihat sisi kanan Gambar 2.11 yang menunjukkan enam kantor di setiap sisi lantai (area cakupan tidak ditampilkan untuk menjaga agar ukuran gambar tidak terlalu besar).

Terlepas dari biaya yang dapat dihemat oleh satu WAP (atau bahkan dua WAP) per lantai, kami akan tetap menggunakan satu WAP per empat kantor (total 10 WAP), karena area jangkauan bukan satu-satunya masalah. Ingatlah bahwa WAP adalah bandwidth bersama. Jika 20 pengguna berbagi satu WAP, kinerja dapat menurun dengan sangat baik karena jumlah komunikasi yang digunakan pada satu waktu. Jadi, kami telah memilih jumlah WAP, dan berdasarkan sisi kiri Gambar 4.11, kami telah memutuskan posisinya. Kami khawatir bahwa terlalu banyak WAP dalam jarak dekat dapat menyebabkan interferensi co-channel; namun, keputusan untuk mengurangi jumlah WAP mungkin diperlukan. Karena jaringan kami relatif kecil, kami memutuskan untuk tidak membeli atau menggunakan pengontrol nirkabel.

Selanjutnya, kita harus menghubungkan WAP ke jaringan kabel kita. Kita dapat membangun ini dengan dua cara. Pertama, kita bisa menghubungkan setiap WAP ke sebuah router. Kelemahan dari solusi ini adalah organisasi memiliki satu router, dan kami harus membeli beberapa router atau menjalankan kabel dari setiap WAP ke satu router. Juga, ingat bahwa router tunggal kami memiliki empat koneksi ke dalamnya (empat sakelar) dan satu koneksi ke ISP kami. Sekarang, kami harus menambah jumlah koneksi ke router kami sebanyak 10. Jika router kami memiliki kurang dari 14 port, kami perlu mengganti router. Jadi, sebagai gantinya, kami akan membentuk sistem distribusi dari WAP kami dan hanya memiliki satu WAP di setiap lantai yang terhubung ke router. Dengan cara ini, pesan dari jaringan kabel ke perangkat nirkabel akan dikirim dari router ke salah satu dari dua WAP yang terhubung dengannya (berdasarkan lantai tempat perangkat tujuan berada) dan dari sana ke WAP lain dan ke nirkabel tujuan. perangkat.

Langkah kita selanjutnya adalah mengkonfigurasi WAP kita. Kami dapat mengatur WAP kami untuk mengeluarkan alamat IP ke klien kami (menggunakan DHCP) dan melakukan terjemahan alamat jaringan (NAT, tercakup dalam Bab 3). Karena ini adalah konsep untuk bab selanjutnya, kami tidak akan melakukannya dan sebagai gantinya mengandalkan router organisasi kami untuk menangani tugas ini. Sebagai gantinya, kami akan menerima konfigurasi WAP dasar.

Kami telah memilih untuk menggunakan WAP sederhana, yaitu WAP nonrouter. Ini sebenarnya agak rumit, karena saat ini kebanyakan WAP adalah kombinasi dari WAP dan router. Perangkat semacam itu akan memiliki alamat IP sendiri yang dikeluarkan untuknya, dan Anda dapat mengonfigurasinya melalui Internet dengan mereferensikan alamat IP tersebut. Dalam kasus kami, WAP tidak akan memiliki alamat IP, jadi, kami harus mengonfigurasinya dengan cara lain.

ita dapat melakukannya dengan langsung menghubungkan komputer ke WAP melalui port Ethernet yang terdapat di bagian belakang WAP (port ini akan digunakan untuk menghubungkan WAP ke router, jika kita ingin langsung menghubungkannya ke router).

Sebagian besar WAP mengizinkan konfigurasi baris perintah atau berbasis web. Kami akan membahas secara singkat hanya pendekatan berbasis web. Pendekatan baris perintah tersedia di sistem Unix/Linux dengan menggunakan file di subdirektori /etc/config.

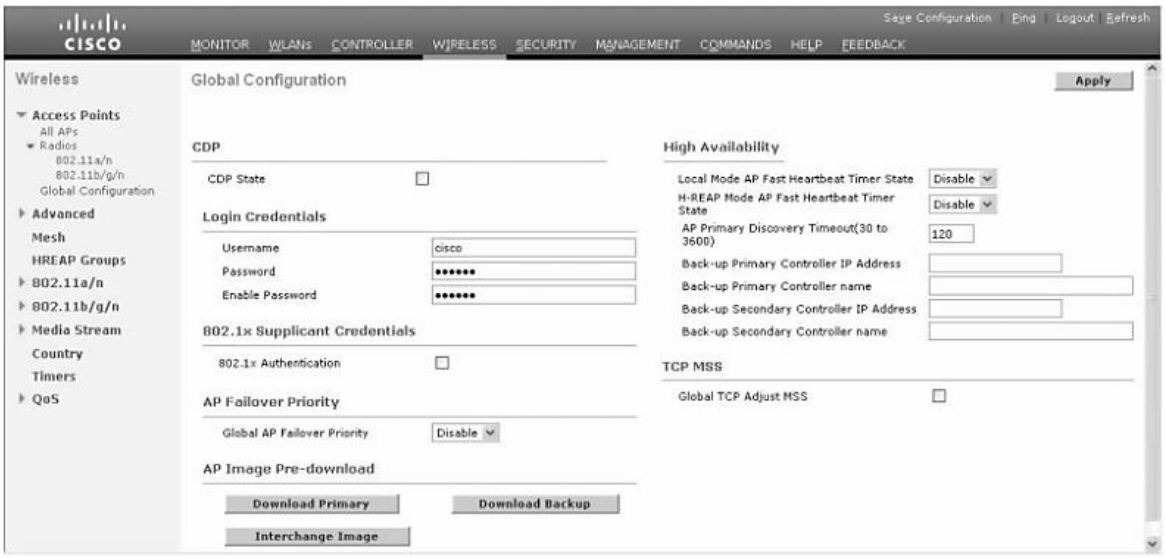
WAP akan datang dengan URL yang akan digunakan untuk konfigurasi. Setelah komputer Anda terhubung ke WAP, buka browser Anda dan masukkan URL tersebut. Sebagai contoh, WAP TP-LINK menggunakan salah satu alamat ini: 192.168.1.1, 192.168.0.1, dan tplinklogin.net. Saat terhubung ke situs web ini, Anda akan diminta untuk memberikan login dan kata sandi administrator. Informasi ini harus diberikan kepada Anda dalam kemasan dengan WAP.

Dari portal web, Anda sekarang akan menetapkan beberapa informasi: Alamat IP, Subnet Mask, SSID, wilayah, frekuensi, mode, saluran dan lebar saluran, mode siaran SSID, mode infrastruktur, dan keamanan. Pada Gambar 4.12, Anda dapat melihat halaman konfigurasi global portal web untuk mengonfigurasi CISCO WAP. Mari kita jelajahi beberapa detail dari apa yang harus Anda tentukan. Alamat IP tidak diperlukan untuk WAP nonrouter; namun, jika Anda menyediakannya, ini dapat membuat upaya untuk mengonfigurasi ulang perangkat menjadi lebih mudah. Subnet mask adalah subnet dari WLAN atau subnet yang sesuai dengan router untuk WLAN. SSID adalah nama alfanumerik untuk WLAN. Frekuensi akan menjadi 2,4 GHz atau 5 GHz, atau lainnya jika WAP mengizinkan frekuensi yang berbeda. Saluran dan lebar saluran memungkinkan Anda menentukan saluran mana yang dapat digunakan berdekatan dengan frekuensi. Ada 11 saluran yang tersedia untuk frekuensi tertentu. Mode siaran SSID memungkinkan Anda mematikan kemampuan WAP untuk menyiarkan bingkai suar. Mode infrastruktur menunjukkan apakah WAP merupakan bagian dari sistem distribusi atau jaringan ad hoc. Karena peran WAP adalah menjadi bagian dari sistem distribusi, Anda akan memilih mode infrastruktur (Anda dapat memilih ad hoc jika Anda menggunakan WAP sebagai router saja). Terakhir, mode keamanan menunjukkan apakah dan bentuk enkripsi apa yang akan Anda gunakan.

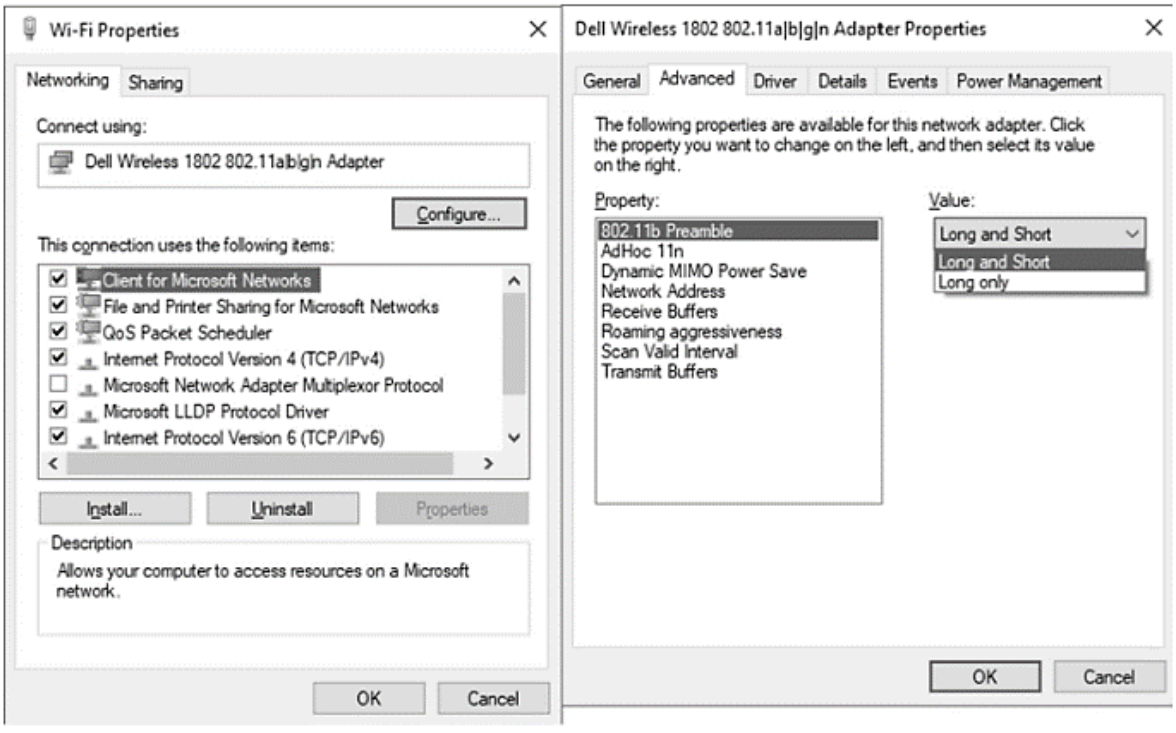
Langkah terakhir adalah mengonfigurasi perangkat nirkabel Anda untuk berkomunikasi dengan WLAN. Karena semua karyawan di organisasi fiktif kami memiliki komputer desktop sendiri, jika mereka juga ingin menggunakan WLAN, mereka harus mengganti NIC mereka dengan WNIC dan melakukan konfigurasi yang benar. Dengan WNIC, mereka masih dapat mengakses LAN kabel, sehingga pengguna memiliki pilihan untuk beralih antara LAN kabel dan WLAN. Di sini, kami memeriksa konfigurasi untuk hanya membuat jaringan nirkabel pada perangkat. Perlu diingat bahwa WNIC akan secara otomatis mencari WAP melalui bingkai suar, sehingga hanya sedikit yang harus dilakukan administrator untuk menjalin komunikasi dengan WLAN. Namun, setelah WNIC diinstal, beberapa konfigurasi minimal diperlukan.

Di Windows (misalnya, 7 atau 10), untuk menetapkan dan mengubah pengaturan konfigurasi pada WNIC Anda, gunakan fitur Jaringan dan Berbagi dari Panel Kontrol. Mendapatkan jendela properti WNIC Anda, Anda akan melihat daftar fitur, seperti yang ditunjukkan di sebelah kiri Gambar 4.13. Dalam hal ini, kita melihat bahwa koneksi nirkabel komputer melalui Kartu Setengah Mini Nirkabel DW1530. Memilih Configure… memungkinkan Anda untuk mengubah konfigurasi WNIC. Sisi kanan Gambar 4.13

menunjukkan bagian lanjutan dari konfigurasi, di mana Anda dapat, misalnya, menetapkan mode 802.11, lebar pita, mode infrastruktur, dan banyak aspek kartu lainnya.



Gambar 4.12 Mengkonfigurasi CISCO WAP.



Gambar 4.13 Mengkonfigurasi WNIC.

Teknologi terkait

Kami menyelesaikan pemeriksaan komunikasi nirkabel kami dengan melihat teknologi nirkabel terkait yang bukan WiFi. Di sini, kami akan berkonsentrasi pada jaringan ad hoc: Bluetooth dan komunikasi seluler.

Kami telah melihat jaringan nirkabel ad hoc, yang terkadang disebut sebagai WANET. WANET terdiri dari perangkat nirkabel tanpa WAP, sehingga perangkat nirkabel berkomunikasi secara langsung satu sama lain. Perangkat nirkabel dapat digunakan untuk meneruskan pesan yang tidak ditujukan untuk mereka untuk menjembatani jarak antara sumber dan node tujuan. Hal ini menambah beban pada perangkat nirkabel, tetapi keuntungan dari WANET adalah harganya lebih murah (tidak memerlukan jenis infrastruktur apa pun) dan dapat dikonfigurasi ulang secara dinamis. Namun, WANET hanyalah salah satu bentuk jaringan ad hoc. Secara umum, jaringan ad hoc adalah kumpulan perangkat apa pun yang berkomunikasi satu sama lain tanpa infrastruktur di tempat.

Ada dua bentuk jaringan ad hoc lain yang perlu disebutkan. Yang pertama dikenal sebagai mobile ad hoc network (MANET). Meskipun ini terdengar seperti WANET di mana setiap perangkat diperbolehkan untuk bergerak, perbedaannya adalah semua perangkat harus berfungsi sebagai router saat dipanggil. Artinya, perangkat apa pun di MANET mungkin memiliki tanggung jawab untuk menerima pesan dan meneruskannya ke node lain di jaringan. WANET, di sisi lain, umumnya memiliki beberapa poin tetap, setidaknya untuk saat ini, dan pesan biasanya disiarkan.

Ada beberapa jenis MANET, termasuk jaringan ad hoc kendaraan (VANET), yang merupakan jaringan nirkabel ad hoc yang mengizinkan kendaraan (mobil pintar) untuk berkomunikasi satu sama lain. Meskipun bentuk MANET ini sedang dalam tahap penelitian, kami berharap MANET menjadi komponen penting untuk mobil self-driving, dimana mobil di jalan dapat berkomunikasi satu sama lain untuk mempelajari niat masing-masing mobil (misalnya, untuk berpindah jalur dan untuk memperlambat). Varian dari VANET adalah MANET militer, yang memungkinkan kendaraan dan unit darat (pasukan) untuk berkomunikasi. Bentuk lain dari MANET adalah smart phone ad hoc network (SPAN). SPAN menambahkan kemampuan Bluetooth ke komunikasi.

Dua bentuk lain dari jaringan nirkabel ad hoc adalah jaringan mesh nirkabel dan jaring sensor nirkabel. Bentuk jaringan nirkabel ini berbeda dari WANET dan MANET karena masing-masing node yang membentuk jaringan tidak dimaksudkan untuk menjelajah. Di sisi lain, mereka tidak memiliki infrastruktur untuk menghubungkan mereka bersama, sehingga mereka berkomunikasi bersama melalui sinyal radio. Selain itu, jaringan ini diatur untuk menangani situasi di mana node ditambahkan atau dihapus secara acak.

Jaringan mesh nirkabel sering diterapkan pada situasi di mana jarak antar node jauh lebih besar daripada jarak antar perangkat di WANET. Misalnya, jaring nirkabel dapat dibentuk dari perangkat nirkabel, satu per rumah tangga, untuk memantau pengiriman daya. Dalam kasus seperti itu, setiap simpul akan berjarak beberapa ratus kaki dari simpul lainnya. Sebuah perusahaan listrik mungkin ingin menggunakan jaringan mesh untuk menentukan keadaan seperti brownouts dan pemadaman listrik dan permintaan daya meningkat atau menurun.

Jaringan sensor dapat berupa kabel atau nirkabel. Kami tertarik dengan bentuk nirkabel, karena dapat membentuk jaringan ad hoc. Perbedaan antara jaringan sensor nirkabel dan jaringan mesh nirkabel adalah bahwa node jaringan sensor terdiri dari dua bagian: sensor dan kemampuan untuk berkomunikasi. Dalam jaringan mesh, sebuah node memiliki kemampuan untuk memproses dan menyimpan informasi. Sensor mungkin unit yang

sangat sederhana, yang memiliki kemampuan untuk merasakan hanya satu atau beberapa aspek dari lingkungannya (misalnya suhu dan tekanan). Selain itu, ada sedikit kebutuhan sensor untuk menerima pesan selain yang meminta pengakuan (apakah Anda tersedia?) atau meminta pengiriman data.

Ada banyak aplikasi untuk jaringan sensor. Di antara yang telah menemukan kesuksesan termasuk pemantauan perawatan kesehatan, di mana sensor adalah perangkat yang dapat dipakai atau bahkan perangkat yang ditanamkan di dalam tubuh manusia; pemantauan polusi udara; pemantauan kualitas air; dan pemantauan pembangkit listrik, untuk beberapa nama.

Anda mungkin mendapat kesan bahwa semua bentuk jaringan nirkabel ini serupa, dan memang demikian. Namun, perbedaannya menentukan kebutuhan akan protokol yang berbeda. Di antara masalah yang harus ditangani oleh protokol adalah paket yang dijatuhkan, perutean, penundaan, penerimaan paket yang tidak sesuai pesanan, skalabilitas, dan keamanan. Selain itu, jaringan nirkabel ini harus mampu menangani node lintas platform dan dampak lingkungan pada penyiaran pesan.

Bagian 2.3 telah membahas komunikasi nirkabel untuk membentuk jaringan. Bluetooth menyediakan banyak fungsi yang sama tetapi telah menemukan penggunaan yang berbeda. Bluetooth menggunakan sinyal radio dalam rentang frekuensi yang sama (2,4 GHz) seperti perangkat dalam jaringan nirkabel tetapi umumnya memiliki kecepatan bit dan jarak yang jauh berkurang. Faktanya, Bluetooth umumnya mentransmisikan dengan kecepatan sekitar 2 Mbps. Keuntungan utama Bluetooth adalah jauh lebih murah dan membutuhkan konsumsi daya yang jauh lebih sedikit daripada yang dibutuhkan perangkat untuk jaringan nirkabel.

Pengurangan konsumsi dayalah yang menjadi perbedaan terbesar antara perangkat Bluetooth dan perangkat non-Bluetooth. Segala bentuk perangkat nirkabel akan memiliki baterai untuk memberi daya pada perangkat saat tidak dicolokkan. Namun, dengan perangkat Bluetooth, perangkat tersebut tidak dimaksudkan untuk dicolokkan atau baterai diisi ulang, tidak seperti ponsel pintar atau laptop, yang dapat diisi ulang. sering diperlukan (mungkin setiap hari). Sebaliknya, baterai perangkat Bluetooth dapat diganti setelah habis. Kami menemukan Bluetooth digunakan di perangkat seperti headset nirkabel, keyboard nirkabel, mouse nirkabel, dan perangkat Sistem Pemosisian Global (GPS) nirkabel. Dalam semua kasus, pengurangan konsumsi daya membuatnya menarik, dan karena tidak satu pun dari perangkat ini yang mentransfer banyak data, pengurangan laju bit tidak menjadi masalah. Selain itu, meskipun WiFi umumnya berkomunikasi pada jarak hingga 100 meter, Bluetooth hanya ditujukan untuk beberapa meter, hingga jangkauan maksimal 30 meter.

Jaringan seluler adalah kumpulan stasiun pangkalan yang dikenal sebagai menara seluler dan tulang punggung yang menghubungkannya ke jaringan telepon. Jaringan ini mengakomodasi komunikasi telepon seluler. Karena ponsel sering digunakan saat orang sedang bergerak, jaringan seluler mendukung roaming dengan mengirimkan sinyal dari satu menara seluler terdekat ke menara lainnya.

Teknologi radio yang digunakan oleh ponsel hampir sama dengan yang kita lihat pada WiFi dan Bluetooth. Perbedaan utama kemudian adalah bahwa teknologi ponsel

menggunakan protokol untuk menghubungi menara seluler terdekat dan beralih dari satu menara ke menara lainnya saat ponsel menjelajah. Pemilihan menara seluler dilakukan secara otomatis oleh ponsel Anda, tidak seperti WiFi, yang menyerahkan pengaitan perangkat ke WAP kepada pengguna.

Nama seluler berasal dari area (misalnya kota) yang dibagi menjadi kisi-kisi, atau sekelompok sel, dengan menara dimasukkan ke dalam setiap posisi kisi. Sebuah sel kira-kira berukuran 10 mil persegi; namun, area cakupan berbentuk lingkaran (biasanya, sel dipetakan dengan menggunakan bentuk heksagonal, sehingga cakupan tumpang tindih di antara sel yang berdekatan). Operator telepon seluler biasanya memiliki lebih dari 800 frekuensi radio yang berbeda. Ponsel beroperasi dalam mode dupleks penuh, membawa sinyal keluar melalui satu frekuensi dan sinyal masuk melalui frekuensi kedua. Ponsel, seperti perangkat nirkabel lainnya, memiliki antena (hampir selalu di dalam unit).

Teknologi ponsel telah berkembang dengan meningkatkan kecepatan transfer, sehingga jenis informasi yang dapat disiarkan dan diterima ponsel telah meningkat. Kecepatan transfer yang ditingkatkan ini telah menghasilkan ponsel yang lebih baik yang dapat melakukan lebih dari sekadar berfungsi sebagai telepon. Tabel 2.5 membandingkan berbagai generasi kemampuan ponsel.

Ponsel beroperasi dengan cara yang mirip dengan perangkat nirkabel melalui WiFi, dan tentu saja, ponsel pintar memiliki kemampuan komunikasi yang sama melalui WiFi. Namun, saat menggunakan ponsel untuk menelepon seseorang melalui menara ponsel, ponsel umumnya menggunakan frekuensi yang lebih rendah (800–2100 MHz). Ponsel dilengkapi dengan kartu Subscriber Identity Module (SIM). Kartu ini berisi chip kecil yang digunakan untuk mengidentifikasi dan mengautentikasi pengguna ponsel. Karena kartu ini dapat dilepas, orang yang mengganti ponsel dapat menukar kartu SIM untuk mempertahankan identitasnya di ponsel baru.

Ponsel memiliki kemampuan seperti mengambil gambar dan video melalui kamera built-in, menampilkan gambar/video dan audio, menjalankan berbagai aplikasi seperti permainan atau kalender, ruang penyimpanan (biasanya dalam bentuk memori flash), dan memprogram nada dering berdasarkan nomor telepon yang masuk. Dengan kompatibilitas WiFi, sebagian besar pengguna dapat mengunggah dan mengunduh konten dari dan ke ponsel mereka. Saat ini, ponsel disebut ponsel pintar karena memiliki prosesor dan sistem operasi yang canggih. Ponsel pintar menggunakan mikroprosesor ARM, yang memiliki kemampuan seperti prosesor komputer desktop 10–15 tahun yang lalu. Sebagian besar mikroprosesor ini hanya mengizinkan satu aplikasi untuk dijalankan pada satu waktu (selain dari panggilan telepon aktif), tetapi ponsel pintar yang lebih baru menyediakan beberapa aspek multitasking. Ruang penyimpanan bervariasi tergantung pada jenis memori yang digunakan dan biaya telepon. Sistem operasi ponsel pintar yang paling populer adalah iOS (Apple) dan Android (suatu bentuk Linux), ditemukan di banyak platform berbeda.

Tabel 4.5 Generasi Ponsel

Generasi	Bertahun-tahun	Kecepatan Bit	Fitur umum
1	Sebelum tahun 1991	1 Kbps	Sinyal analog, hanya suara

2	1991–2000	14–100 Kbps	Komunikasi digital, SMS
3	2001–2009	384 Kbps–30 Mbps	Akses Internet sederhana, video
4	Sejak 2009	35–50 Mbps (unggah), 360 Mbps (unduh)	Ponsel pintar digunakan sebagai perangkat nirkabel untuk akses WiFi

Pada tahun 2011, sekitar 45% populasi dunia memiliki akses ke setidaknya jaringan seluler 3G, sedangkan banyak negara Dunia Pertama menawarkan akses 4G ke sebagian besar populasi mereka. Selain itu, dengan tersedianya telepon pintar yang relatif murah, sebagian besar dunia kini mengakses Internet melalui telepon seluler. Faktanya, di banyak negara, sebagian besar penduduknya mengakses Internet hanya melalui ponsel. Ini secara drastis meningkatkan akses Internet ke 7 miliar orang di dunia. Dari sekitar 3,2 miliar pengguna internet di seluruh dunia, diperkirakan lebih dari setengahnya mengakses internet hanya melalui ponsel, sedangkan sebagian besar pengguna sisanya menggunakan ponsel sebagai salah satu perangkatnya untuk mengakses internet.

Terlepas dari manfaat teknologi ponsel (akses Internet seluler dengan biaya yang sangat berkurang, komputasi seluler, dan sebagainya), ada kekhawatiran serius dengan peningkatan jumlah ponsel yang dibuat dan dijual. Ponsel, seperti perangkat komputasi apa pun, menggunakan papan sirkuit tercetak, yang menggunakan berbagai elemen beracun untuk diproduksi. Kebanyakan orang tidak membuang komputer mereka setelah satu atau dua tahun penggunaan, sedangkan ponsel dibuang pada tingkat yang jauh lebih tinggi. Beberapa orang mengganti ponsel mereka setiap tahun. Tanpa daur ulang dan pembuangan yang tepat dari beberapa komponennya, ponsel bekas menimbulkan masalah lingkungan yang dapat meracuni area di sekitar pembuangannya, termasuk kerusakan pada tabel air bawah tanah.

4.5 MENGAMANKAN JARINGAN LOKAL AREA

Saat melihat LAN dan WLAN, kami tidak melihat detail untuk memastikan keamanan di jaringan. Tanpa keamanan, jaringan terbuka untuk diserang, dan pesan pengguna terbuka. Jika informasi rahasia sedang dikirim melalui jaringan (misalnya, kata sandi dan nomor kartu kredit), tidak adanya keamanan merupakan ancaman besar bagi pengguna dan akan membuat pengguna enggan menggunakan jaringan itu. Mekanisme keamanan berbasis protokol untuk LAN dan WLAN sangat berbeda. Kami akan mengeksplorasi beberapa mekanisme tersebut di sini dan menyelesaikan bagian ini dengan membahas secara singkat salah satu jenis mekanisme keamanan yang dibutuhkan oleh LAN dan WAN: keamanan fisik.

Keamanan Ethernet adalah sesuatu dari sebuah oxymoron. Seperti yang akan kita jelajahi di Bab 3, ketika kita menyelidiki semua lapisan tumpukan protokol TCP/IP, sebagian besar mekanisme keamanan jaringan terjadi pada lapisan yang lebih tinggi daripada lapisan OSI 1 dan 2, yang merupakan lapisan yang diimplementasikan oleh Ethernet. Oleh karena itu, terdapat sedikit dukungan pada physical layer atau data link layer yang dapat mendukung keamanan. Kami akan mengeksplorasi beberapa masalah dan solusi di sini, sambil meninggalkan sebagian besar diskusi ini untuk bab selanjutnya.

Mari kita ajukan masalahnya seperti ini: apa yang tidak aman dari Ethernet? Dari perspektif lapisan fisik, Ethernet menawarkan sarana bagi sinyal untuk berpindah dari satu lokasi ke lokasi lain melalui kabel. Apa pun yang melewati kabel dapat diambil di sepanjang jalan dengan menggunakan salah satu dari beberapa pendekatan berbeda. Misalnya, pertimbangkan jaringan bus yang sudah ketinggalan zaman. Perangkat terhubung melalui konektor-T. Sebagai karyawan yang tidak memiliki akses ke jaringan tertentu, menambahkan konektor-T di suatu tempat di segmen tersebut dan memasang perangkat (misalnya, komputer) ke konektor-T tersebut memungkinkan karyawan tersebut mencegat pesan yang ditujukan untuk orang lain di jaringan. Jelas, untuk melakukan ini, karyawan memerlukan akses ke kabel yang membentuk jaringan.

Dengan jaringan bintang, sebuah saklar mengirimkan pesan ke komputer tujuan dengan menggunakan alamat MAC tujuan. Seperti yang kita lihat sebelumnya di bab ini, alamat MAC unik untuk setiap perangkat (kecuali alamat broadcast). Namun, itu tidak menghalangi seseorang untuk dapat memperoleh pesan yang ditujukan untuk alamat MAC lain. Dua pendekatan adalah spoofing MAC, di mana seseorang dapat mengubah alamat MAC NIC-nya ke alamat lain dan dengan demikian menipu sakelar, dan keracunan ARP, di mana tabel MAC yang dikelola oleh sakelar itu sendiri diubah sehingga sakelar percaya bahwa a Alamat MAC milik tujuan yang salah. Kedua teknik ini dapat diterapkan oleh seseorang internal organisasi. Namun, seperti yang kami sebutkan dengan konfigurasi WAP, sakelar jaringan telah dikonfigurasi sebelumnya, dan merupakan kesalahan umum bagi organisasi untuk tidak mereset kata sandi sakelar dari defaultnya. Dengan demikian, ada kemungkinan bahwa switch jaringan dapat diserang dari luar organisasi, dimana keracunan ARP dapat terjadi.

Cara lain untuk membobol jaringan bintang adalah dengan menghubungkan perangkat lain secara fisik ke sakelar. MAC spoofing kemudian dapat diterapkan, atau perangkat baru dapat digunakan untuk melakukan pengintaian di jaringan untuk serangan selanjutnya. Ada juga kelemahan pada lapisan data link yang dapat dimanfaatkan untuk menyebabkan serangan denial of service, misalnya dengan membanjiri switch dengan banyak pesan.

Ada solusi sederhana untuk semua masalah ini. Pertama, pastikan jaringan aman secara fisik dengan tidak mengizinkan siapa pun memiliki akses ke sakelar. Kunci sakelar di lemari atau ruangan yang kuncinya hanya dimiliki oleh administrator. Selanjutnya, ubah kata sandi default pada sakelar. Ada juga program deteksi intrusi jaringan yang dapat dijalankan untuk menganalisis penggunaan sakelar untuk melihat apakah ada perilaku yang tidak normal. Kami membatasi diskusi kami di sini, karena sebagian besar metode serangan dan keamanan terjadi pada lapisan protokol yang lebih tinggi, seperti melalui penggunaan enkripsi. Selain itu, meskipun terdapat banyak praktik terbaik keamanan lapis 2, pembahasan yang lebih lengkap berada di luar cakupan buku ini.

Jaringan nirkabel memiliki dua kategori kelemahan yang dapat dieksploitasi. Pertama, WAP menyiarkan pesan. Siapa pun di area jangkauan dapat mencegat siaran dan mendapatkan salinan pesan tersebut. Kedua, WAP dapat dikonfigurasi ulang dengan mudah, mengingat alamat IP-nya (jika ada yang ditetapkan). Solusi untuk masalah pertama adalah menggunakan enkripsi pada semua pesan yang dikirimkan antara perangkat nirkabel dan WAP WLAN. Ada tiga teknologi enkripsi yang umum digunakan. Kami akan segera membahas

semuanya. Masalah kedua mudah ditangani dengan memastikan bahwa WAP dikunci. Ini dilakukan dengan mengubah kata sandi default. Sebagian besar WAP dikirimkan dengan kata sandi seperti admin. Mengubah kata sandi memberikan perlindungan dari seseorang yang mencoba mengubah konfigurasi WAP Anda (ingat salah satu pengaturan konfigurasi adalah mengatur perlindungan; jika seseorang dapat mengkonfigurasi ulang WAP Anda, dia dapat mengubah pengaturan ini dengan mematikan segala bentuk enkripsi!).

Tiga bentuk enkripsi nirkabel yang umum digunakan. Ini dikenal sebagai Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), dan Wi-Fi Protected Access II (WPA2). Wired Equivalent Privacy adalah bentuk asli dari teknologi enkripsi yang diterapkan pada nirkabel. Namanya berasal dari gagasan bahwa perlindungan akan sama dengan perlindungan enkripsi yang ditawarkan dalam jaringan kabel. Namun, WEP ditemukan memiliki beberapa kelemahan, dan bentuk enkripsi alternatif, WPA, diperkenalkan. Wi-Fi Protected Access memiliki kelemahannya sendiri, sebagian besar karena itu adalah implementasi terburu-buru yang sebagian didasarkan pada WEP dan membawa salah satu kekurangannya. WPA2 adalah yang terbaik dari ketiganya dan yang direkomendasikan untuk enkripsi nirkabel apa pun. Terlepas dari usia dan masalahnya, WEP terus digunakan dalam beberapa kasus, sebagian karena setiap WAP nirkabel atau router yang memiliki ketiga formulir mencantumkannya dengan WEP terlebih dahulu, dan begitu pula formulir yang dipilih oleh banyak administrator hanya karena ini yang pertama. dalam daftar. Mari kita lihat secara singkat ketiganya dan lihat mengapa WEP dan WPA dianggap cacat.

Diusulkan dengan standar asli 1997 802.11, WEP awalnya adalah algoritma enkripsi 64-bit, di mana kunci 64-bit adalah kombinasi dari kunci pilihan pengguna 40-bit dan vektor inisialisasi 24-bit. Algoritma enkripsi yang diterapkan disebut RC4, atau Rivest Cipher 4. Dengan 64 bit, pesan dibagi menjadi blok 64 bit, di mana setiap blok 64 bit pesan eksklusif OR (XOR) dengan kunci untuk menyediakan pesan terenkripsi . Operasi XOR bekerja secara bit-wise, sehingga bit pertama dari kunci dan pesan di-XOR bersama untuk menghasilkan bit pertama dari pesan terenkripsi. Kemudian, bit kedua dari kunci dan pesan di-XOR bersama untuk menghasilkan bit kedua dari pesan terenkripsi. Dengan demikian, enkripsi ditangani oleh 64 operasi XOR. Operasi XOR menghasilkan 1 jika dua bit berbeda dan 0 sebaliknya. Misalnya, 1 XOR 0 adalah 1 dan 1 XOR 1 adalah 0. Mari kita tinjau secara singkat dua byte yang di-XOR secara bersamaan. Urutan 00001111 XOR 10101010 menghasilkan byte 10100101. Dekripsi ditangani dengan mengambil kunci asli dan pesan terenkripsi dan meng-XOR-kannya secara bit-wise. Kami melihat ini sebagai berikut:

Kunci Pesan Asli XOR = Pesan Terenkripsi
Kunci Pesan Terenkripsi XOR = Pesan Asli

Ada beberapa masalah dengan WEP. Yang pertama adalah bahwa algoritme enkripsi 64-bit menjadi cukup mudah diretas dengan komputer modern. Pencarian brute-force untuk mengidentifikasi kunci 64 bit dapat dilakukan hanya dalam beberapa menit. Pemerintah AS telah membatasi teknologi enkripsi untuk dibatasi hingga 40-bit jika teknologi itu akan diterapkan di luar Amerika Serikat. RC4 memenuhi syarat karena 40 dari 64 bit disediakan oleh

pengguna. Pembatasan ini dikurangi menjadi 56 bit pada tahun 1996 dan semakin berkurang sekitar tahun 1999. Batasan ini dihapus seluruhnya pada tahun 2000 untuk perangkat lunak sumber terbuka, yang algoritme enkripsinya dapat dipelajari dan ditingkatkan. Masih ada batasan untuk batasan 64-bit untuk teknologi enkripsi yang diekspor ke negara nakal dan organisasi teroris. Pada, atau sekitar tahun 1999, WEP diperluas ke kunci 128-bit, seperti yang sekarang diizinkan, untuk mengimbangi risiko keamanan kunci 64-bit. Kelemahan lain dengan WEP adalah dengan penggunaan XOR untuk enkripsi. Pertimbangkan untuk menerima dua pesan terenkripsi dengan kunci yang sama. Seperti yang kita lihat sebelumnya, XOR pesan dengan kunci memberi kita pesan terenkripsi, dan XOR pesan terenkripsi dengan kunci memberi kita pesan asli. Jika dua pesan dienkripsi menggunakan kunci yang sama, kita dapat bekerja mundur dari pesan terenkripsi ini untuk mendapatkan kunci melalui operasi XOR. Ini membutuhkan beberapa analisis statistik dari hasil, tetapi bekerja dengan cara ini, kuncinya dapat diperoleh tanpa banyak usaha.

Ada juga kekurangan pada pembuatan vektor inisialisasi. Nilai ini dikenal sebagai nomor pseudorandom, tetapi pembuatan nomornya agak didasarkan pada sandi yang digunakan. Karena WEP menggunakan RC4, semua vektor inisialisasi akan dihasilkan menggunakan teknik yang sama. Ini membuat nilai pseudorandom sebagian dapat diprediksi. Dalam kunci 64-bit, vektor inisialisasi 24-bit juga ukurannya terlalu terbatas untuk banyak digunakan. Misalnya, kami dijamin bahwa vektor inisialisasi 24-bit yang sama akan dihasilkan lagi dengan pesan baru dalam waktu dekat (penelitian menunjukkan bahwa diperlukan sedikitnya 5000 pesan sebelum vektor inisialisasi 24-bit yang sama digunakan kembali) .

Kekurangan lainnya adalah WEP tidak menggunakan nilai nonce. Nilai nonce adalah nilai yang dihasilkan secara acak satu kali yang dapat membantu pengacakan kunci lebih lanjut. Sebaliknya, WEP menerapkan kunci yang sama (apakah 64-bit, 128-bit, atau bahkan 256-bit) untuk semua urutan pesan 64-bit/128-bit/256-bit. Pengulangan kunci yang sama ini membuat pesan terenkripsi WEP lebih mudah diretas. Bahkan, kini tersedia peranti lunak yang dapat memecahkan kata sandi WEP hanya dalam beberapa menit atau kurang, tergantung pada seberapa banyak lalu lintas pesan yang dicegat.

Terakhir, WEP menggunakan CRC 32-bit untuk integritas data. Nilai CRC ini digunakan untuk mengkonfirmasi bahwa data tidak diubah. Nilai CRC 32-bit dihasilkan melalui komputasi linier sederhana sehingga dapat dibuat ulang dengan mudah. Misalnya, bayangkan pesan asli menghasilkan nilai CRC 12345. Kita ingin mengganti pesan dengan pesan kita sendiri. Pesan kita menghasilkan nilai CRC 9876. Kita dapat membuat perbedaan dengan menambahkan beberapa bit padding ke pesan kita, selama padding kita menghasilkan nilai CRC 12345 – 9876 = 2469.

WPA dirilis pada tahun 2003 untuk menggantikan WEP, dengan WEP ditinggalkan mulai tahun 2004. Perbedaan utama antara WPA dan WEP adalah penggunaan vektor inisialisasi satu kali yang dihasilkan dengan menggunakan Protokol Integritas Kunci Temporal (TKIP). Vektor ini dihasilkan bukan per pesan tetapi per paket. Dengan cara ini, menganalisis pesan untuk satu kunci tidak mungkin karena, pada kenyataannya, pesan itu dihasilkan dengan menggunakan beberapa kunci yang berbeda. WPA juga memperpanjang panjang kunci dari panjang 64 dan 128-bit WEP menjadi 256 bit.

Kunci yang dihasilkan TKIP berfungsi seperti nilai nonce karena merupakan kunci satu kali. Jika penyadap memperoleh pesan terenkripsi, mendekripsinya memerlukan strategi brute-force pada setiap paket. Tidak ada petunjuk yang dapat ditemukan dalam mengerjakan paket-paket yang digabungkan. TKIP juga menerapkan bentuk integritas data yang lebih canggih. Alih-alih nilai CRC 32-bit, ia menggunakan Pemeriksaan Integritas Pesan 64-bit (dikenal sebagai MICHAEL).

Sayangnya, WPA memiliki kekurangannya sendiri, yang paling signifikan adalah bagaimana WPA dirilis. Sebagai langkah cepat untuk mengatasi kekurangan yang diketahui di WEP, WPA sebagian besar dirilis sebagai solusi firmware yang ditangkap dalam sebuah chip yang kemudian dapat ditambahkan ke router dan WAP yang sudah ada. Pembuatan kunci TKIP mendaur ulang beberapa strategi yang sama seperti bagaimana WEP menghasilkan vektor inisialisasi. Ini mengarah ke kunci yang agak dapat diprediksi. Selain itu, karena algoritme tersedia di firmware, algoritme ini dapat dicoba, dalam upaya untuk lebih memahami bagaimana kunci dibuat. Dengan WPA2, pendekatan berbeda digunakan untuk menghasilkan kunci satu kali. Oleh karena itu, WPA2 telah tersedia untuk menggantikan WPA dan WEP. Di WPA2, daripada TKIP, pendekatan yang disebut Penghitung dengan Protokol CBC-MAC (CCMP) digunakan, di mana CBC-MAC adalah singkatan dari Kode Otentikasi Pesan Rantai Blok Cipher. CCMP menggunakan algoritma enkripsi AES dengan kunci 128-bit dan MICHAEL untuk pemeriksaan integritas. Meskipun AES memiliki kekurangan yang lebih sedikit daripada TKIP dan WEP, masih ada risiko keamanan yang diketahui dalam menggunakan WPA2. Ini terutama ketika kata sandi enkripsi yang lemah diberikan oleh pengguna atau jika seseorang telah memiliki akses ke jaringan aman untuk bekerja dari dalam. Jadi, meskipun WPA2 adalah pilihan yang baik untuk jaringan rumah, untuk mengamankan LAN nirkabel dari organisasi yang lebih besar, keamanan tambahan mungkin diperlukan. Satu komentar terakhir tentang WPA2 adalah bahwa beberapa teknologi lama mungkin tidak mampu mendukungnya. Dalam kasus seperti itu, pemilik perangkat keras harus mempertimbangkan risiko keamanan dan biaya penggantian atau peningkatan perangkat keras.

Untuk mengatur bentuk keamanan yang sesuai, Anda mengkonfigurasi WAP Anda. Akan selalu ada pengaturan keamanan nirkabel. Anda mungkin hanya menemukan sedikit pilihan, tergantung pada jenis WAP yang Anda gunakan. Tabel 4.6 memberikan perbandingan yang bermanfaat dari sebagian besar pengaturan yang akan Anda temukan.

Ada kelemahan lain yang bisa dimanfaatkan dalam WLAN. Salah satunya dikenal sebagai jalur akses nakal. Ingatlah bahwa WAP dapat menyiarkan ke WAP lain. Setiap WAP dikonfigurasi, sehingga setiap WAP di WLAN organisasi dapat diatur untuk mengenkripsi pesan saat disiarkan ke klien mereka. Namun, WAP jahat mungkin tidak dikonfigurasi dengan cara ini. Menyisipkan jalur akses palsu adalah mudah jika WAP ingin berkomunikasi dengan WAP lain.

Tersedia peranti lunak yang akan menganalisis pesan WAP untuk setiap WAP nakal di area tersebut. Area cakupan yang lebih besar dari yang diperlukan dapat membantu seseorang dengan WAP nakal. Jadi, strategi lainnya adalah membatasi area jangkauan ke ruang kantor organisasi. Ingat cakupan (karena berbentuk lingkaran) mencakup area di luar ruang kantor.

Terakhir, meskipun mengamankan sistem Anda dari penyadapan atau akses tidak sah, cara lain untuk mengamankan jaringan Anda, baik kabel maupun nirkabel, adalah melalui keamanan fisik. Menambahkan titik akses jahat ke WLAN dan menyambungkan perangkat ke sakelar adalah dua alasan mengapa keamanan fisik sangat penting. Alasan lain adalah untuk memberikan tindakan pencegahan terhadap pencurian atau kerusakan berbahaya. Hal ini terutama berlaku jika ada bagian dari jaringan yang dapat diakses secara fisik oleh non-karyawan. LAN kampus universitas, misalnya, sangat rentan terhadap pencurian dan kerusakan. Tindakan pencegahan keselamatan sederhana berguna dengan biaya rendah. Ini termasuk yang berikut:

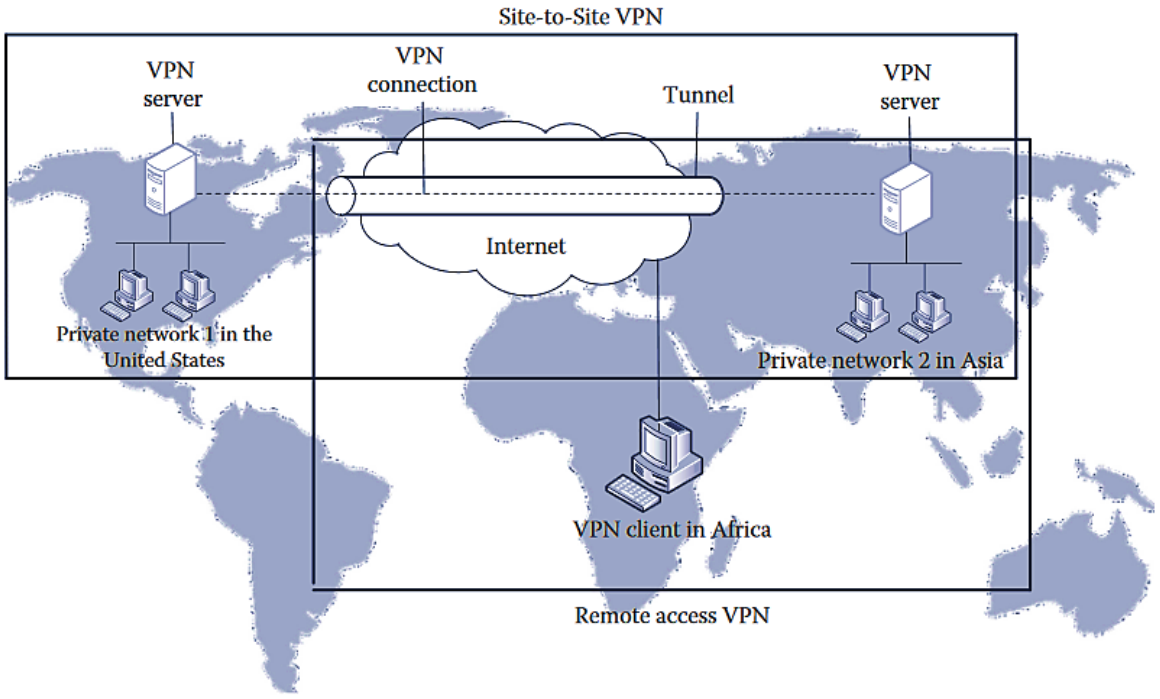
- Seseorang untuk memantau peralatan (misalnya, monitor laboratorium)
- Mengunci perangkat keras (untuk mencegah perangkat keras diambil)
- Kamera ditempatkan di dekat atau di sekitar peralatan (meskipun tidak sedang merekam)
- Transponder GPS di dalam peralatan portabel (untuk pelacakan)
- BitLocker atau enkripsi disk serupa

Tabel 4.7 Pilihan Keamanan Nirkabel'

Jenis	Deskripsi Khasiat	Panjang Kunci (dalam Karakter ASCII)
WEP	Tidak digunakan lagi dan tidak aman	10 karakter (kunci 40-bit) atau 26 karakter (kunci 128-bit)
WPA pribadi	Lebih baik dari WEP; tidak boleh digunakan jika WPA2 tersedia	8–63 karakter
perusahaan WPA	Lebih baik dari WPA pribadi tetapi membutuhkan server radius	8–63 karakter
pribadi WPA2	Lebih baik dari WPA pribadi	8–63 karakter
WPA 2 perusahaan	Lebih baik dari WPA2 pribadi tetapi membutuhkan server radius	8–63 karakter
Mode campuran WPA/WPA2	Dapat menggunakan salah satu algoritma	8–63 karakter

JARINGAN SWASTA VIRTUAL

Pandangan singkat tentang jaringan pribadi virtual (VPN). Jaringan pribadi adalah jaringan yang didedikasikan untuk organisasi tertentu. VPN adalah perpanjangan dari jaringan pribadi di jaringan publik. Ini hampir pribadi dan publik secara fisik. Teknologi VPN membangun koneksi virtual (koneksi logis) melalui jaringan publik, seperti Internet, untuk menghubungkan jaringan pribadi dari suatu organisasi di wilayah geografis yang berbeda. Teknologi VPN juga mengizinkan koneksi antara klien jarak jauh dan jaringan pribadi suatu organisasi.



Gambar 4.13 Arsitektur VPN.

Rute aktif adalah sebagai berikut:

Tujuan Jaringan	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.20	25
10.0.0.0	255.0.0.0	On-link	10.150.128.58	257
10.0.0.0	255.0.0.0	10.0.0.1	10.150.128.58	2

Ada dua jenis arsitektur VPN. Yang pertama adalah VPN akses jarak jauh. Dalam bentuk VPN ini, klien jarak jauh dapat terhubung ke jaringan pribadi organisasi melalui jaringan publik. Yang kedua adalah VPN situs-ke-situs. Dalam bentuk VPN ini, dua jaringan pribadi terhubung bersama melalui jaringan publik. Kedua jenis arsitektur diilustrasikan pada Gambar 4.14.

Dari Gambar 4.14, kami melihat klien di Afrika yang menggunakan Internet untuk terhubung ke VPN organisasi. Dalam contoh, klien jarak jauh di Afrika memulai koneksi virtual ke server VPN dari jaringan pribadi di Asia. Otentikasi diperlukan untuk mendapatkan akses ke sumber daya internal jaringan pribadi. Server VPN melakukan otentikasi. Jaringan pribadi menggunakan alamat IP pribadinya sendiri. Server VPN memberikan alamat IP pribadi ke klien jarak jauh. Tabel perutean pada klien jarak jauh diubah, sehingga lalu lintas dialihkan melalui koneksi virtual. Misalnya, PC Windows 10 jarak jauh menjalankan perangkat lunak klien Cisco VPN untuk membuat sambungan ke server VPN. Setelah koneksi dibuat, PC mendapatkan alamat IP pribadi baru, 10.150.128.58. Entri perutean baru ditambahkan ke tabel rute PC.

Arsitektur VPN site-to-site juga ditunjukkan pada gambar, di mana dua jaringan pribadi dari suatu organisasi dihubungkan bersama. Dalam contoh, organisasi memiliki satu situs di Asia dan satu lagi di Amerika Serikat. Setiap situs memiliki LAN sendiri. Dengan VPN yang menghubungkan LAN, pengguna salah satu LAN dapat mengakses sumber daya LAN lain,

seolah-olah mereka adalah bagian dari LAN-nya. Tanpa VPN, organisasi perlu menerapkan jalur fisik khusus untuk menghubungkan dua jaringan pribadi, yang sangat mahal.

Mari kita pertimbangkan sebuah contoh untuk mengilustrasikan biaya menghubungkan jaringan pribadi bersama tanpa menggunakan VPN. Sebuah organisasi memiliki n jaringan pribadi di wilayah geografis yang berbeda. Untuk menghubungkan mereka dalam topologi mesh penuh, kita membutuhkan n * (n – 1)/2 garis fisik. Untuk dua lokasi, hanya diperlukan 1 baris, dan untuk tiga lokasi, diperlukan tiga baris. Namun, untuk 4 situs, diperlukan 6 baris, dan untuk 5 situs, kami membutuhkan 10 baris. Pertumbuhan lebih besar saat n meningkat; misalnya, 10 situs berbeda akan membutuhkan 45 baris. Ingatlah bahwa situs-situs ini secara geografis berjauhan satu sama lain, sehingga garis-garisnya mungkin sepanjang ratusan hingga ribuan mil.

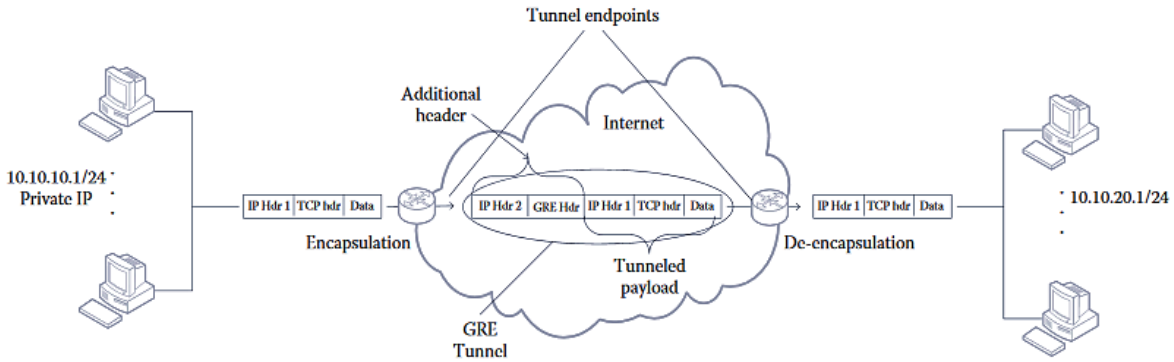
Berbeda dengan menghubungkan secara fisik dua atau lebih jaringan pribadi secara bersamaan, VPN menggunakan saluran bersama (mis., Internet). Karena organisasi hampir pasti memiliki akses ke Internet, membuat VPN melalui Internet hampir tidak memerlukan biaya. Di sisi lain, kami ingin komunikasi antara jaringan pribadi menjadi aman. Jika dua situs terhubung melalui jalur fisik khusus, keamanan tambahan tidak diperlukan. Namun, tidak demikian jika menggunakan VPN.

Untuk membuat koneksi VPN pribadi, VPN menggunakan teknologi tunneling untuk mengamankan komunikasi data antara dua situs melalui Internet. Tunneling adalah teknologi untuk mengenkapsulasi satu jenis paket protokol dalam jenis paket protokol lainnya. Misalnya, kita dapat mengenkapsulasi paket IPv6 dalam paket IPv4 dan mengirimkannya melalui jaringan IPv4, dalam hal ini disebut tunneling IPv6 melalui jaringan IPv4. Gambar 4.15 menunjukkan bahwa VPN menggunakan terowongan Generic Routing Encapsulation (GRE) untuk menghubungkan dua situs melalui Internet. GRE adalah protokol tunneling OSI layer 3 yang digunakan untuk merangkum paket-paket dari satu protokol lapisan jaringan di atas protokol lapisan jaringan lainnya. Terowongan GRE adalah koneksi point-to-point virtual untuk mentransfer paket. Di router situs pengirim (10.10.10.1/24), paket data yang akan ditransfer, muatan terowongan, dienkapsulasi ke dalam paket protokol jaringan lain dengan header tambahan. Header tambahan menyediakan informasi perutean, sehingga muatan yang dienkapsulasi dapat dikirim melalui terowongan. Di router situs penerima (10.10.20.1/24), paket yang dienkapsulasi di-de-enkapsulasi. Header tambahan dihapus, dan muatan diteruskan ke tujuan akhirnya. Terowongan GRE mendukung penyiaran dan multicasting. Namun, itu tidak mengenkripsi data. Keamanan Protokol Internet (IPSec) dapat digunakan dengan GRE untuk menyediakan koneksi yang aman antara dua situs.

Dalam salah satu arsitektur VPN, koneksi antara LAN dan LAN atau antara LAN dan klien jarak jauh adalah koneksi point-to-point daripada beberapa topologi yang tersedia untuk LAN (seperti LAN yang berisi banyak batang). Ini melarang komunikasi gaya siaran di seluruh VPN. Siaran apa pun dibatasi untuk perangkat tersebut dalam satu LAN.

Komunikasi mungkin dienkripsi dan didekripsi pada titik akhir dengan menggunakan beberapa teknologi enkripsi yang disepakati seperti enkripsi kunci privat dan kunci bersama. Selain itu, VPN memerlukan autentikasi untuk mendapatkan akses ke setidaknya beberapa sumber daya LAN internal. Biasanya, server autentikasi digunakan, di mana, setelah saluran

terenkripsi dibuat, pengguna mengirimkan nama pengguna dan kata sandi. Setelah diautentikasi, pengguna memiliki hak istimewa yang sama untuk sumber daya jaringan seolah-olah pengguna secara fisik berada di lokasi. Kami menghilangkan detail lebih lanjut dari diskusi kami, karena kami belum membahas beberapa konsep yang disebutkan di sini, seperti terowongan SSL/TLS. Kami akan kembali ke beberapa konsep VPN nanti di buku teks saat kami memeriksa komputasi awan.



Gambar 4.15 Tunneling.

BAB 5
PROTOKOL KONTROL TRANSMISI/PROTOKOL INTERNET

5.1 PENDAHULUAN

Kami memperkenalkan beberapa protokol di Bab 1, menekankan model *Open System Interconnection (OSI)* dan tujuh lapisannya. Perkembangan OSI dan *Transmission Control Protocol/Internet Protocol* (TCP/IP) dimulai pada era yang sama, tahun 1970-an; namun, tak satu pun dari keduanya selesai hingga awal 1980-an. Karena keduanya dikembangkan sekitar waktu yang sama dan keduanya dikembangkan oleh insinyur jaringan yang menggunakan pengalaman mereka dalam pengembangannya, keduanya memiliki banyak kesamaan. Faktanya, TCP/IP dan model OSI mungkin lebih mirip daripada perbedaannya.

Dengan demikian, Anda mungkin bertanya-tanya mengapa TCP/IP berhasil dan OSI tidak. Ada banyak alasan untuk ini. Salah satu alasannya adalah sejarah TCP/IP, termasuk bentuk awal protokol IP, telah diimplementasikan dalam Jaringan Badan Proyek Penelitian Lanjutan (ARPANET). Selain itu, OSI adalah model yang dikembangkan secara internasional, sedangkan pengembangan TCP/IP lebih berbasis di Amerika Serikat, di mana ARPANET terutama berada. Terjalin ke dalam dua ide ini adalah fakta bahwa TCP/IP bukan hanya sebuah model tetapi juga sebuah produk. Artinya, TCP/IP sedang diimplementasikan saat sedang dikembangkan. Model OSI tidak pernah diimplementasikan sendiri dan malah menjadi target pengembang jaringan. Dengan demikian, jika seseorang perlu membangun jaringan, memilih pendekatan yang memiliki solusi nyata akan lebih menarik.

Meskipun TCP/IP adalah dasar komunikasi Internet, bukan berarti model OSI tidak berharga atau tidak terpakai. Kita telah menerapkan banyak istilah OSI ke Internet. Sebagai contoh, kami menyebut switch perangkat Layer 2 karena mereka beroperasi di lapisan kedua OSI dan perangkat router Layer 3 karena mereka beroperasi di lapisan ketiga OSI, daripada merujuknya dengan penempatannya di TCP/IP (lapisan tautan dan lapisan jaringan, masing-masing) .

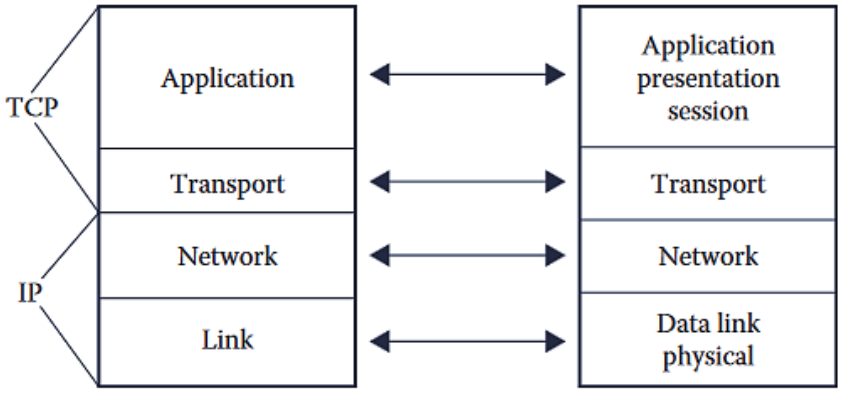
Kami berkonsentrasi pada TCP/IP dalam bab ini, sesekali membuat perbandingan dengan model OSI. TCP/IP adalah tumpukan protokol. Ini terdiri dari dua protokol yang berbeda, Transmission Control Protocol (TCP) dan Internet Protocol (IP), yang masing-masing terdiri dari dua lapisan. Gambar 5.1 mengilustrasikan empat lapisan TCP/IP dibandingkan dengan tujuh lapisan model OSI. Banyak yang membagi lapisan tautan TCP/IP menjadi dua lapisan terpisah, tautan data dan fisik. Jadi, tergantung pada siapa Anda berbicara, TCP/IP dapat dianggap memiliki empat lapisan atau lima lapisan. Kami akan menggunakan empat lapisan di seluruh teks ini tetapi perlu diingat bahwa lapisan tautan data terdiri dari dua bagian yang dapat dipisahkan.

Seperti halnya protokol apa pun, lapisan TCP/IP dijelaskan secara fungsional. Artinya, setiap lapisan mengharapkan jenis masukan tertentu dan menghasilkan jenis keluaran tertentu saat pesan berpindah dari atas tumpukan protokol ke bawah, atau sebaliknya, saat pesan berpindah dari bagian bawah tumpukan dan ke bawah. atas. Fungsi layer

menginformasikan kita tidak hanya perilaku input/output tetapi juga jenis operasi yang harus dipenuhi oleh layer. Apa yang tidak diberitahukan kepada kita adalah bagaimana mengimplementasikan fungsi-fungsi tersebut. Sebagai gantinya, detail implementasi diserahkan kepada pelaksana jaringan.

Saat kita memeriksa TCP/IP, kita akan melihat bahwa untuk setiap lapisan, beberapa implementasi tersedia, masing-masing melalui protokolnya sendiri. Misalnya, pada lapisan aplikasi, kita melihat protokol seperti *Hypertext Transfer Protocol* (HTTP), *HTTP Secure* (HTTPS), *Internet Message Access Protocol* (IMAP), Post Office Protocol (POP), *Simple Mail Transfer Protocol* (SMTP), Protokol Akses Direktori Ringan (LDAP), *Transport Layer Security* (TLS), *Secure Sockets Layer* (SSL), dan *Secure Shell* (SSH). Kami juga melihat skema pengalamatan pada lapisan ini di mana aplikasi spesifik akan ditunjukkan melalui nomor port. Pada lapisan transport, kami biasanya membatasi implementasi pada dua jenis paket: TCP dan *User Datagram Protocol* (UDP). Pada lapisan Internet, banyak protokol berbeda terlibat dengan pengalamatan dan perutean. Ini termasuk IP versi 4 (IPv4) dan IPv6 untuk pengalamatan dan Protokol Konfigurasi Host Dinamis (DHCP) untuk penetapan alamat secara dinamis. Pada layer ini, kita juga dapat menentukan cara menerjemahkan alamat dari alamat eksternal ke alamat internal dengan menggunakan *Network Address Translation* (NAT).

Dalam bab ini, kami mengeksplorasi setiap lapisan secara mendetail. Kami melihat beberapa tapi tidak semua protokol yang saat ini diimplementasikan untuk layer tersebut. Kami melihat detail bagaimana pesan diubah dari lapisan ke lapisan.



Gambar 5.1 Lapisan TCP/IP versus lapisan OSI.

5.2 LAPISAN APLIKASI

Setiap pesan jaringan akan dimulai dengan sebuah aplikasi. Aplikasi bertanggung jawab untuk mengemas pesan bersama. Paling sering, pesan akan dihasilkan sebagai respons terhadap keinginan pengguna untuk komunikasi jaringan. Misalnya, di browser web, pengguna mengklik hyperlink. Operasi klik ini diterjemahkan ke dalam permintaan HTTP. Ini adalah browser web yang menghasilkan pesan awal, permintaan HTTP. Saat pengguna menggunakan program klien email, pengguna mungkin mengetik email dan mengklik kirim. Pada titik ini, klien email akan menghasilkan pesan email untuk dikirim.

Namun, tidak semua pesan dihasilkan dari permintaan pengguna. Beberapa pesan akan dihasilkan secara otomatis sebagai respons terhadap beberapa sistem operasi yang telah diprogram sebelumnya atau rutinitas perangkat lunak. Misalnya, sebagian besar perangkat lunak diprogram untuk memeriksa pembaruan secara otomatis. Perangkat lunak semacam itu akan melakukan ini dengan menghasilkan pesan jaringan untuk dikirim ke situs web yang telah ditentukan sebelumnya. Dalam sistem operasi Linux dan Unix, Protokol Manajemen Jaringan Sederhana (SNMP) digunakan untuk menjelajahi dan memantau perangkat jaringan. Protokol lain yang digunakan secara otomatis adalah *Network Time Protocol* (NTP), digunakan untuk menyinkronkan komputer tertentu dengan *Universal Time Coordinated* (UTC), yang dikelola oleh serangkaian server di Internet.

Aplikasi memanggil protokol yang berbeda. Setiap protokol memiliki format pesan yang sangat spesifik. Misalnya, permintaan HTTP terdiri dari perintah HTTP, nama sumber daya yang diminta, dan versi protokol yang diterapkan. Permintaan tersebut dapat diikuti dengan perintah negosiasi konten dan potongan informasi lainnya. Tabel 5.1 menunjukkan beberapa permintaan HTTP yang berbeda. Protokol lain dengan formatnya sendiri adalah *Domain Name System* (DNS). Seperti halnya HTTP, ada permintaan DNS dan tanggapan DNS. Untuk sisa bagian ini, kami akan memeriksa beberapa protokol populer lainnya yang digunakan dalam lapisan aplikasi TCP/IP.

Tabel 5.1 Contoh Permintaan HTTP

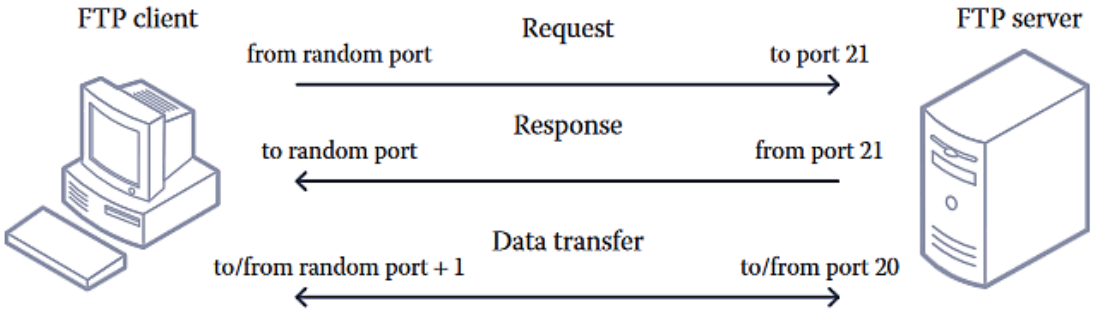
Pesan HTTP	Arti
GET / HTTP/1.0 Host: www.place.com	Dapatkan beranda www.place.com dengan menggunakan HTTP versi 1.0
GET /stuff/myfile.html HTTP/1.0 Host: www.place.com Accept-Language: en fr de	Dapatkan file /stuff/myfile.html dari www.place.com dengan menggunakan HTTP versi 1.0, dengan preferensi tertinggi ke file dalam bahasa Inggris, diikuti oleh bahasa Prancis, diikuti oleh bahasa Jerman
GET http://www.place.com HTTP/1.1 If-Modified-Since: Fri, May 15 2015	Dapatkan halaman beranda www.place.com (perhatikan di sini bahwa kami menyertakan nama host dengan nama file, bukan pernyataan host terpisah) jika halaman telah dimodifikasi sejak tanggal yang ditentukan
HEAD /dir1/dir2 HTTP/1.1	Dapatkan header respons hanya untuk permintaan file di bawah /dir1/dir2, yang namanya cocok dengan file INDEX (mis., index.html), menggunakan HTTP 1.1

File Transfer Protocol

File Transfer Protocol (FTP) adalah protokol tertua yang kami periksa. Sesuai dengan namanya, protokol ini digunakan untuk mengizinkan transfer file dari satu komputer ke komputer lainnya. Sebelum World Wide Web, FTP digunakan secara luas untuk memungkinkan pengguna memindahkan file di Internet. Dengan tersedianya situs web dan server web, FTP menjadi jauh kurang populer tetapi masih tersedia dan bermanfaat, terutama saat mengunggah konten ke server web. Anda mungkin pernah menggunakan program FTP di masa lalu. Di sini, kita melihat cara kerjanya.

FTP, seperti kebanyakan protokol yang dibahas di sini, didasarkan pada model client-server. Pengguna ingin mentransfer file ke atau dari klien saat ini dari atau ke server FTP. Server FTP menyimpan repositori yang digunakan oleh banyak pengguna untuk mengumpulkan dan menyebarkan file. Oleh karena itu, komputer harus dilambangkan sebagai server FTP agar ini berfungsi. Server semacam itu membutuhkan perangkat lunak server FTP yang sedang berjalan. Komputer klien menjalankan perangkat lunak klien FTP. Server FTP dapat diatur untuk meminta login. Sebagai alternatif, server FTP juga dapat mengizinkan akses anonim, di mana pengguna dapat masuk dengan menggunakan kata anonim (atau tamu) sebagai nama pengguna dan alamat emailnya sebagai kata sandi. Ini dikenal sebagai FTP anonim. Jika Anda akan mengizinkan login FTP anonim, Anda mungkin akan membatasi pengguna anonim untuk hanya dapat mengakses direktori publik.

FTP menggunakan paket TCP. Untuk membuat sambungan, FTP menggunakan salah satu dari dua mode, aktif dan pasif. Mari kita periksa mode aktif terlebih dahulu untuk mengetahui mengapa kita membutuhkan mode pasif. Sebelum kita melihat mode-mode ini, perhatikan bahwa FTP memiliki dua port khusus yang ditetapkan untuknya: port 20 untuk transfer data dan port 21 untuk perintah. Kedua port ditugaskan untuk digunakan oleh server; klien tidak harus menggunakan dua port ini dan sering menggunakan dua port yang dipilih secara acak seperti 3850 dan 3851. Kita melihat contoh komunikasi seperti itu pada Gambar 5.2.



Gambar 5.2 Komunikasi klien dan server FTP.

Mari kita asumsikan bahwa klien mengirimkan perintah ke server untuk membuka koneksi. Klien mengirimkan permintaan ini menggunakan port 3850 (port tanpa pagu). Server FTP mengakui permintaan koneksi ini. Jadi, klien telah mengirim dari port 3850 komputernya ke port 21 server FTP. Ini adalah saluran kontrol. Sekarang, saluran data harus dibuka. Server mengirim dari port 20 bukan ke port klien 3850 atau 21 melainkan ke 3851 (satu lebih besar dari port kontrol). Klien menerima permintaan untuk membuka koneksi ke server dari port 20 ke port 3851.

Uraian di atas tampaknya langsung, tetapi ada masalah. Jika klien menjalankan firewall, pilihan port 3850 dan 3851 terlihat acak, dan port ini mungkin diblokir. Sebagian besar firewall diatur untuk memungkinkan komunikasi melalui port seperti 3850 jika port tersebut digunakan untuk membuat koneksi keluar. Firewall kemudian dapat mengharapkan

respons melalui port 3850 tetapi tidak melalui port 3851, karena tidak digunakan dalam pesan keluar. Selain itu, server FTP tidak mengirim pesan melalui port 3851 miliknya sendiri.

Ada dua solusi untuk masalah ini. Pertama, jika pengguna mengetahui port apa yang akan dia gunakan untuk FTP, dia dapat membuka blokir port tersebut dan port berikutnya secara berurutan. Pengguna yang naif tidak akan memahami ini. Jadi sebagai gantinya, FTP bisa digunakan dalam mode pasif. Dalam mode ini, klien akan membuat kedua koneksi, sekali lagi kemungkinan besar dengan menggunakan port berurutan. Artinya, klien akan mengirimkan permintaan koneksi pasif ke server. Server tidak hanya mengakui permintaan ini tetapi juga mengirimkan kembali nomor port. Setelah diterima, klien sekarang bertanggung jawab untuk membuka koneksi data ke server. Itu akan melakukannya dengan menggunakan port berturut-turut berikutnya dari ujungnya tetapi akan meminta nomor port yang dikirimkan kepadanya dari server untuk port server. Artinya, dalam mode pasif, port 20 tidak digunakan untuk data, melainkan nomor port yang dikirim server digunakan.

Mode aktif adalah mode default. Mode pasif adalah pilihan. Kedua mode ini tidak boleh dikacaukan dengan dua jenis mode lainnya, seperti mode pengkodean data (American Standard Code for Information Interchange [ASCII] vs biner vs bentuk pengkodean karakter lainnya) dan mode transfer data (stream, blok, atau dikompresi).

Mari kita periksa mode transfer data ini. Dalam mode terkompresi, beberapa algoritme kompresi digunakan pada file data sebelum transmisi dan kemudian tidak dikompresi di ujung lainnya. Dalam mode blok, klien FTP memecah file menjadi blok, yang masing-masing dienkapsulasi dan ditransmisikan secara terpisah. Dalam mode aliran, FTP melihat transfer file sebagai seluruh file, menyerahkannya kepada lapisan transport untuk membagi file ke dalam paket TCP individual.

FTP awalnya diimplementasikan dalam program berbasis teks, sering disebut ftp. Anda akan memasukkan perintah pada prompt. Tabel 5.2 memberikan daftar beberapa perintah FTP yang lebih berguna. Ini ditampilkan dalam bentuk mentahnya (bagaimana mereka muncul dalam protokol) dan bagaimana mereka akan muncul ketika dikeluarkan menggunakan program FTP baris perintah. Hari ini, FTP diimplementasikan di sejumlah program antarmuka pengguna grafis (GUI) seperti WinSock FTP (WS-FTP), WinSCP, dan FileZilla. Ini juga tersedia dalam program seperti Putty. FTP, sebagai protokol, juga dapat ditangani oleh server web. Oleh karena itu, Anda mungkin dapat mengeluarkan perintah FTP get di browser web; namun, ada sedikit perbedaan antara itu dan mengeluarkan perintah get dengan menggunakan HTTP melalui browser web.

Setelah menyelesaikan setiap perintah yang dikirim dari klien ke server, server akan merespons dengan kode pengembalian tiga digit. Kode respons yang paling penting adalah 200 untuk sukses. Tanggapan 100 menunjukkan bahwa tindakan yang diminta telah dimulai tetapi belum selesai dan kode pengembalian lainnya akan dikirim setelah selesai. Kode 400 menunjukkan bahwa perintah tidak diterima karena kesalahan sementara, sedangkan kode 500 menunjukkan perintah yang tidak valid secara sintaksis. Ada banyak kode lain. Jika Anda berencana mengimplementasikan server atau klien FTP Anda sendiri, Anda perlu menjelajahi perintah FTP dan mengembalikan kode secara lebih mendetail.

Cara lain untuk mentransfer file adalah melalui rcp (salinan jarak jauh), salah satu program r-utilitas Unix/Linux. Dengan program r-utility, Anda melewati autentikasi saat komputer klien dan server Unix/Linux berbagi server autentikasi yang sama. Artinya, pengguna memiliki akun yang sama di kedua mesin. rcp adalah r-utilitas yang dapat melakukan perintah cp (operasi penyalinan Unix/Linux).

dari komputer ke komputer. Ada juga FTP Sederhana (perhatikan bahwa ini tidak disingkat SFTP karena SFTP memiliki arti yang berbeda seperti yang dijelaskan di bawah). Protokol lainnya adalah BitTorrent, yang, tidak seperti FTP dan HTTP, merupakan bentuk komunikasi peer-to-peer. Kami tidak akan mengeksplorasi protokol ini di sini.

Tabel 5.2 Perintah FTP

FTP Command	Arti	Setara Baris Perintah
ABOR	Batalkan transfer saat ini	N/A
CWD dir	Ubah direktori di komputer klien ke dir	lcd dir
DELE file	Menghapus berkas	delete file
LIST	Daftar file di direktori server saat ini	ls, dir
MKD dir	Buat direktori direktori di direktori kerja saat ini di server	mkdir dir
MODE S | B | C	Ubah mode transfer ke streaming (S), blok (B), atau terkompresi (C)	N/A
PASV	Setel mode pasif (bukan mode aktif)	N/A
PWD	Cetak direktori kerja saat ini di server	pwd
QUIT	Tutup koneksi	quit, exit (closes connection and exits the FTP program)
RETR file	Transfer (ambil) file dari server di direktori saat ini ke klien Anda di direktori saat ini	get file
RMD dir	Hapus dir direktori dari server	rmdir dir
STOR file	Transfer (kirim) file dari klien Anda di direktori saat ini ke server di direktori saat ini	put file
TYPE A | E | I | L	Ubah jenis transfer ke format ASCII (A), EBCDIC (E), biner (I), atau lokal (L)	ascii, binary (also i)

FTP mengirimkan perintah dan mengembalikan file ASCII dalam teks yang jelas, termasuk kata sandi yang dikirim ke server saat memulai komunikasi. Ada beberapa pilihan untuk melakukan transfer file dengan cara yang aman. Pertama, Anda dapat membuat saluran aman dengan menggunakan SSL, lalu, dari dalam SSL, Anda membuka komunikasi FTP. Dengan cara ini, komunikasi FTP itu sendiri tidak aman, tetapi setiap komunikasi antara klien dan server aman karena koneksi SSL dibuat terlebih dahulu. Ini dikenal sebagai FTPS. Pendekatan serupa adalah membuka koneksi SSH dan kemudian melakukan FTP dari dalam koneksi SSH, yang disebut SFTP. Karena SSL dan SSH tidak menggunakan port yang sama dengan FTP, komunikasi menggunakan FTPS dan SFTP berbeda dari apa yang disajikan sebelumnya pada

Gambar 5.2. Misalnya, komunikasi SSH biasanya dilakukan melalui port 22, sehingga SFTP juga akan dilakukan melalui port 22.

Jangan bingung antara FTPS dan SFTP. Meskipun FTPS dan SFTP memiliki sifat yang serupa, SSL dan SSH sangat berbeda. Dalam kedua kasus tersebut, kami melakukan tugas FTP dengan menggunakan tunneling karena kami menggunakan saluran komunikasi yang sudah ada untuk membuat saluran lain. Kami menjelajahi SSL dan SSH nanti di bagian ini. Pilihan lainnya adalah menggunakan bentuk aman dari salinan jarak jauh yang disebut SCP.

Protokol konfigurasi host dinamis

Ada dua cara bagi perangkat untuk mendapatkan alamat IP: secara statis dan dinamis. Alamat IP yang ditetapkan secara statis adalah yang diberikan ke perangkat oleh administrator jaringan atau sistem. Alamat disimpan dalam file, dan sejak saat itu, alamat IP tidak berubah (kecuali administrator mengedit file untuk mengubah alamat).

Alamat IP statis lebih disukai untuk sebagian besar jenis server Internet karena alamat server harus mudah diidentifikasi. Alamat dinamis dapat berubah dalam waktu singkat dan karenanya tidak menawarkan stabilitas alamat IP statis. Sebagian besar komputer klien tidak memerlukan stabilitas seperti itu sehingga tidak perlu memiliki alamat IP statis. Pada tahun 1984, sebuah protokol dikembangkan untuk memungkinkan komputer memperoleh alamat IP yang ditetapkan secara dinamis dari server jaringan. Dalam protokol awal ini, pertukaran antara permintaan dan tanggapan dilakukan pada lapisan tautan, yang mengharuskan server hadir di jaringan yang diberikan. Untuk alasan ini dan lainnya, protokol berbeda yang disebut *Bootstrap Protocol* (BOOTP) dikembangkan. Itu juga telah ketinggalan zaman dan telah digantikan oleh DHCP.

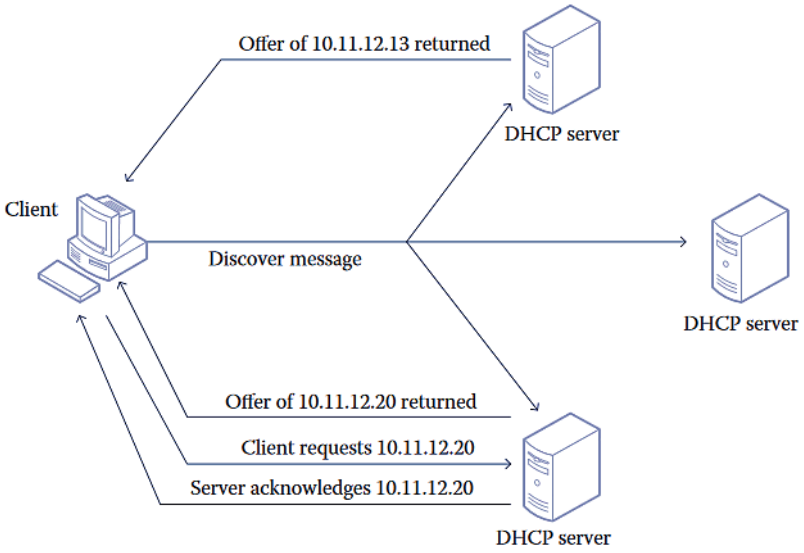
DHCP dikenal sebagai model server tanpa koneksi di mana sesi komunikasi tidak diperlukan untuk mengeluarkan alamat IP. Sebaliknya, DHCP menggunakan proses empat langkah di mana komputer klien pertama kali menemukan server yang dapat mengeluarkan alamat IP untuk itu. Server menawarkan alamat IP ke klien. Klien kemudian menanggapi dengan permintaan untuk mengambil alamat IP yang ditawarkan. Server kemudian mengakui tanda terima. Keempat langkah tersebut disingkat menjadi DORA (*Discover, Offer, Request*, dan *Acknowledge*). Anda mungkin bertanya-tanya mengapa diperlukan empat langkah. Karena tidak ada koneksi khusus yang dibuat, klien dapat menemukan beberapa server, yang masing-masing mungkin menawarkan alamat IP yang berbeda. Klien hanya akan memerlukan satu alamat, dan karenanya, baru setelah klien merespons dengan permintaan untuk alamat tertentu, alamat tersebut akan diberikan. Dengan cara ini, server juga mengetahui bahwa alamat IP yang diberikan telah diterima oleh klien sehingga alamat IP tersebut tidak dapat ditawarkan ke klien lain.

DHCP menggunakan satu format untuk masing-masing dari empat komunikasi antara klien dan server (temukan pesan, pesan penawaran, pesan permintaan, dan pesan pengakuan). Saat pesan berpindah dari klien ke server dan kembali ke klien, konten berubah, karena setiap perangkat akan mengubah informasi. Bidang pesan termasuk alamat IP klien, alamat IP server, alamat kontrol akses media (MAC) klien, alamat IP yang ditawarkan, alamat gateway jaringan, netmask jaringan, dan informasi lainnya, beberapa di antaranya akan menjadi dijelaskan kemudian.

Pesan penemuan dikirim pada subnet klien dengan alamat tujuan 255.255.255.255. Ini memastikan bahwa permintaan tidak menyebar di luar subnet. Alamat IP sumber adalah 0.0.0.0, karena sampai saat ini, klien tidak memiliki alamat IP. Komunikasi dilakukan melalui dua port: klien menggunakan port 68 dan server menggunakan port 67. Karena subnet tempat pengiriman pesan penemuan ini akan dilayani oleh router, pesan penemuan akan diterima oleh router subnet. Merupakan tanggung jawab router ini untuk menangani permintaan DHCP secara langsung (jika diimplementasikan untuk melakukannya) atau meneruskan pesan ke server DHCP yang dikenal.

Setelah menerima pesan penemuan, perangkat yang menangani pesan penemuan DHCP akan mengubah pesan untuk menjadikannya penawaran. Penawaran tersebut menyediakan sewa alamat IP. Istilah sewa digunakan karena server DHCP menyediakan alamat IP hanya untuk waktu yang terbatas. Sewa mungkin berjam-jam, berhari-hari, atau berminggu-minggu. Server DHCP menyimpan daftar alamat perangkat keras klien sebelumnya, bersama dengan alamat IP yang terakhir diberikan. Jika alamat IP ini tersedia, ia akan menggunakan alamat yang sama dalam sewa ini, sehingga klien mendapatkan kembali alamat IP yang sama. Namun, tidak ada jaminan bahwa alamat yang sama akan ditawarkan karena mungkin terdapat lebih banyak klien dalam jaringan daripada jumlah alamat IP yang tersedia. Kumpulan alamat IP yang tersedia untuk diterbitkan dikenal sebagai kumpulan server. Pesan penawaran memperbarui alamat IP klien dari 0.0.0.0 ke IP yang ditawarkan. Itu juga mengatur netmask untuk alamat, alamat IP DHCP (bukan 255.255.255.255), dan durasi sewa dalam hitungan detik. Adalah opsional untuk memiliki satu atau lebih alamat server nama DNS. Beberapa server DHCP mungkin telah menanggapi permintaan klien, sehingga beberapa penawaran dapat dikembalikan ke klien. Klien memilih satu penawaran untuk ditanggapi. Klien mengembalikan permintaan ke semua server yang merespons dengan penawaran, tetapi permintaan tersebut hanya menyertakan alamat IP dari satu server. Server lain yang menerima permintaan ini menghapus penawaran mereka, mempertahankan alamat IP yang ditawarkan di kumpulan mereka. Satu server yang alamatnya ada dalam permintaan tahu bahwa klien menginginkan alamat IP yang ditawarkannya. Server itu kemudian memodifikasi tabelnya sendiri untuk menunjukkan bahwa alamat IP yang ditawarkan sekarang sedang digunakan bersama dengan durasi sewa sehingga server mengetahui kapan alamat tersebut dapat diklaim kembali. Dengan pembaruan ini, server DHCP sekarang merespons klien dengan pemberitahuan sehingga klien dapat mulai menggunakan alamat IP ini.

Gambar 5.3 mengilustrasikan proses ini. Dalam hal ini, ada tiga server DHCP yang tersedia untuk klien yang membuat permintaan. Setelah mengirimkan pesan penemuan, dua server DHCP merespons, masing-masing menawarkan alamat IP 10.11.12.13 dan 10.11.12.20. Kedua server adalah bagian dari subnet yang sama, tetapi keduanya memiliki kumpulan alamat IP yang berbeda untuk disewa. Klien, untuk alasan apapun, memutuskan untuk memilih 10.11.12.20. Ini mungkin karena ini adalah alamat IP terakhirnya atau karena penawaran ini yang pertama kali diterima. Klien merespons dengan permintaan untuk 10.11.12.20, dan server (yang paling bawah pada gambar) mengembalikan sebuah pengakuan. Saat ini, klien dapat mulai menggunakan 10.11.12.20 sebagai alamat IP-nya.



Gambar 5.3 Pesan DHCP dengan beberapa server.

Protokol pesan DHCP memungkinkan sejumlah variabel parameter untuk digunakan dalam fase penemuan, penawaran, permintaan, atau pengakuan. Setiap opsi dimulai dengan nomor kode 1-byte, yang menentukan jenis opsi. Nomor opsi berkisar dari 0 hingga 255; namun, saat ini hanya ada 20 opsi yang digunakan (per RFC 2132). Salah satu pilihan, misalnya, daftar router yang tersedia dalam urutan preferensi di mana router harus digunakan. Opsi khusus ini mencakup daftar alamat IP 4-byte. Opsi lain mengizinkan penyertaan nama domain (setidaknya sepanjang 1 byte). Pilihan lain adalah dimasukkannya daftar server nama domain untuk jaringan, dalam urutan preferensi di mana mereka harus digunakan. Informasi lain juga dapat disertakan melalui opsi seperti server waktu default, server web default, dan server email default. Kami akan menjelajahi DHCP nanti di bab ini saat kami melihat alamat IP statis versus dinamis..

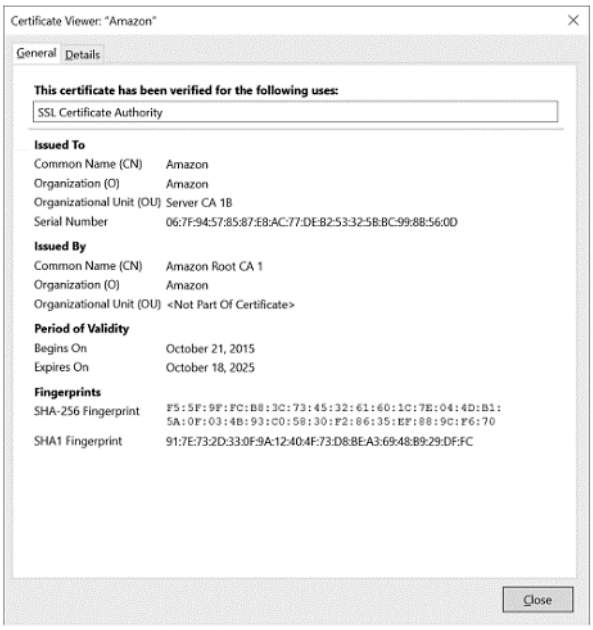
Keamanan lapisan soket dan keamanan lapisan transportasi

Kami telah mencatat bahwa TCP/IP diatur untuk menangani pesan yang dikirim dalam teks ASCII. Pesan dikirim dalam format yang terlihat, dan setiap informasi rahasia yang dikirim dalam pesan tersebut tidak aman, karena pesan tersebut dapat dicegat oleh pihak ketiga dengan menggunakan berbagai mekanisme, beberapa di antaranya telah disebutkan di Bab 2. Seperti yang kami inginkan komunikasi yang aman untuk memastikan bahwa informasi rahasia, seperti kata sandi dan nomor kartu kredit, tidak dapat dilihat bahkan jika pesan dicegat, apakah kita memerlukan pelaksana jaringan untuk mengubah TCP/IP atau pengguna perlu mengenkripsi dan mendekripsi pesan di kedua ujung komunikasi. Pendekatan terakhir adalah salah satu yang telah diadopsi untuk TCP/IP.

Hal ini dilakukan dengan menambahkan protokol pada lapisan aplikasi, dimana pesan dapat dienkripsi dan kemudian didekripsi. Dua protokol yang paling umum adalah SSL dan TLS. Meskipun mereka adalah protokol yang berbeda, mereka memiliki fungsi yang serupa.

Kami akan menggabungkan diskusi kami tentang SSL dan TLS dan memperlakukannya sebagai hal yang sama. Komunikasi paling aman melalui Internet menggunakan TLS versi 1.2 atau SSL versi 3.1. Sebagian besar TLS diimplementasikan berdasarkan versi SSL sebelumnya

(3.0). *Transport Layer Security* 1.3 telah dirancang tetapi belum diimplementasikan. Kami akan merujuk kedua protokol tersebut bersama-sama di seluruh buku teks ini sebagai TLS/SSL. Perhatikan bahwa TLS/SSL beroperasi bersama dengan protokol lain. Artinya, protokol yang tidak aman (misal FTP) mungkin disalurkan dari dalam koneksi TLS/SSL, sehingga transmisi kini aman. Dengan demikian, TLS/SSL tidak menggunakan satu bentuk paket (mis., TCP dan UDP) melainkan mengizinkan keduanya. Awalnya, TLS hanya mengimplementasikan TCP, tetapi kemudian dimodifikasi untuk menangani UDP.



Gambar 3.4 Contoh sertifikat X.509.

TLS/SSL menggunakan enkripsi kunci publik, di mana ada dua kunci: kunci publik untuk mengenkripsi pesan dan kunci pribadi untuk mendekripsi pesan. TSL/SSL bekerja dengan menempatkan kunci publik dalam sertifikat digital X.509. X.509 adalah salah satu standar untuk mengimplementasikan infrastruktur kunci publik (PKI). Secara khusus, X.509 melarang kebijakan penerapan enkripsi kunci publik untuk sertifikat digital, termasuk format sertifikat, pencabutan sertifikat, dan algoritme untuk memverifikasi keaslian sertifikat. Sertifikat kemudian harus berisi informasi untuk memvalidasi server tujuan sebagai yang sah. Ini juga akan mencakup deskripsi organisasi tujuan, kunci publik, dan informasi kedaluwarsa. Saat kita melihat server web. Untuk saat ini, kami akan membahas beberapa ide di balik X.509 dan TLS/SSL.

Sertifikat X.509 akan menyertakan nomor versi, nomor seri, pengidentifikasi algoritme enkripsi, informasi kontak organisasi yang memegang sertifikat (yang mungkin mencakup nama, kontak, alamat, dan informasi berguna lainnya), dan tanggal validitas . Sertifikat juga akan berisi kunci publik dan algoritma kunci publik. Juga akan ada tanda tangan. Tanda tangan inilah yang digunakan untuk memverifikasi penerbit. Tanda tangan dihasilkan oleh otoritas tanda tangan, yang tugasnya memverifikasi organisasi.

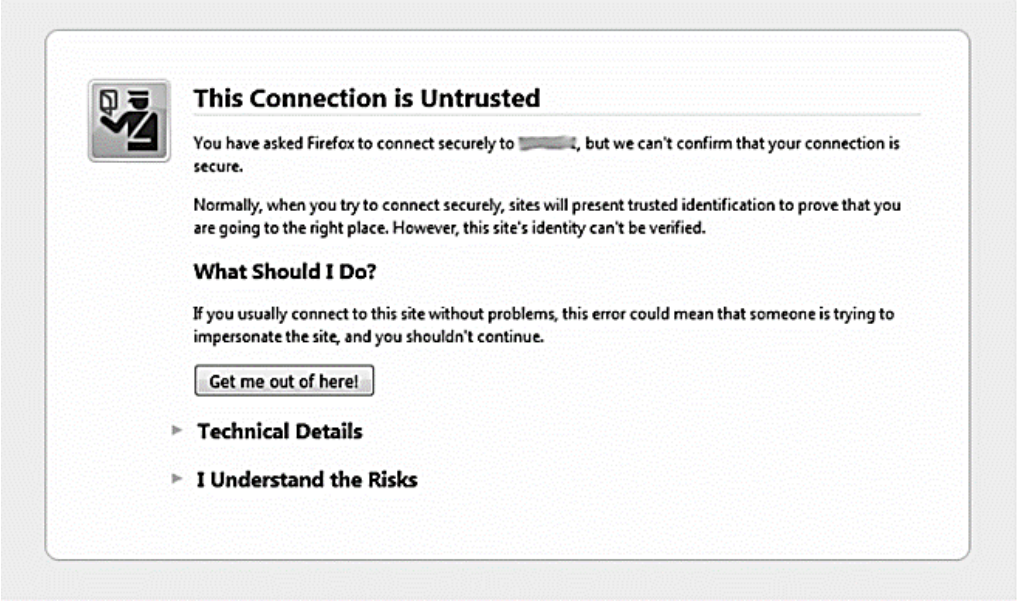
Contoh sertifikat, seperti yang terlihat di browser Firefox Mozilla, ditunjukkan pada Gambar 5.4. Sertifikat ini dari perusahaan amazon.com, ditandatangani oleh otoritas tanda tangan Symantec. Pada gambar, Anda mungkin memperhatikan bahwa beberapa detail yang dijelaskan di atas tampak hilang.

Faktanya, informasi tambahan dapat ditemukan di tab Detail. Sertifikat ini menggunakan versi 3 dan nomor seri 06:7F:94:57:85:87:E8:AC:77:DE:B2:53:32:5B:BC:99:8B:56:0D (ini adalah bidang 19-byte ditampilkan menggunakan notasi heksadesimal). Kunci publik juga dapat ditemukan di bawah rincian. Dalam hal ini, kunci publik adalah nilai 2048-bit yang ditampilkan sebagai daftar 16 baris dari 16 angka heksadesimal dua digit (16 baris, 16 angka [masing-masing 2 digit atau 8 bit masing-masing adalah 16 * 16 * 8 = 2048 bit]). Dua baris kunci ini ditampilkan sebagai berikut:

94 9f 2e fd 07 63 33 53 b1 be e5 d4 21 9d 86 43
70 0e b5 7c 45 bb ab d1 ff 1f b1 48 7b a3 4f be

Bagian detail juga berisi nilai tanda tangan, nilai 2048-bit lainnya. Memperoleh tanda tangan dari otoritas tanda tangan dapat memakan biaya mulai dari nol hingga ratusan atau ribuan dolar (AS) per tahun. Biaya tergantung pada faktor-faktor seperti perusahaan yang menandatangani sertifikat, tanggal validitas, dan kelas sertifikat. Beberapa perusahaan akan menandatangani sertifikat secara gratis untuk jenis sertifikat tertentu: CACert, StartSSL, atau COMODO. Lebih umum, perusahaan mengenakan biaya untuk layanan, di mana reputasi mereka sedemikian rupa sehingga pengguna yang menerima sertifikat yang ditandatangani oleh perusahaan tersebut dapat mempercayai mereka. Perusahaan-perusahaan ini termasuk Symantec (yang menandatangani sertifikat di atas dari amazon.com), Verisign, GoDaddy, GlobalSign, dan COMODO yang disebutkan di atas.

Organisasi yang ingin berkomunikasi menggunakan TLS/SSL tidak harus membeli tanda tangan tetapi dapat membuat tanda tangannya sendiri. Sertifikat semacam itu disebut self-signed. Aplikasi apa pun yang memerlukan TLS/SSL dan menerima sertifikat yang ditandatangani sendiri dari server akan memperingatkan pengguna bahwa sertifikat tersebut mungkin tidak dapat dipercaya. Pengguna kemudian memiliki pilihan apakah akan melanjutkan atau membatalkan komunikasi. Anda mungkin pernah melihat pesan seperti yang ditunjukkan pada Gambar 5.5 saat mencoba berkomunikasi dengan server web, yang mengembalikan sertifikat yang ditandatangani sendiri.

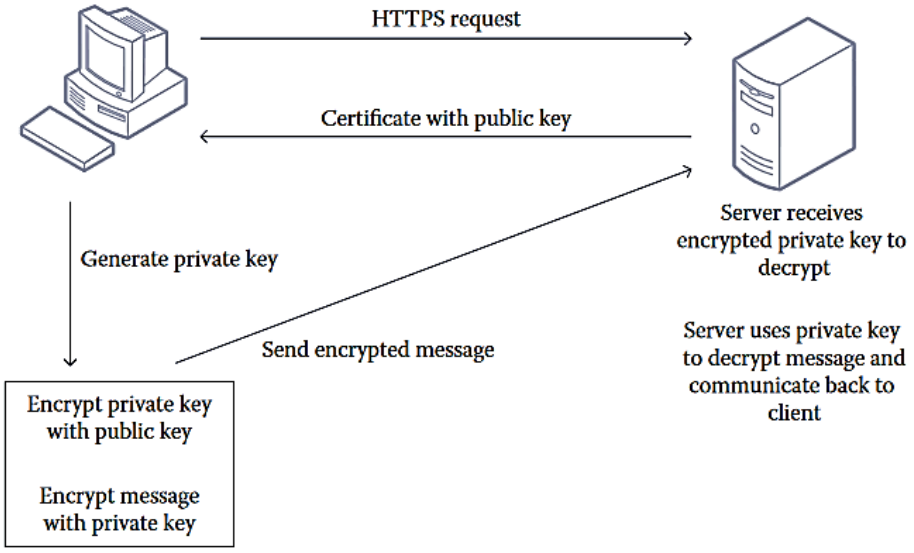


Gambar 5.5 Halaman browser menunjukkan sertifikat digital yang tidak dapat dipercaya.

Untuk memanfaatkan TLS/SSL, server harus menyadari bahwa komunikasi akan dienkripsi. Hal ini dilakukan dengan salah satu dari dua pendekatan yang berbeda. Pertama, banyak protokol meminta port tertentu untuk digunakan. Protokol yang beroperasi di bawah TLS/SSL akan menggunakan port yang berbeda dari versi yang tidak beroperasi di bawah TLS/SSL. Misalnya, HTTP dan HTTPS adalah protokol yang sangat mirip. Keduanya digunakan untuk berkomunikasi antara klien web dan server web. Namun, HTTPS digunakan untuk komunikasi terenkripsi, sedangkan HTTP digunakan untuk komunikasi tidak terenkripsi. HTTP biasanya beroperasi melalui port 80 server, sedangkan HTTPS beroperasi melalui port 443 server. Pendekatan kedua adalah memiliki perintah khusus yang akan dikirim klien ke server, meminta agar komunikasi beralih dari tidak terenkripsi menjadi terenkripsi. LDAP menawarkan perintah seperti itu, StartTLS.

Apa pun pendekatan yang digunakan, klien harus mendapatkan sertifikat digital dari server. Ini biasanya merupakan transaksi satu kali di mana klien akan menyimpan sertifikat digital tanpa batas waktu setelah diterima. Transmisi sertifikat memerlukan bentuk jabat tangan sendiri. Pertama, klien menunjukkan bahwa ia menginginkan komunikasi yang aman dengan server. Koneksi sudah dibuat, dan sekarang, klien menunjukkan bahwa ia ingin mengenkripsi pesannya lebih lanjut. Klien menyediakan ke server daftar algoritme yang telah tersedia untuk digunakan. Ada sejumlah algoritma berbeda yang digunakan seperti DES, Triple DES, AES, RSA, DSA, SHA, MDn (mis., MD5), dan sebagainya. Lihat Bab 1 untuk detailnya. Server memilih algoritme yang tersedia untuknya (mungkin berdasarkan daftar prioritas) dan merespons klien dengan pilihannya. Server mengirimkan sertifikat digital ke klien. Terakhir, klien menghasilkan nomor acak dan mengenkripsinya, mengirimkannya ke server. Server mendekripsi nomor ini, dan klien serta server menggunakan nomor ini sebagai nomor sesi. Nomor sesi digunakan untuk keamanan tambahan dalam algoritma enkripsi/dekripsi. Dengan sertifikat digital yang tersedia, klien kini memiliki kunci publik dari server. Namun, menggunakan enkripsi asimetris untuk pesan yang panjang tidak terlalu efisien. Sebaliknya,

kami lebih suka menggunakan enkripsi kunci simetris. Ini memperumit masalah karena klien memiliki kunci publik untuk mengenkripsi pesan yang dapat didekripsi oleh server menggunakan kunci pribadinya, tetapi mereka tidak memiliki kunci pribadi yang sama. Jadi, langkah selanjutnya adalah klien membuat kunci pribadi satu kali. Kunci publik, yang diterima melalui sertifikat, kemudian digunakan untuk menyandikan kunci privat yang baru dibuat. Kunci pribadi baru digunakan untuk mengenkripsi pesan, termasuk data rahasia seperti kata sandi dan nomor kartu kredit.



Gambar 5.6 Enkripsi kunci publik menggunakan sertifikat X.509.

Pesan terenkripsi kemudian dikirim ke server. Server memperoleh bagian dari pesan yang merupakan kunci pribadi terenkripsi milik klien. Karena kunci dienkripsi menggunakan kunci publik server, server dapat mendekripsi ini dengan menggunakan kunci pribadinya sendiri. Sekarang, server memiliki kunci pribadi dari klien. Server sekarang dapat menggunakan kunci pribadi untuk mendekripsi sisa pesan, yang merupakan pesan sebenarnya yang dimaksudkan klien untuk server. Selanjutnya, sekarang klien dan server memiliki kunci pribadi bersama, mereka dapat terus menggunakan kunci pribadi ini sementara koneksi ini tetap terjalin. Proses ini ditunjukkan pada Gambar 5.6.

Protokol Email

Ada beberapa protokol email yang biasa digunakan di Internet. Kami membaginya menjadi dua set: pengiriman email dan protokol pengiriman dan protokol pengambilan email. Kumpulan sebelumnya terdiri dari protokol-protokol yang menangani pengiriman pesan email klien ke servernya, yang kemudian mengirimkan pesan ke server penerima. Kumpulan protokol terakhir terdiri dari yang memungkinkan klien untuk mengambil email yang menunggu dari servernya. Protokol penyerahan/pengiriman yang paling umum adalah SMTP. Ada dua protokol pengambilan umum yang digunakan: POP dan IMAP. Pada subbagian ini, kita melihat secara singkat ketiga protokol ini.

SMTP tanggal kembali ke awal 1980-an. Hingga saat ini, sebagian besar protokol email dimaksudkan untuk digunakan dari dalam satu komputer mainframe (atau dalam jaringan

mainframe organisasi) atau dalam jaringan area luas yang diperkecil, seperti halnya dengan ARPANET pada tahun 1970-an ketika ada 100 atau kurang host. Selain itu, protokol sebelumnya sebagian besar diimplementasikan di Unix, karena ini adalah sistem operasi utama untuk host ARPANET. SMTP memodelkan dirinya sendiri setelah dua konsep. Yang pertama adalah protokol komunikasi satu-ke-banyak. Artinya, pesan email tidak harus atau secara implisit ditujukan untuk satu tujuan. Sebaliknya, email apa pun berpotensi dikirim ke pengguna yang berbeda dari server email yang berbeda. Kedua, SMTP ditujukan untuk server yang bertahan di jaringan daripada server yang mungkin ditambahkan ke jaringan untuk sementara dan kemudian dihapus atau dimatikan.

Karena sebagian besar Internet berbasis Unix, protokol SMTP pertama kali diterapkan di Unix, dalam hal ini dalam program yang disebut sendmail. Versi sendmail yang direvisi akhirnya diimplementasikan, disebut postfix. Akibatnya, postfix menggunakan sendmail sehingga postfix menjadi program yang lebih baru dan lebih komprehensif. Anda dapat menjalankan sendmail tanpa postfix, tetapi tidak sebaliknya.

Maksud asli dari SMTP adalah klien (di komputer klien) akan memasukkan pesan email. Pesan kemudian akan ditransfer ke server email klien. Server akan terdiri dari dua bagian: agen pengiriman surat dan agen transfer surat. Kedua bagian ini dapat berjalan di komputer terpisah tetapi kemungkinan besar akan berjalan di satu komputer dan menjadi bagian dari program yang sama. Agen pengiriman surat akan menerima email dari klien dan meneruskannya ke agen transfer surat untuk dikirim melalui Internet, apakah itu satu program di satu komputer atau dua (atau lebih) program di dua (atau lebih) komputer. Email kemudian akan dikirim. Jika alamat tujuan email adalah internal, agen transfer pada dasarnya akan mengirim email ke dirinya sendiri. Jika tidak, pesan email akan melintasi Internet dan tiba di server email pengguna (atau pengguna) tujuan. Sekali lagi, akan ada dua proses yang berbeda. Email tersebut akan diterima oleh pertukaran email, yang kemudian akan meneruskan email tersebut ke agen pengiriman email. Ini lagi bisa menjadi satu komputer atau beberapa komputer.

Pesan SMTP menggunakan paket TCP untuk memastikan pengiriman dengan port default 587. Layanan email lama menggunakan port 25, dan port ini juga diperbolehkan. SMTP menentukan bagaimana pesan email dikirim dan dikirim. Awalnya, isi pesan diharapkan berbasis teks ASCII. Dengan Multipurpose Internet Mail Extensions (MIME), SMTP kini dapat membawa segala bentuk konten, di mana konten tersebut harus dikodekan dari bentuk aslinya (kemungkinan besar, file biner) menjadi setara berbasis teks.

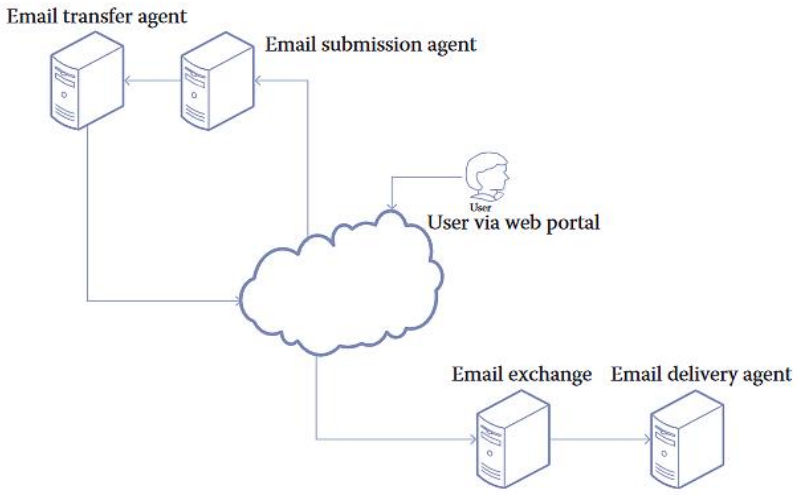
Bentuk awal SMTP mensyaratkan bahwa pesan email harus berasal dari dalam domain atau jaringan yang sama dengan agen pengiriman dan transfer. Namun, pada akhir 1990-an, gagasan ini sudah ketinggalan zaman. Dengan portal web yang memungkinkan akses ke server email, menjadi diinginkan untuk mengizinkan seseorang mengakses server email mereka dari jarak jauh, sehingga mereka dapat menyiapkan dan mengirimkan pesan dari satu domain tetapi server mereka ada di domain lain. Fitur ini ditambahkan ke SMTP pada tahun 1998. Gambar 5.7 mengilustrasikan ide ini. Pada tahun 1999, keamanan tambahan ditambahkan ke SMTP untuk memastikan bahwa pesan email yang dikirim dari jarak jauh ke server sebenarnya sah. Tanpa ini, spammer dapat membanjiri server email, mengharapkan email tersebut

terkirim. Siapa pun yang menerima pesan spam mungkin berpikir bahwa email tersebut berasal dari domain server yang mengirimkan pesan, tetapi, pada kenyataannya, spammer membuat pesan tersebut berasal dari beberapa situs jarak jauh.

Di SMTP, server yang mengirim pesan membuka dan menjaga koneksi dengan server yang menerima pesan. Klien juga dapat terlibat dalam koneksi terbuka ini, karena sesi email dapat bersifat interaktif. Tiga jenis pesan disampaikan antara klien, server pengirim, dan server penerima. Pertama adalah perintah, MAIL, untuk menetapkan alamat pengirim. Setiap email dicap dengan alamat pengirim. Kedua adalah RCPT untuk membangun koneksi antara pengirim dan penerima. Pesan ini digunakan sekali per penerima. Misalnya, jika pesan email ditujukan untuk empat pengguna, RCPT digunakan empat kali. Terakhir, perintah DATA digunakan untuk menunjukkan awal pesan. Pesan terdiri dari dua bagian: header, yang menyertakan alamat email pengirim dan penerima, stempel waktu/tanggal, subjek, dan informasi lainnya, dan isi pesan. Tubuh dipisahkan dari header dengan baris kosong. Tubuh diakhiri dengan titik untuk menunjukkan akhir pesan.

Saat menerima pesan email, server harus menghubungi kliennya untuk memberi tahu bahwa ada email baru yang menunggu. Dua protokol umum untuk pengiriman pesan adalah *Post Office Protocol* (POP) dan *Internet Message Access Protocol* (IMAP). POP lebih tua dan lebih primitif dari keduanya. Kami akan mulai dengan itu.

Gagasan di balik POP adalah klien email membuka koneksi untuk mengakses kotak surat di server. Semua email yang menunggu diunduh dan disimpan di mesin klien. Kemudian, pengguna dapat mengakses pesan email tersebut melalui perangkat lunak klien. Pengguna memiliki opsi untuk meninggalkan email apa pun di server, dalam hal ini email tersebut disajikan sebagai email baru saat klien masuk ke server POP di lain waktu. Jika tidak, secara default, semua email dihapus dari server setelah dikirim ke klien.



Gambar 5.7 Server SMTP menerima permintaan email dari jarak jauh.

Transfer cepat, di mana ada sedikit variabilitas fungsi, membuat POP mudah diimplementasikan. Selain itu, kotak surat di server diperlakukan sebagai file tunggal, di mana pesan disegmentasikan oleh beberapa pemisah formal. Penyederhanaan ini, di sisi lain, juga

membatasi kegunaan POP. Kotak surat bersama tidak diperbolehkan. Server POP hanya melayani satu klien pada satu waktu. Selain itu, pesan diunduh secara keseluruhan. Jika pesan berisi lampiran yang disandikan MIME, pengguna mungkin hanya ingin mengunduh pesan atau satu lampiran, tetapi hal ini tidak dapat dilakukan dengan POP.

Dalam banyak hal, IMAP beroperasi berlawanan dengan POP. Pesan email ditinggalkan di server, memungkinkan satu kotak surat untuk dibagikan. Selain itu, klien dapat memiliki beberapa kotak surat dengan IMAP. Klien menyalin pesan ke komputernya sendiri untuk melihat, menyimpan, atau mengeditnya. Karena kotak surat bukan satu file, pesan di kotak surat ditangani jauh berbeda dari pada POP. Pesan dapat diakses secara keseluruhan atau sebagian (misalnya, satu lampiran dapat diakses tanpa mengakses pesan atau lampiran lainnya). Kotak surat untuk klien dapat disegmentasi ke dalam folder. Email individual dapat dihapus dari server, tanpa memengaruhi kotak surat secara keseluruhan. Di sisi lain, membiarkan email Anda di server mungkin lebih aman dari sudut pandang keandalan. Dengan POP, semua email Anda ada di komputer lokal Anda, dan jika sesuatu terjadi pada hard disk komputer Anda, Anda berisiko kehilangan email Anda, sedangkan di IMAP, email berada di server, yang mungkin akan sering dicadangkan.

Ada kerugian pada IMAP. Ini sebagian besar karena IMAP harus lebih rumit. IMAP memerlukan sumber daya pemrosesan dan penyimpanan yang lebih besar. Ada permintaan yang lebih besar untuk komunikasi jaringan melalui IMAP. Selain itu, dengan email tertinggal di server di IMAP, keamanan mungkin menjadi perhatian. Meskipun server POP juga menyimpan email Anda, email tersebut dihapus segera setelah Anda siap melihatnya. Kekurangan lainnya tidak dibahas di sini karena lebih tidak jelas atau rumit.

Ketiga protokol yang dibahas di sini memperlakukan komunikasi pesan email bukan sebagai satu pesan tetapi sebagai rangkaian pesan yang ada antara dua server atau antara server dan klien. Dalam kasus SMTP, pesan menggunakan salah satu dari beberapa header pesan: MAIL, RCPT, dan DATA (seperti yang telah dibahas), serta HELO (sapaan dari klien ke server) dan QUIT (untuk mengakhiri koneksi). Protokol Kantor Pos memiliki pesan yang mencakup STAT (status), LIST (mengembalikan jumlah pesan yang menunggu), RETR untuk mengambil pesan, DELE untuk menghapus pesan (klien harus memberi tahu server untuk menghapus pesan), dan QUIT. Di antara pesan, server akan merespons dengan nilai status seperti OK. IMAP, karena lebih kompleks, memiliki lebih banyak jenis pesan, dan kami akan mengabaikan detail tersebut.

Saat ini, POP3 menjadi standar pilihan jika menggunakan POP. Server POP3 mendengarkan port 110 untuk koneksi. POP3 juga dapat terhubung menggunakan TLS/SSL, dalam hal ini menggunakan port 995. IMAP juga memiliki varian yang lebih baru, dengan IMAP2 berubah menjadi IMAP2bis, yang memungkinkan pengkodean MIME, dan yang terbaru IMAP4. IMAP menggunakan port 143, dan IMAP melalui SSL menggunakan port 993.

Shell dan telnet yang aman

Selama hari-hari awal ARPANET, sebagian besar perangkat di jaringan adalah host. Kami membedakan host dari komputer jarak jauh atau klien bahwa host adalah jenis komputer yang menawarkan layanan, sumber daya, atau akses ke aplikasi. Cara lain untuk melihat host adalah komputer yang dapat diakses dari jarak jauh melalui jaringan. Komputer

mainframe, misalnya, adalah komputer host, sedangkan komputer pribadi bukanlah komputer host hingga tahun 1990-an, ketika teknologi sistem operasi mengizinkannya. Saat ini, hampir semua sumber daya komputer dapat berfungsi sebagai host, sehingga istilah tersebut telah kehilangan sebagian artinya.

Bagaimana cara seseorang masuk dari jarak jauh ke komputer? Kami membutuhkan beberapa protokol untuk ini. Dua protokol masuk paling populer adalah Telnet dan SSH. Telnet mendahului SSH, yang dibuat pada tahun 1969, dan karena Telnet tidak aman, Telnet tidak digunakan lagi. Hampir setiap orang yang melakukan login jarak jauh akan menggunakan formulir aman seperti SSH.

Telnet menggunakan paket TCP dan berkomunikasi melalui port 23 (komputer remote dan host menggunakan port yang sama). Telnet pada dasarnya melakukan dua aktivitas. Yang pertama adalah memasukkan pengguna ke host melalui proses autentikasi host sendiri. Setelah masuk, Telnet beroperasi seperti sinyal TCP mentah di mana perintah dikirim ke host dan host mengeksekusi perintah hampir secara verbatim, mengembalikan output apa pun dari perintah ke klien. Namun, Telnet juga memiliki sejumlah perintah untuk mengubah perilaku ini. Perintah-perintah ini termasuk menggemakan setiap perintah dalam keluarannya, mentransmisikan perintah dan respons dalam biner, mengubah tipe terminal (dulu diperlukan ketika pengguna yang berbeda menggunakan berbagai jenis terminal bisu), mengubah kecepatan komunikasi (berguna saat berkomunikasi dengan MODEM), mengirim jeda baris, menekan jeda baris, dan memutuskan sambungan, antara lain. Host Telnet harus menjalankan program layanan Telnet untuk mengizinkan login Telnet.

Pada 1990-an, SSH dikembangkan untuk menggantikan Telnet. Secure Shell sangat mirip dengan fungsi Telnet, tetapi menggunakan enkripsi kunci publik untuk menyediakan komunikasi yang aman. Ada banyak mekanisme dimana kunci publik dapat dipertahankan dan dibagikan di antara pengguna. Misalnya, dalam sistem Unix, kunci publik-privat resmi disimpan di direktori home masing-masing pengguna. Baru-baru ini, SSH versi 2 (SSH-2) menambahkan fungsionalitas pertukaran kunci, di mana kunci pribadi pengguna dapat ditukar dengan aman. Pertukaran kunci SSH-2 didasarkan pada algoritme yang dikembangkan oleh Ralph Merkel dan diimplementasikan oleh Whitfield Diffie dan Martin Hellman (sehingga disebut pertukaran kunci Diffie–Hellman atau D–H).

Pesan SSH dikirim menggunakan paket TCP, seperti Telnet, dan berkomunikasi melalui port 22. Host SSH harus menjalankan program layanan SSH untuk mengizinkan login SSH. Di Linux/Unix, program ini adalah sshd. Dengan terbentuknya saluran SSH, seseorang dapat menyediakan bentuk komunikasi lain dari dalam saluran tersebut. Ini adalah konsep yang dikenal sebagai tunneling. Misalnya, setelah melakukan ssh ke komputer, Anda kemudian dapat membuka koneksi FTP ke komputer tersebut. Ini dikenal sebagai SFTP. Komunikasi FTP ada di dalam terowongan SSH. Di Unix/Linux, rcp memungkinkan pengguna untuk menyalin file dari satu komputer jaringan ke komputer lainnya. Dari dalam terowongan SSH, ini dikenal sebagai SCP (salinan aman).

Selain Telnet dan SSH, protokol login jarak jauh lainnya adalah rlogin, khusus untuk sistem Unix/Linux. Rlogin adalah salah satu dari banyak utilitas r. Program-program ini beroperasi pada jaringan yang sumber dayanya berbagi server autentikasi yang sama,

sehingga pengguna yang memiliki akun di satu mesin memiliki akun yang sama di setiap mesin. Ini sering digunakan di jaringan area lokal workstation tanpa disk, di mana semua workstation menggunakan file server yang sama. Rlogin sendiri adalah Telnet yang setara dengan r-utilitas. Hal yang menyenangkan tentang rlogin adalah sekali masuk ke satu komputer jaringan, Anda dapat masuk ke komputer lain, tanpa melalui proses otentikasi lagi. Namun, seperti Telnet, rlogin tidak aman (walaupun akan ada perdebatan mengenai apakah Anda memerlukan keamanan di jaringan komputer yang kecil).

5.3 LAPISAN TRANSPORTASI

Di lapisan aplikasi, aplikasi telah menghasilkan pesan awal di dalam protokolnya sendiri. Apa yang terjadi dengan pesan itu sekarang? Ada empat hal yang harus dilakukan. Pertama, koneksi harus dibuat antara mesin sumber dan tujuan. Ini dicapai melalui jabat tangan. Jabat tangan adalah proses jaringan dasar di mana komputer sumber mengirimkan permintaan komunikasi ke tujuan dan komputer tujuan merespons. Di TCP, jabat tangan adalah proses tiga langkah. Kami secara singkat melihat ini di sub-bagian pertama bab ini. Dengan sesi yang sekarang ditetapkan, perangkat sumber bebas untuk memulai komunikasinya. Pesan, seperti yang diproduksi di lapisan aplikasi, sekarang harus disiapkan untuk transmisi. Ini berarti bahwa pesan harus disegmentasi menjadi satu atau lebih potongan. Peran kedua dari lapisan transport adalah membuat potongan-potongan ini, yang dikenal sebagai paket (atau datagram). Di TCP/IP, ada dua bentuk paket yang populer: TCP dan UDP. Ada jenis paket lain yang tersedia, yang akan kita lihat secara singkat. Lapisan transport akan membuat paket-paket awal ini, menandai setiap paket dengan nomor urut (khusus paket TCP) dan checksum.

Di sisi penerima, merupakan tanggung jawab lapisan transport untuk menyusun paket-paket secara berurutan, berdasarkan nomor urut. Karena paket-paket tersebut mungkin telah tiba dengan tidak sesuai urutan, paket-paket tersebut harus di-buffer sampai seluruh rangkaian paket selesai (khusus paket TCP). Penerima juga bertanggung jawab untuk memastikan pengiriman paket yang akurat. Dengan demikian, pada layer ini penerima akan memeriksa checksum dan jumlah paket yang diterima. Jika salah satunya tidak akurat, lapisan transportlah yang akan meminta agar paket tertentu (atau beberapa paket) ditransmisikan ulang.

Lapisan transport juga bertanggung jawab untuk kontrol aliran dan multiplexing. Dalam flow control, ada kemungkinan bahwa transport layer dapat menunda pengiriman paket ketika kepadatan jaringan tinggi. Itu juga dapat meminta komputer tujuan merespons pada tingkat yang lebih lambat jika atau ketika paket datang terlalu cepat untuk diproses di lapisan aplikasi, sehingga paket meluap dari buffer. Multiplexing dapat terjadi pada dua lapisan: lapisan transport dan lapisan tautan. Pada lapisan transport, multiplexing dilakukan dengan mengirimkan paket melalui port yang berbeda. Ini mungkin terjadi jika pesan ditangani oleh dua protokol/aplikasi yang berbeda seperti membuat koneksi FTP dan mengirim data FTP.

Tanggung jawab akhir dari transport layer adalah memelihara koneksi yang telah dibuat. Koneksi biasanya dibiarkan terbuka sementara paket dan ucapan terima kasih

diperdagangkan bolak-balik. Menutup koneksi biasanya terjadi dengan perintah eksplisit dari sumber daya, yang menunjukkan bahwa sesi harus ditutup, atau karena batas waktu. Timeout terjadi ketika tidak ada pesan yang diterima selama beberapa waktu yang ditentukan sebelumnya. Dengan cara ini, jika terjadi sesuatu pada konektivitas satu sumber daya, koneksi tidak dipertahankan tanpa batas waktu.

Pada bagian ini, kami berkonsentrasi pada tiga aspek dari lapisan transport. Aspek pertama adalah membangun, memelihara, dan menutup koneksi. Ini adalah tugas lapisan pertama dan keempat. Selanjutnya, kita melihat paket. Kami berkonsentrasi pada dua bentuk paket yang dominan, TCP dan UDP, dan kami juga secara singkat mempertimbangkan yang lain. Akhirnya, kami melihat kontrol aliran dan multiplexing.

Protokol kontrol transmisi handshake dan koneksi

handshake jaringan adalah peristiwa terkenal, hadir di semua protokol jaringan. Jabat tangan tradisional melibatkan dua langkah: permintaan dan pengakuan. Sumber daya pengirim mengirimkan maksudnya untuk berkomunikasi dengan sumber daya tujuan. Saat menerima permintaan, jika sumber daya tujuan tersedia dan siap, ia merespons dengan pengakuan. Di TCP/IP, langkah ketiga ditambahkan. Setelah menerima pengakuan, sumber juga mengakui. Alasan untuk langkah ketiga adalah memberi tahu tujuan bahwa sumbernya siap untuk memulai. Dalam paket TCP, ketiga langkah ini masing-masing disebut sebagai SYN, SYN-ACK, dan ACK. Ada dua bit khusus dalam paket TCP untuk menunjukkan apakah paket tersebut adalah SYN atau ACK. Kedua bit disetel untuk SYN-ACK. Dengan menggunakan tiga pesan yang berbeda, masing-masing sumber tahu jika sebuah sesi sedang dibuka (SYN), disinkronkan (SYN-ACK), atau dibuat (ACK). Selain bit jabat tangan, paket dapat menyertakan nomor pengakuan, yang menunjukkan jumlah byte yang diterima sebelumnya selama komunikasi khusus ini.

Dengan sesi yang dibuat, sesi tetap terbuka untuk beberapa waktu. Jumlah waktu ditunjukkan oleh dua kondisi. Kondisi pertama adalah sesi terbuka hingga ditutup secara eksplisit. Untuk menutup koneksi, sumber daya sumber melakukan jabat tangan tiga arah lainnya. Sumber mengirimkan sinyal FIN ke tujuan. Tujuan mengakui paket FIN-ACK. Terakhir, sumber merespons dengan ACK-nya sendiri. Alternatifnya, tujuan dapat merespons dengan dua pesan terpisah, satu dengan ACK dan satu lagi dengan FIN. Selama sumber menunggu tanggapan terhadap sinyal FIN-nya, koneksi tetap terbuka; namun, dalam kasus ini, dikenal sebagai setengah terbuka, karena saluran sekarang hanya tersedia untuk ditutup, namun port tidak dapat digunakan kembali hingga sambungan benar-benar tertutup.

Cara kedua agar koneksi ditutup adalah melalui batas waktu. Ada dua bentuk time-out. Ketika koneksi dibuat, waktu tetap hidup diatur. Setiap kali paket diterima, pengatur waktu ini diatur ulang. Jika tidak, pengatur waktu ini menghitung mundur waktu saat koneksi tidak digunakan. Saat mencapai 0, sumber daya memiliki dua opsi. Itu dapat mengasumsikan bahwa koneksi telah ditinggalkan dan menutupnya, atau dapat mencoba untuk membangun kembali koneksi tersebut. Di TCP, waktu tetap hidup default adalah 2 jam. Namun, beberapa aplikasi mungkin mengesampingkan ini untuk waktu yang lebih singkat. Misalnya, koneksi HTTP mungkin menggunakan waktu hidup 1–2 menit.

Di Unix dan Linux, tiga nilai batas waktu berguna diatur secara otomatis. Yang pertama adalah waktu tcp_keepalive_. Seperti namanya, nilai ini (dalam detik) adalah berapa lama koneksi tetap terbuka sementara tidak ada pesan yang dikirim atau diterima. Saat mencapai batas waktu ini, komputer kemudian mengeluarkan probe keepalive. Probe adalah permintaan ke sumber daya lain untuk memastikan bahwa itu masih tersedia. Jika pengakuan ke probe diterima, pengatur waktu tetap hidup diatur ulang. Nilai kedua di Unix/Linux adalah tcp_keep_alive_intvl. Ini adalah satuan waktu lain yang digunakan sehingga, jika probe pertama tidak berhasil, jumlah waktu ini harus berlalu sebelum mengirim probe kedua. Terakhir, tcp_keepalive_probes adalah bilangan bulat dari jumlah upaya pemeriksaan yang harus dicoba sebelum komputer ini memutuskan bahwa sesi telah ditinggalkan di ujung lain dan sambungan harus ditutup. Di RedHat Linux, 3 nilai ini masing-masing ditetapkan ke 7200, 75, dan 9. Jadi, setelah 2 jam (7200 detik) berlalu, komputer Linux mulai menyelidiki sumber daya lain untuk mendapat tanggapan. Probing akan berlangsung hingga sembilan kali selama 75 detik per probe, atau gabungan 11¼ menit.

Paket UDP dikenal sebagai connectionless. Saat mentransmisikan paket UDP, tidak ada jabat tangan untuk menjalin komunikasi, atau jabat tangan untuk mengakhiri komunikasi. Sebaliknya, paket dikirim oleh satu sumber daya ke sumber daya lainnya. Apakah diterima atau tidak, pengirim terus mengirim, kecuali sumber penerima berkomunikasi kembali ke sumber untuk menghentikan transmisi.

Datagram: protokol kontrol transmisi, protokol datagram pengguna, dan lainnya

Paket TCP memiliki nomor urut dan dikirim hanya setelah sesi dibuat, tetapi paket UDP tidak terhubung. Mengapa kita harus menggunakan UDP ketika UDP tidak berusaha memastikan penerimaan paket dan melakukannya tanpa membuat koneksi sebelumnya? Jawabannya adalah beberapa informasi harus disampaikan lebih cepat dengan mengorbankan pengiriman yang andal. Ini terutama benar ketika paket-paket tersebut merupakan data audio atau video yang dialirkan secara real time. Pengorbanan yang dilakukan UDP sedemikian rupa sehingga paket menjadi lebih kecil dan dapat ditransmisikan lebih cepat. Selanjutnya, paket yang diterima dapat diproses segera daripada menunggu semua paket tiba.

Pada lapisan transport, segmen data lebih dikenal sebagai datagram daripada paket. Sebagian besar komunikasi melalui Internet menggunakan datagram TCP. Ini termasuk, misalnya, semua bentuk protokol email, HTTP, HTTPS, FTP, Telnet, dan SSH. Itu tidak termasuk permintaan DNS, permintaan DHCP, pesan NTP, dan streaming data menggunakan *Real-Time Transport Protocol* (RTP), *Real-Time Streaming Protocol* (RTSP), *Internet Protocol Television* (IPTV), atau *Routing Information Protocol* (RIP) . Tampaknya berlawanan dengan intuisi bahwa pesan penting yang dikirim ke server nama DNS atau server DHCP dikirim dengan menggunakan UDP, sehingga mengabaikan keandalan TCP. Namun, idenya adalah jika tidak ada pesan pengakuan, sumber dapat mengirim pesan lagi atau mencoba server lain. Dalam kasus email, HTTP atau SSH, paket yang tidak sampai ke tujuannya akan menyebabkan kerusakan data. Ini tidak bisa dibiarkan terjadi. Paket yang dijatuhkan selama streaming adalah bentuk lain dari kerusakan data, tetapi ini dapat diterima karena satu atau beberapa paket yang dijatuhkan tidak akan terlalu memengaruhi kualitas audio atau video yang dialirkan. Faktor yang lebih penting dalam streaming adalah bahwa paket tiba tepat waktu.

Mari kita lihat struktur kedua jenis datagram ini. Ingatlah bahwa bagian data dari pesan kita telah disusun oleh perangkat lunak aplikasi di lapisan aplikasi. Aplikasi tujuan akan memberi tahu kami nomor port tujuan. Lapisan transport akan menghasilkan nomor port sumber, seringkali menggunakan nomor port unreserved apa pun yang tersedia berikutnya. Kami akan membahas ini segera. Gambar 5.8 mengilustrasikan struktur paket TCP dan UDP. Dalam hal ini, kami hanya berfokus pada header TCP dan UDP. Potongan informasi ini ditambahkan ke data yang sudah dikemas dari lapisan aplikasi.

Perhatikan seberapa pendek datagram UDP daripada datagram TCP. Sebagian besar informasi yang menyusun datagram TCP adalah urutan dan nomor pengakuan. Nomor-nomor ini digunakan untuk memastikan pengiriman semua paket dan menyusun kembali paket-paket itu dalam urutan yang benar. Karena UDP tidak menjamin pengiriman, nilai ini dihilangkan. Di sisi lain, keduanya memiliki port sumber dan tujuan, checksum, dan ukuran (dalam datagram TCP, ukurannya disebut ukuran jendela). Ada juga bendera dan bidang pointer mendesak. Kami akan mengeksplorasi ini secara lebih rinci di bawah ini.

Gambar 5.8 TCP (kiri) versus header datagram UDP (kanan).

Mari kita mulai dengan dasar-dasarnya. Apa pun jenisnya, datagram memerlukan sumber dan nomor port tujuan. Keduanya 16 bit. Dengan 16 bit untuk alamat port, memungkinkan maksimal 65.356 port yang berbeda, bernomor 0 sampai 65.355. Dari alamat ini, 1024 port pertama dicadangkan untuk apa yang dikenal sebagai port terkenal. Nomor port ini dialokasikan ke, atau dicadangkan untuk, aplikasi terkenal. Dari 1024 port terkenal, banyak yang saat ini tidak digunakan, memungkinkan untuk pertumbuhan di masa mendatang. Misalnya, port 14–16, 26, 28, dan 30 tidak terikat pada aplikasi apa pun. Port bernomor 1024 hingga 49151 adalah port terdaftar. Port ini dapat disediakan berdasarkan permintaan ke layanan. Beberapa terikat pada aplikasi tertentu seperti port 1025 untuk NFS (Sistem File Jaringan Unix) dan 1194 untuk OpenVPN. Port yang tersisa, 49152 hingga 65355, tersedia untuk ditetapkan secara dinamis. Bagaimana ini digunakan? Saat aplikasi ingin membuka koneksi, port tujuan harus salah satu alamat port yang terkenal (misal 80 untuk HTTP), tetapi port sumber harus menjadi port dinamis berikutnya yang tersedia. Jika 49152 adalah port

terbaru yang ditetapkan, maka port sumber mungkin 49153. Dalam praktiknya, lapisan transport mungkin menetapkan salah satu dari 1024 hingga 65355 sebagai alamat port sumber.

Jadi, kita telah melihat terdiri dari setengah dari header paket UDP dan sebagian dari header paket TCP. Kita juga telah melihat apa itu checksum (lihat Bab 1). Dalam hal ini, panjang checksum adalah 16 bit, artinya setelah dihitung (menggunakan penjumlahan semua byte atau pendekatan CRC), nilainya dibagi dengan 65.356 dan sisanya dimasukkan ke dalam bidang ini. Checksum yang ditempatkan di header dihasilkan dengan menggabungkan data dari datagram dengan sisa informasi header. Dengan cara ini, checksum tidak hanya terikat pada data tetapi juga seluruh paket. Bidang panjang adalah ukuran, dalam byte, dari bagian data paket dan header. Jadi, untuk paket UDP, ini akan menjadi 8 byte lebih besar dari ukuran data. Untuk paket TCP, ukurannya akan bervariasi tergantung pada apakah paket menyertakan bidang opsional yang mengikuti penunjuk mendesak.

Sekarang, mari kita jelajahi bidang yang ditemukan di datagram TCP yang tidak ada di datagram UDP. Untuk memastikan pengiriman paket yang tepat, TCP menggunakan nomor urut dan nomor pengakuan. Saat paket dikirim dari satu lokasi ke lokasi lain, sumber daya tujuan akan mengirimkan paket pengakuan kembali ke sumbernya. Mari kita asumsikan bahwa kita memiliki komunikasi klien-server, seperti klien yang meminta halaman web dari server web. Server web telah membagi halaman menjadi beberapa paket TCP. Setiap paket diberi nomor dengan nomor urut. Angka ini adalah nilai 4-byte, yang menunjukkan byte pertama dari bagian data ini. Misalnya, jika ada tiga paket yang dikirim, masing-masing menyimpan 1500 byte data, nomor urut paket pertama adalah 0, nomor urut paket kedua adalah 1500, dan nomor urut paket ketiga adalah 3000. paket, komputer penerima mengirimkan pengakuan, yang mencakup nomor pengakuan. Nomor ini akan menjadi satu lebih besar dari nomor urut yang ditanggapi.

Mari kita asumsikan bahwa penerima menerima ketiga paket data TCP dengan nomor urut 0, 1500, dan 3000. Penerima kemudian mengirimkan pemberitahuan kembali ke server dengan nomor pemberitahuan 3001. Alasannya adalah untuk menunjukkan kepada server tempat ketiga paket diterima. Jika nomor pengakuan adalah 1 atau 1501, maka server akan mengetahui bahwa setidaknya satu paket tidak diterima. Jika nomor pengakuan kembali adalah 1501, server akan mengirim ulang paket ketiga. Jika nomor pengakuan yang dikembalikan adalah 1, server memiliki dua kemungkinan tindakan: ia dapat mengirim ulang paket kedua saja atau paket kedua dan ketiga. Dengan asumsi bahwa itu hanya mengirim paket kedua, jika klien telah menerima paket pertama dan ketiga, maka itu akan merespons dengan nomor pengakuan 3001, yang menunjukkan bahwa ketiga paket telah tiba. Jika tidak, itu akan mengirimkan nomor pengakuan 1501, sehingga server dapat mengirim paket ketiga. Nyatanya, skema yang dijelaskan di sini tidak akurat. Komputer asli (klien dalam contoh kami) mengirimkan nomor urut awal untuk digunakan server dalam paket pengembaliannya. Angka ini dihasilkan secara acak (algoritme angka acak yang digunakan untuk situasi ini dibangun ke dalam sistem operasi).

Jika Anda perhatikan, bidang nomor urut dan pengakuan dari paket TCP adalah masing-masing 4 byte. Dengan 4 byte (32 bit), terdapat 232 nilai berbeda yang dapat disimpan.

Ini sedikit lebih dari 4 miliar. Alasan untuk nomor urut yang dibuat secara acak adalah untuk membantu memastikan bahwa permintaan awal yang dibuat adalah permintaan yang sah daripada upaya membajak paket TCP. Sisa bidang header datagram TCP adalah bendera status dan pointer mendesak. Kami telah menyebutkan penggunaan beberapa bendera status: ACK, SYN, dan FIN. Ketiga flag ini (masing-masing satu bit) menunjukkan apakah paket TCP adalah paket SYN, paket SYN-ACK, paket ACK, paket FIN, atau paket FIN-ACK. Ada enam bidang 1-bit lainnya, seperti yang dijelaskan pada Tabel 3.3. Sisa flag status 16-bit ini terdiri dari ukuran 4-bit dari header TCP dalam potongan berukuran 32-bit (32 bit sama dengan 4 byte atau biasanya satu kata) dan 3 bit diatur ke 000 (3 bit ini disisihkan untuk penggunaan masa depan). Ukuran minimum header TCP adalah lima kata (20 byte), dan ukuran maksimumnya adalah 15 kata (60 byte). Variabilitas tergantung pada jika dan berapa banyak opsi yang mengikuti penunjuk mendesak.

Tabel 5.3 Bendera Status Termasuk ACK, SYN, FIN

Bidang Bit	Singkatan	Arti
7	NS	Eksperimental untuk perlindungan penyembunyian
8	CWR	Jendela Kemacetan Dikurangi—jika diatur,
9	ECE	menunjukkan bahwa paket ECE telah diterima
10	URG	Jika SYN adalah 1, maka paket TCP kompatibel dengan NS; jika tidak, sebuah paket dengan CWR diterima selama lalu lintas yang tidak berat
11	ACK	Menunjukkan bahwa pointer mendesak digunakan
12	PSH	Seperti yang dijelaskan sebelumnya
13	RST	Fungsi push untuk memindahkan data yang di-buffer ke aplikasi
14	SYN	Setel ulang koneksi
15	FIN	Seperti yang dijelaskan sebelumnya

Penunjuk urgensi, jika digunakan, berisi nilai 16-bit, yang menunjukkan offset ke nomor urut untuk byte data urgensi terakhir. Data yang diindikasikan sebagai mendesak ditandai sedemikian rupa untuk memberi tahu aplikasi yang menunggu untuk segera mulai memproses data mendesak daripada menunggu semua paket tiba. Jika data mendesak hanya sebagian dari paket yang diberikan, maka aplikasi diminta untuk segera memproses data sampai ke lokasi ini dalam paket tetapi tidak memproses data apa pun yang mengikuti lokasi ini.

Penggunaan penunjuk mendesak dan data mendesak belum tentu terkait dengan penggunaan Internet modern kita. Sebagai gantinya, bayangkan seorang pengguna yang menggunakan Telnet di komputer jarak jauh. Saat memasuki beberapa operasi, pengguna memutuskan untuk membatalkan koneksi. Perintah batalkan lebih mendesak daripada bagian lain dari data paket saat ini yang dikirim dari komputer lokal ke komputer jarak jauh. Seperti yang kami katakan sebelumnya, datagram TCP mungkin memiliki beberapa bidang opsional hingga 40 byte (10 kata). Setiap opsi akan terdiri antara satu dan tiga bagian. Pertama adalah tipe, deskriptor 1-byte. Kedua adalah panjangnya, yang bersifat opsional. Ketiga, juga

opsional, adalah data opsi apa saja. Sebagian besar opsi akan berukuran 3 byte. Namun, opsi NO-OP (no operation) dan End-Of-Options hanyalah tipe (1 byte) sendiri. Opsi lain tercantum dalam Tabel 5.4 bersama dengan data yang mungkin disertakan dan ukurannya. Perhatikan bahwa ukuran dalam byte adalah ukuran bidang data itu sendiri. Misalnya, opsi Ukuran Segmen Maksimum akan memiliki panjang 6 byte: 1 byte untuk tipe, 1 byte untuk ukuran, dan 4 byte untuk bidang data.

Ingatlah bahwa UDP tidak terhubung. TCP mempertahankan koneksi. Koneksi TCP dapat berada di salah satu dari beberapa status yang berbeda. Status ini ditentukan oleh kata kunci, seperti yang tercantum dalam Tabel 5.5. Anda mungkin melihat beberapa kata ini muncul saat menggunakan berbagai perintah jaringan. Misalnya, di Unix/Linux, perintah netstat akan menampilkan status setiap koneksi TCP.

Meskipun sebagian besar komunikasi TCP/IP akan menggunakan datagram TCP atau UDP, ada jenis lain yang tersedia. Pada tahun 2000, *Stream Control Transmission Protocol* (SCTP) diperkenalkan. Ini menawarkan kesederhanaan transmisi tanpa koneksi UDP sambil memastikan transportasi TCP yang berurutan.

Setiap paket SCTP terdiri dari dua bagian: header dan data. Header, seperti halnya UDP, berisi port sumber dan port tujuan serta checksum. Selain itu, terdapat tag verifikasi 32-bit, yang digunakan untuk menunjukkan apakah paket ini merupakan bagian dari transmisi saat ini atau paket lama yang sudah kedaluwarsa. Nilai tag verifikasi awal dihasilkan secara acak. Paket yang berurutan memiliki nilai yang lebih besar. Paket yang diterima yang nomornya tidak berada dalam urutan nilai verifikasi dari yang mengikuti dianggap sebagai paket basi dan dibuang.

Tabel 5.4 Opsi TCP

Jenis opsi	Ukuran dalam byte	Data
Ukuran segmen maksimum	4	Ukuran maksimum yang diperbolehkan untuk paket tcp ini.
Pengakuan selektif	4	Penerima menggunakan bidang ini untuk mengakui penerimaan paket perantara (misalnya, jika paket 1 hilang tetapi paket 2-100 diterima, bidang ini digunakan). Data membentuk nomor urut paket pertama dan terakhir yang diterima (mis., 2 dan 100).
Pengakuan selektif diizinkan	2	Digunakan dalam pernyataan syn untuk mengaktifkan fitur ini selama komunikasi.
Stempel waktu	10	Paket tcp diberi stempel waktu (menggunakan beberapa nilai awal acak), sehingga paket dapat dipesan berdasarkan stempel waktu. Jika nomor urut besar dan melebihi 232, maka nomor urut akan berputar untuk memulai kembali dari 0. Stempel waktu kemudian dapat digunakan untuk memesan paket. Kolom ini

		menunjukkan bahwa stempel waktu harus dikirim dan dikembalikan dalam ucapan terima kasih apa pun.
Skala jendela	3	Secara default, ukuran data paket tcp dibatasi hingga 65.536 byte. Dengan field ini, ukuran ini bisa ditimpa menjadi sebesar 1 gbyte. Ini hanya digunakan dalam komunikasi jaringan bandwidth tinggi.

Mengikuti 12 byte ini (dua port, tag verifikasi, dan checksum) adalah serangkaian potongan data. Setiap potongan memiliki tipe, bendera, dan nilai panjang, diikuti oleh datanya. Bisa ada sejumlah bongkahan; setiap potongan akan menunjukkan jenis informasi yang berbeda. Jenis termasuk potongan data (data sebenarnya); inisiasi komunikasi; pengakuan inisiasi; permintaan detak jantung (apakah Anda masih merespons?) dan pengakuan detak jantung; batalkan, kesalahan, dan data cookie atau pengakuan; dan shutdown (potongan terakhir) dan shutdown pengakuan. Hanya ada empat bendera; namun ada ruang hingga delapan bendera. Panjang menunjukkan jumlah byte yang mengikuti bagian data chunk.

Saat ini, tampaknya belum ada sistem operasi yang mengimplementasikan SCTP, tetapi dimungkinkan untuk membuat tunnel SCTP dari dalam UDP. Lalu, mengapa SCTP harus ada? Ini menawarkan beberapa keunggulan dibandingkan TCP. Yakni, ia tidak menggunakan skema sekuensing yang rumit seperti TCP, sambil tetap menawarkan sekuensing, sehingga paket, dijatuhkan atau dikirim, akan ditampilkan dalam urutan yang tepat. Itu juga mampu melakukan operasi push, sehingga aplikasi yang menunggu akan mulai memproses paket yang dikirimkan. Akhirnya, sementara TCP rentan terhadap penolakan serangan layanan, SCTP berupaya menghindarinya.

Protokol Kontrol Kemacetan Datagram (DCCP) memiliki kontrol aliran yang mirip dengan TCP tetapi tanpa batasan keandalan TCP. Tujuan utama DCCP adalah untuk memastikan pengiriman data secara real time ketika aplikasi, seperti streaming audio atau video, memiliki batasan waktu tertentu. Tidak seperti UDP, di mana paket dapat dibuang dan diabaikan begitu saja, DCCP memberi stempel waktu pada datagram sehingga datagram yang kedaluwarsa dapat dibuang tetapi diproses. Seperti TCP, datagram DCCP diakui pada saat diterima.

Jenis datagram lainnya adalah Reliable User Datagram, bagian dari Reliable UDP (RUDP). Datagram ini merupakan perpanjangan dari datagram UDP. Itu menambahkan bidang untuk menunjukkan pengakuan dari paket yang diterima dan memiliki fitur tambahan yang memungkinkan pengiriman ulang paket yang hilang serta windowing dan kontrol aliran yang fleksibel. Diperkenalkan oleh Cisco, RUDP merupakan upaya untuk menjembatani kesenjangan antara kurangnya keandalan dalam UDP dan kompleksitas TCP.

Dua protokol tambahan dengan jenis paketnya sendiri adalah *Resource Reservation Protocol* (RSVP), yang digunakan untuk mengirim paket kontrol untuk menerima informasi tentang keadaan jaringan, dan Venturi Transport Protocol (VTP), protokol berpemilik yang digunakan oleh Verizon untuk transportasi data nirkabel. VTP, seperti RUDP, juga berusaha

memperbaiki beberapa inefisiensi TCP. Meskipun ada protokol transport lain dengan format datagramnya sendiri, kami menghilangkannya di sini. Kami akan merujuk ke TCP dan UDP di hampir setiap kasus sepanjang sisa buku teks ini.

Tabel 5.5 Status TCP

Negara	Arti
CLOSED	Tidak ada koneksi
CLOSE-WAIT	Menunggu pernyataan dekat dari tuan rumah lokal
CLOSING	Menutup tetapi menunggu pengakuan dari host jarak jauh
ESTABLISHED	Jabat tangan selesai, koneksi terjalin, dan tidak kehabisan waktu
FIN-WAIT-1	Dari ujung server, koneksi aktif tetapi saat ini tidak digunakan
FIN-WAIT-2	Klien telah menerima sinyal FIN dari server untuk menutup koneksi
LAST-ACK	Server sedang mengirimkan sinyal FIN-nya sendiri
LISTEN	Untuk server, port terbuka dan menunggu koneksi masuk
SYN-RECEIVED	Server telah menerima SYN dari klien
SYN-SENT	Pembukaan koneksi, menunggu sinyal ACK/SYN-ACK
TIME-WAIT	Dari ujung klien, koneksi aktif tetapi saat ini tidak digunakan

Pengendalian Aliran Dan Multipleks

Tugas lapisan transport terakhir adalah menangani transmisi dan penerimaan beberapa paket dengan cara yang tumpang tindih. Banyak, dan mungkin sebagian besar, pesan yang dikirim oleh aplikasi akan didekomposisi menjadi beberapa paket. TCP memastikan tidak hanya pengiriman paket yang andal tetapi juga pemesanan paket. Pemesanan ditangani melalui nomor urut. Di ujung penerima, paket ditempatkan ke dalam buffer. Ini adalah tugas lapisan transport untuk menempatkan paket-paket ini dalam urutan yang benar di buffer ini. Ini dikenal sebagai kontrol aliran. Untuk menangani kontrol aliran, banyak teknik berbeda yang tersedia. Kami menjelajahinya di sini.

Untuk lebih memahami kontrol aliran, kami membuat contoh. Klien web telah meminta halaman web dari server web. Server web mengirimkan kembali 10 paket yang berisi data halaman. Peramban web tidak melakukan apa pun dengan paket sampai semua 10 paket diterima. Tapi apa yang dilakukan lapisan transport saat setiap paket tiba? Hal ini tergantung pada metode kontrol aliran yang digunakan.

Metode kontrol aliran yang paling sederhana, tetapi paling tidak efisien dikenal sebagai stop and wait. Dalam mode ini, setiap paket dikirim satu per satu, di mana pengirim menunggu pengakuan sebelum mengirim paket berikutnya. Hal ini memastikan bahwa paket tidak akan datang rusak karena paket berikutnya tidak dikirim sampai penerimaan paket sebelumnya telah diakui. Jika pengakuan itu tidak kembali dalam waktu yang wajar, pengirim mengirimkannya kembali. Waktu di mana server menunggu pengakuan adalah interval waktu habis. Gagasan bahwa sebuah paket harus ditransmisikan ulang dikenal sebagai permintaan pengulangan otomatis (ARQ), yang merupakan tanggung jawab dari lapisan transport. Dalam contoh kami, server web mengirimkan setiap paket, satu per satu, dan dengan demikian,

pendekatan ini kemungkinan akan memakan waktu paling lama bagi klien untuk menerima halaman web lengkap.

Alih-alih membatasi transmisi ke satu paket pada satu waktu, varian ini dikenal sebagai sliding window. Dalam hal ini, penerima memberi tahu pengirim tentang ukuran jendela dalam paket. Ini memungkinkan pengirim untuk mengirim sejumlah paket sebelum berhenti. Penerima mengakui telah menerima paket, seperti yang telah kita bahas di Bagian 3.3.2. Artinya, penerima akan mengembalikan nomor pengakuan dari paket berikutnya yang diharapkan. Jika salah satu paket yang dikirim oleh server tidak diterima, penerima mengirimkan nomor pengakuan yang berbeda, yang menunjukkan bahwa pengirim harus mengirim ulang beberapa paket. Ini dikenal sebagai ARQ pengulangan selektif. Permintaan berulang juga dapat dikeluarkan untuk data yang salah (checksum salah), dalam hal ini seluruh jendela dapat dikirim ulang. Ukuran jendela dalam skema ini tidak sama dengan buffer perangkat lunak aplikasi. Sebaliknya, ukurannya akan lebih kecil. Ini memungkinkan lapisan transport untuk memilih ukuran yang masuk akal dan meningkatkan ukuran ini jika transmisi berjalan tanpa masalah. Bentuk flow control ini lebih baik daripada stop and wait, karena ada beberapa tumpang tindih dalam transmisi paket. Namun, jumlah waktu tunggu klien dipengaruhi oleh ukuran jendela.

Kontrol aliran loop tertutup memungkinkan perangkat di jaringan melaporkan kemacetan jaringan kembali ke pengirim. Umpan balik semacam itu memungkinkan pengirim mengubah perilakunya untuk memanfaatkan jaringan dengan lebih efisien atau mengurangi lalu lintas. Pendekatan ini mendukung pengendalian kemacetan. Jadi, jika jendela geser delapan digunakan dan server menerima laporan bahwa jaringan macet, server dapat mundur pada ukuran jendela ke angka yang lebih kecil. Jelas, jika ukuran jendela diturunkan, klien harus menunggu lebih lama, tetapi keuntungannya di sini adalah berpotensi menurunkan jumlah paket yang dijatuhkan, sehingga pada akhirnya, klien sebenarnya memiliki waktu tunggu yang lebih singkat.

Bentuk lain dari flow control adalah untuk mengontrol kecepatan transmisi aktual. Ini mungkin terjadi, misalnya, ketika dua komputer berkomunikasi dengan MODEM atau antara komputer dan elemen switching. Tingkat transmisi umum harus ditetapkan.

Multiplexing, secara umum, adalah gagasan bahwa sekumpulan data dapat berasal dari berbagai sumber dan kita harus memilih sumber untuk menerima data. Ada dua proses, multiplexing dan demultiplexing. Dalam telekomunikasi, multiplexing dapat terjadi dalam beberapa bentuk yang berbeda. Beberapa bentuk multiplexing terjadi pada lapisan tautan, jadi kami menunda pembahasannya untuk saat ini. Salah satu bentuk multiplexing ada di lapisan transport, yang dikenal sebagai port multiplexing. Idenya adalah bahwa klien mungkin ingin mengirim beberapa permintaan ke server. Permintaan tersebut dapat dikirim dari port sumber yang berbeda. Karena server mengharapkan untuk menerima permintaan melalui port tunggal, paket permintaan akan dikirim ke tujuan yang sama. Namun, respons server akan dikirim kembali ke klien ke port yang diindikasikan sebagai sumber yang berbeda.

Misalnya, bayangkan browser web telah mengambil halaman web dari server. Halaman web memiliki banyak komponen yang disimpan dalam file yang berbeda (misalnya, file gambar, lembar gaya kaskade, dan file HTML itu sendiri). Browser web akan membuat

permintaan terpisah, satu per file. Setiap permintaan ini dikirim ke server web yang sama dan kemungkinan besar ke port 80. Dalam multiplexing port, setiap permintaan ini dikirim dari port yang berbeda, misalnya port 1024, 1025, 1026, dan 1027.

Tampaknya tidak ada gunanya membagi komunikasi dengan cara ini. Jadi, mengapa tidak menggunakan satu port saja? Pertimbangkan server web yang didistribusikan di beberapa komputer berbeda. Jika masing-masing server menerima salah satu permintaan dan masing-masing merespons kira-kira pada waktu yang sama, maka klien mungkin menerima file dengan cara yang tumpang tindih. Klien dapat mulai menyatukan file saat paket diterima, bergantian di antara file. Jika semua permintaan dikirim ke satu port sumber, permintaan tersebut harus diserialkan sehingga hanya satu permintaan yang dapat dipenuhi pada satu waktu, secara keseluruhan, sebelum permintaan berikutnya dapat mulai ditangani. Dengan demikian, multiplexing pada tingkat port memungkinkan jaringan Anda memanfaatkan paralelisme inheren yang tersedia melalui Internet dengan bekerja pada beberapa komunikasi yang berbeda pada waktu yang bersamaan.

5.4 LAPISAN INTERNET

Lapisan Internet adalah lapisan atas IP. Pada lapisan ini, datagram (paket) dari lapisan sebelumnya ditangani. Ini membutuhkan tiga proses yang berbeda. Pertama, paket harus dialamatkan. Dalam IP, dua bentuk pengalamatan digunakan: IPv4 dan IPv6. Pada lapisan yang lebih tinggi, satu-satunya alamat yang tersedia adalah alamat port. Namun, sekarang, kami menyediakan alamat jaringan aktual untuk sumber daya sumber dan tujuan. Kedua, pada lapisan ini, router beroperasi. Di Bab 1, kami menyebut router sebagai perangkat Lapisan 3 karena Lapisan 3 mengacu pada lapisan jaringan OSI, yang setara dengan lapisan ini. Router akan menerima paket yang masuk, melihat alamat IP tujuan, dan meneruskan paket itu ke kaki berikutnya dari rutenya. Karena router tidak tertarik pada konsep seperti aplikasi yang dimaksud atau pengurutan paket berdasarkan nomor urut, router hanya perlu memeriksa data yang berkaitan dengan lapisan ini. Dengan kata lain, sebuah router menerima sebuah paket dan mengambilnya dari link layer hingga ke Internet layer untuk membuat keputusan routing. Paket tidak melangkah lebih jauh ke tumpukan protokol saat router memeriksanya. Ini menghemat waktu. Terakhir, lapisan Internet menangani deteksi kesalahan data di header paket. Ini dilakukan melalui checksum header. Itu tidak berusaha menangani deteksi kesalahan dari data itu sendiri; tugas ini ditangani di lapisan transport.

Dalam Bab 1, kami secara singkat memeriksa IPv4 dan IPv6. Kami akan melakukannya lagi di dua subbagian pertama dan memberikan detail yang jauh lebih besar. Kami juga memperkenalkan dua protokol lainnya, Internet Control Message Protocol (ICMP) dan ICMPv6. Kami juga akan menjelaskan cara membuat sumber daya dengan alamat IP statis atau alamat dinamis (menggunakan DHCP). Kami hanya berfokus pada Unix/Linux saat membahas alamat IP statis versus dinamis. Kami mengakhiri bagian ini dengan memeriksa NAT, di mana alamat IP yang digunakan secara internal di jaringan mungkin tidak cocok dengan alamat eksternal, seperti yang digunakan di luar jaringan.

Alamat protokol internet versi 4 dan protokol internet versi 6

Alamat IP, apakah IPv4 atau IPv6, terdiri dari dua bagian: alamat jaringan dan alamat host. Alamat jaringan menentukan jaringan spesifik di Internet tempat penerima berada. Alamat host adalah alamat perangkat di dalam jaringan itu. Ini juga disebut sebagai nomor node. Alamat IPv4 awal dibagi menjadi satu dari lima kelas, yang menunjukkan apa yang kami sebut jaringan berkelas. Hari ini, kami juga memiliki gagasan jaringan nonclassful. Untuk jaringan yang berkelas, menentukan bagian mana dari alamat IPv4 yang merupakan alamat jaringan dan mana yang merupakan alamat host didasarkan pada kelas. Alamat IPv4 jaringan nonclassful memerlukan mekanisme tambahan untuk memperoleh alamat jaringan. Mekanisme ini disebut netmask jaringan. Ini tidak berlaku dengan jaringan IPv6.

Untuk jaringan classful, lima kelas dikenal sebagai kelas A, kelas B, kelas C, kelas D, dan kelas E. Kelas milik jaringan ditentukan oleh oktet pertama alamat (ingat bahwa alamat IPv4 terdiri dari angka 32-bit yang dipisahkan menjadi empat angka 8-bit; oktet adalah angka 8-bit yang biasanya ditampilkan sebagai bilangan bulat tunggal dari 0 hingga 255). Jumlah alamat IP spesifik yang tersedia untuk jaringan juga didasarkan pada kelasnya. Tabel 5.6 menggambarkan kelima kelas tersebut. Perhatikan bahwa sementara kelas E memiliki alamat yang disediakan untuknya, kelas E adalah kelas eksperimen dan tidak digunakan secara umum. Demikian pula, jaringan kelas D hanya digunakan untuk tujuan multicast.

Perhatikan dari tabel bahwa alamat yang dimulai dengan oktet 0 dan 127 tidak digunakan. Selain itu, perhatikan bahwa ada banyak alamat yang dicadangkan untuk kelas D dan E. Jadi, tidak semua 4,29 miliar alamat yang tersedia digunakan.

Tabel 5.6 Lima Kelas Jaringan

Kelas	Nomor Jaringan dalam Bit (Oktet)	Nomor Host dalam Bit (Oktet)	Jumlah Jaringan Kelas Ini	Jumlah Host dalam Jaringan Kelas Ini	Alamat Awal	Alamat Akhir
A	8 (1)	24 (3)	128	16.777.216	1.0.0.0	126.255.255.255
B	16 (2)	16 (2)	16.384	65.536	128.0.0.0	191.255.255.255
C	24 (3)	8 (1)	Lebih dari 2 juta	256	192.0.0.0	223.255.255.255
D	Tak terdefinisikan	Tak terdefinisikan	Tak terdefinisikan	Tak terdefinisikan	224.0.0.0	239.255.255.255
E	Tak terdefinisikan	Tak terdefinisikan	Tak terdefinisikan	Tak terdefinisikan	240.0.0.0	255.255.255.255

Diberi alamat IP, harus jelas kelas mana yang dimilikinya — cukup periksa oktet pertama. Jika oktet itu diberi nomor 1 sampai 126, itu milik jaringan kelas A, dan jika dimulai dengan 128 sampai 191, itu milik jaringan kelas B, dan seterusnya. Untuk memisahkan alamat jaringan dari alamat mesin, cukup bagi alamat menjadi dua bagian pada titik yang ditunjukkan dalam tabel. Misalnya, alamat jaringan 1.2.3.4 adalah jaringan kelas A, dan jaringan kelas A memiliki satu oktet untuk alamat jaringan dan tiga oktet untuk alamat host. Oleh karena itu, alamat jaringan 1.2.3.4 adalah 1, dan perangkat memiliki alamat host 2.3.4. Alamat 159.31.55.204 adalah jaringan kelas B, yang jaringannya dilambangkan sebagai 159.31, dan perangkat memiliki alamat 55.204.

Kami telah mengembangkan notasi untuk menyatakan alamat jaringan ini. Alamat jaringan umumnya dinyatakan dengan menggunakan awalan alamat IP. Meskipun disebut sebagai awalan, kami sebenarnya menambahkan informasi tentang jaringan di akhir alamat. Formatnya adalah alamat jaringan/n, di mana n adalah jumlah bit dari alamat IPv4 32-bit yang

menyusun jaringan. Seperti yang telah kami tunjukkan, alamat IPv4 1.2.3.4 memiliki satu oktet (8 bit) yang didedikasikan untuk jaringan, dan oleh karena itu, awalan alamat IP-nya diindikasikan sebagai 1.0.0.0/8, sedangkan alamat 159.31.55.204 memiliki dua oktet (16 bit) dan akan dilambangkan sebagai 159.31.0.0/16. Anda mungkin bertanya-tanya mengapa perlu memiliki awalan (bagian / n), karena jelas di mana alamat jaringan berhenti karena oktet nol. Kami akan kembali ke ini segera.

Jaringan classful pertama kali diusulkan pada tahun 1981 dan digunakan untuk pengalamatan Internet hingga tahun 1993. Namun, mereka memiliki kekurangan yang serius. Bayangkan beberapa organisasi besar (pemerintah atau perusahaan yang sangat besar) telah dianugerahi jaringan kelas A. Karena bagian jaringan dari alamat adalah 8 bit, ada 24 bit yang tersisa untuk alamat host. Ini memberi organisasi 224 (16.777.216) alamat kekalahan untuk digunakan secara internal ke jaringannya. Pada tahun 1993, tidak biasa bagi organisasi mana pun untuk membutuhkan lebih dari 16 juta alamat, dan oleh karena itu, beberapa, mungkin banyak, dari alamat ini tidak dapat digunakan. Ini semakin memperburuk masalah alamat IPv4 yang tidak tersedia.

Mulai tahun 1993, lima kelas dan jaringan classful digantikan oleh classless inter-domain routing (CIDR). Untuk mendukung CIDR, subnetting dikembangkan untuk membagi satu jaringan (berdasarkan alamat jaringan) menjadi banyak jaringan, masing-masing dengan alamat jaringannya sendiri dan kumpulan alamat hostnya sendiri. Jaringan CIDR memerlukan cara yang berbeda untuk menentukan alamat jaringan, dan oleh karena itu, kami beralih menggunakan netmask subnet.

Subnet jaringan adalah jaringan di mana semua perangkat berbagi alamat jaringan yang sama. Konsep ini berlaku untuk jaringan classful dan CIDR. Misalnya, semua perangkat di jaringan kelas A yang oktet pertamanya adalah 1 berbagi alamat jaringan yang sama, 1.0.0.0. Sekarang, bayangkan administrator jaringan dari jaringan kelas A ini memutuskan untuk membagi alamat yang tersedia sebagai berikut:

1.0.0.0
1.1.0.0
1.2.0.0
1.3.0.0
…
1.255.0.0

Artinya, mereka telah membagi alamat mereka menjadi 256 subnet yang berbeda. Setiap subnet tertentu, katakanlah 1.5.0.0/16, akan berisi perangkat yang semua alamat mesinnya memiliki alamat jaringan yang sama, 1.5.0.0. Organisasi ini kemudian dapat menjual sebagian besar alamat IP ini ke organisasi lain. Dalam setiap subnet, ada 16 bit yang tersedia untuk alamat host, atau 65.536 alamat berbeda. Katakanlah satu organisasi membeli alamat 1.5.0.0. Mungkin, pada gilirannya, membagi jaringannya menjadi subnet lebih lanjut dengan alamat berikut:

1.5.0.0
1.5.1.0
1.5.2.0
1.5.3.0
…
1.5.255.0

Sekarang, masing-masing dari 256 subnet ini memiliki 256 alamat individual. Dalam contoh ini, setiap subnet diatur sekitar oktet penuh. Namun, tidak harus seperti ini. Mari kita pertimbangkan bahwa sebuah organisasi memiliki jaringan kelas B, yang semua alamatnya dimulai dengan 129.205. Ini memutuskan bahwa daripada membaginya menjadi 256 subnet, itu akan membaginya menjadi empat subnet yang ditunjukkan oleh rentang alamat IP. Untuk ini, mari kita pertimbangkan kembali alamat ini dalam biner. 129.205 adalah 10000001.11001101. Keempat subnet kemudian dialamatkan sebagai berikut:

10000001.11001101.00 dengan rentang alamat host 000000.00000000 hingga 111111.11111111
10000001.11001101.01 dengan rentang alamat host 000000.00000000 hingga 111111.11111111
10000001.11001101.10 dengan rentang alamat host 000000.00000000 hingga 111111.11111111
10000001.11001101.11 dengan rentang alamat host 000000.00000000 hingga 111111.11111111

Kami telah menetapkan subnet untuk alamat IP 129.205.0.0–129.205.63.255, 129.205.64.0– 129.205.127.255, 129.205.128.0–129.205.191.255, dan 129.205.192.0–129.2 05.255.25. Pada strategi ini, kita tidak lagi dapat mengidentifikasi jumlah bit yang dibutuhkan untuk menentukan alamat jaringan hanya dengan melihat oktet pertama. Jadi, kami menambahkan informasi lain ke setiap pesan yang dikirimkan dengan pesan tersebut: netmask (atau subnet mask). Ingatlah bahwa awalan jaringan menyatakan n bit untuk menunjukkan berapa banyak bit yang menyusun jaringan. Kami secara implisit menyandikan ini ke dalam netmask dengan menghasilkan bilangan biner n 1s diikuti oleh 32−n 0s. Misalnya, jaringan kelas A akan memiliki n = 8, sehingga netmask akan menjadi 8 1s, diikuti oleh 32−8 atau 24 0s. Dalam contoh saat ini, alamat jaringan kami terdiri dari 18 bit pertama, menyisakan 32−18 atau 14 bit untuk alamat host.
Netmask kita adalah sebagai berikut: 11111111.11111111.11000000.00000000.

Perhatikan bahwa setiap netmask terdiri dari tiga oktet, yang nilainya 11111111 atau 00000000. Sebagai bilangan bulat, kedua oktet ini masing-masing adalah 255 dan 0. Hanya oktet yang tersisa yang mungkin terdiri dari nilai yang berbeda. Dalam contoh kita di atas, itu adalah oktet ketiga yang nilainya 11000000, yang sama dengan bilangan bulat 192. Oleh karena itu, netmask dapat ditulis ulang menjadi 255.255.192.0.

Diberi netmask, bagaimana kita menggunakannya untuk mengidentifikasi alamat jaringan untuk perangkat Internet? Kami menggunakan operator AND biner (atau Boolean) yang diterapkan ke alamat IPv4 perangkat dan netmask-nya. Operasi AND biner mengembalikan 1 jika kedua bit adalah 1 dan 0 sebaliknya. Misalnya, jika alamat IP kita adalah 129.205.216.44 dan netmask kita adalah 255.255.192.0, kita melakukan operasi berikut:

```
      10000001.11001101.11011000.00101100  (129.205.216.44)
AND   11111111.11111111.11000000.00000000  (255.255.192.0)
      10000001.11001101.11000000.00000000 = 129.205.192.0
```

Jadi, 129.205.216.44 memiliki alamat jaringan 129.205.192.0.

Kembali ke pengertian awalan alamat IP, kita dapat melihat apa itu awalan jaringan hanya dengan menghitung jumlah 1 berturut-turut di netmask. Dalam contoh di atas, netmask

kita adalah 255.255.192.0 atau 11111111.11111111.11000000.00000000, yang memiliki 18 detik berturut-turut. Oleh karena itu, awalan jaringan adalah 192.205.192.0/18. Alternatifnya, dengan diberikan awalan, kita dapat menurunkan netmask dengan menulis n 1s berturut-turut diikuti dengan 0s untuk menyelesaikan nilai 32-bit. Kami kemudian mengubah bilangan biner yang dihasilkan menjadi desimal sebagai empat oktet.

Mengingat alamat IP dan netmask perangkat, bisakah kita menentukan alamat host? Ya, kita bisa melakukannya dengan cukup mudah, mengikuti dua langkah ini. Pertama, kami menentukan topeng tuan rumah. Ini dapat diturunkan baik dengan menulis n 0s berturut-turut diikuti oleh 32−n 1s, atau dengan XORing netmask dengan semua 1s. Kami melihat kedua pendekatan. Karena n dalam contoh kita adalah 18, topeng host kita akan terdiri dari 18 0s diikuti oleh 32−18 atau 14 1s, atau 00000000.00000000.00111111.11111111. Atau, kami mengambil netmask kami dan XOR dengan semua 1s. Dengan XOR, hasilnya adalah 1 jika dua bit berbeda (satu adalah 0 dan satu adalah 1), jika tidak hasilnya adalah 0. Kedua, diberikan topeng host, DAN dengan alamat IP asli. Jadi, apa alamat host dari 129.205.216.44? Ambil netmask dan XOR dengan semua 1s.

```
      11111111.11111111.11000000.00000000
XOR   11111111.11111111.11111111.11111111
      00000000.00000000.00111111.11111111
```

Sekarang, DAN nilai ini dengan alamat IP.

```
      10000001.11001101.11011000.00101100  (129.205.216.44)
AND   00000000.00000000.00111111.11111111  (0.0.127.255)
      00000000.00000000.00011000.00101100 = 0.0.24.44
```

Perhatikan bahwa host's mask sebenarnya adalah kebalikan dari netmask.
Dalam contoh kita, netmask adalah 11111111.11111111.11000000.00000000, jadi mask host kita diperoleh dengan mengubah setiap bit:
1 menjadi 0 dan 0 menjadi 1, atau 00000000.00000000.00111111.11111111.

Melihat kembali contoh kita sebelumnya, oktet pertama dan kedua dari alamat jaringan sama dengan alamat IPv4, dan oktet keempat adalah 0. Untuk menentukan alamat host, kita juga dapat mengubah dua oktet pertama menjadi 0 dan yang keempat oktet ke oktet keempat dari alamat IPv4. Artinya, 129.205.216.44 akan memiliki alamat jaringan 129.205.x.0 dan 0.0.y.44. Apa itu x dan y? Menariknya, mengingat alamat IP dan x atau y, kita dapat menentukan yang lain. Oktet ketiga dari alamat IPv4 adalah 216. Oleh karena itu, x = 216 − y dan y = 216 − x. Dari netmask, kami menghitung bahwa oktet ketiga dari alamat jaringan kami adalah 192. Oleh karena itu, oktet ketiga dari alamat host kami adalah 216 − 192 = 24. Jadi, alamat host adalah 0.0.24.44.

Alamat signifikan lainnya dikenal sebagai alamat broadcast. Alamat ini digunakan untuk mengirim sinyal siaran ke semua perangkat di subnet yang diberikan. Untuk menghitung alamat broadcast, lengkapi netmask (balikkan semua bit) dan ATAU hasilnya dengan alamat jaringan. Operator OR adalah 1 jika salah satu bitnya adalah 1, dan 0 jika sebaliknya. Untuk

alamat kelas A, kelas B, dan kelas C, penerapan operator OR menghasilkan angka yang diakhiri dengan 255 untuk setiap oktet, dengan netmask 0. Misalnya, jaringan kelas A 100.0.0.0 akan memiliki perangkat siaran bernomor 100.255.255.255 , jaringan kelas B 150.100.0.0 akan memiliki perangkat siaran bernomor 150.100.255.255, dan jaringan kelas C yang beralamat 193.1.2.0 akan memiliki perangkat siaran bernomor 193.1.2.255. Namun, pertimbangkan alamat CIDR 10.201.97.13/20. Di sini, pertama-tama kita mendapatkan alamat jaringan dengan menggunakan netmask, yang terdiri dari 20 1s diikuti dengan 12 0s.

```
10.201.97.13 = 00001010.11001001.01100001.00001101
Netmask = 11111111.11111111.11110000.00000000
Network address = 00001010.11001001.01100000.00000000 = 10.201.96.0
```

Sekarang, kami ATAU alamat jaringan dengan komplemen dari netmask.

```
Network address = 00001010.11001001.01100000.00000000
Complement = 00000000.00000000.00001111.11111111
Broadcast address = 00001010.11001001.01101111.11111111 = 10.201.111.255
```

Perhatikan di sini bahwa dua oktet pertama sama dengan dua oktet pertama dari alamat IP perangkat, dan oktet keempat adalah 255. Hanya oktet ketiga yang perlu kita hitung.

Seperti disebutkan dalam Bab 1 dan diperkenalkan sebelumnya dalam bab ini, TCP/IP menggunakan alamat penting lainnya, alamat port. Ini adalah angka 16-bit yang menunjukkan protokol (atau aplikasi) yang akan menggunakan paket TCP/IP. Misalnya, port 80 dicadangkan untuk permintaan HTTP (untuk server web), port 443 digunakan untuk HTTPS (HTTP aman), dan port 53 digunakan untuk DNS, hanya untuk menyebutkan beberapa. Meskipun alamat port ditempatkan di bidang paket yang terpisah dari alamat IP, kita dapat menambahkan alamat IP dengan nomor port setelah ":" seperti pada 1.2.3.4:80 atau 129.205.216.44:53.

Seperti yang kami sebutkan sebelumnya di subbagian ini, ada banyak alamat IPv4 yang tidak digunakan. Apa alamat ini? Alamat 0.0.0.0 dapat digunakan untuk menunjukkan jaringan saat ini. Ini hanya berguna sebagai alamat sumber. Semua alamat yang dimulai dengan oktet 10 atau dua oktet 172.16 hingga 172.31 atau 192.168 dicadangkan untuk jaringan pribadi. Alamat ini dapat digunakan dari dalam jaringan pribadi, tetapi alamat semacam itu tidak dapat digunakan di Internet. Misalnya, alamat 10.11.12.13 dapat diketahui dari dalam jaringan, tetapi jika itu adalah alamat tujuan yang dikirim keluar dari jaringan pribadi, router Internet tidak akan dapat melakukan apa pun dengannya.

Oktet awal 127 digunakan untuk menunjukkan apa yang dikenal sebagai host lokal, atau perangkat loopback. Ini digunakan di komputer Unix/Linux oleh perangkat lunak yang tidak perlu menggunakan jaringan tetapi ingin menggunakan panggilan fungsi jaringan untuk melakukan tugasnya. Artinya, setiap pesan yang dikirim ke loopback tidak benar-benar masuk ke jaringan, tetapi perangkat lunak dapat melihat loopback seolah-olah itu adalah antarmuka jaringan. Alamat jaringan yang dimulai dengan nilai biner 1110 (yaitu, 224.0.0.0/4–239.0.0.0/4) adalah alamat kelas D dan hanya digunakan untuk pesan multicast. Demikian pula, alamat yang dimulai dengan 1111 (240.0.0.0/4–255.0.0.0/4) adalah alamat kelas E dan

dicadangkan untuk penggunaan di masa mendatang. Terakhir, alamat 255.255.255.255 dicadangkan sebagai alamat broadcast.

Jika Anda menambahkan ini, Anda menemukan bahwa alamat yang dimulai dengan 10 mencakup lebih dari 16 juta alamat yang dipesan, yang dimulai dengan 172,16 hingga 172,31 mencakup lebih dari 1 juta alamat, dan yang dimulai dengan 192,168 mencakup lebih dari 65.000 alamat. Yang dicadangkan untuk kelas D dan E masing-masing mencakup lebih dari 268 juta alamat. Ini berjumlah lebih dari 500 juta alamat yang dicadangkan dan karenanya bukan bagian dari kumpulan alokasi umum IPv4.

Selain reservasi ini, setiap subnet memiliki dua alamat yang dicadangkan: satu untuk menunjukkan jaringan dan yang lainnya untuk menunjukkan alamat broadcast (mengirim pesan ke alamat broadcast dari subnet menyebabkan pesan dikirim ke semua antarmuka di subnet sebagai jika perangkat siaran untuk subnet itu adalah hub). Alamat jaringan adalah apa yang kami hitung saat menerapkan netmask di atas. Misalnya, alamat IP 10.201.112.24 dengan netmask 255.255.224.0 (yaitu, 19 bit pertama membentuk alamat jaringan) akan menghasilkan alamat jaringan 10.201.96.0. Jika netmask jaringan ini adalah 255.255.0.0, jaringan akan memiliki alamat yang dicadangkan 10.201.0.0. Antara alamat yang tidak tersedia bagi kami dengan menggunakan skema penomoran IPv4 dan fakta bahwa setiap saat ada miliaran perangkat yang digunakan di Internet (ketika kami menambahkan perangkat seluler), kami telah mencapai kelelahan IP. Pada bulan Februari 2011, *The Internet Assigned Numbers Authority* (IANA) mengeluarkan lima set alamat IPv4 terakhirnya. Meskipun kami mungkin tidak benar-benar menggunakan semua alamat yang tersedia, tidak ada yang tersisa untuk dikeluarkan ke negara baru atau perusahaan telekomunikasi. Protokol Internet versi 4 dibuat pada saat Internet hanya memiliki beberapa ribu host. Saat itu, tidak ada yang mengharapkan miliaran perangkat Internet.

Protokol internet versi 4

Dengan pemahaman yang lebih jelas tentang alamat IPv4, mari kita pertimbangkan apa yang dilakukan lapisan Internet dengan sebuah paket. Ingatlah bahwa lapisan aplikasi TCP membuat pesan dan lapisan transport TCP membentuk satu atau lebih datagram TCP atau UDP dari pesan tersebut. Datagram terdiri dari header TCP atau UDP dan data itu sendiri. Header berisi setidaknya alamat port sumber dan tujuan, panjang, dan checksum (datagram TCP berisi bidang tambahan). Sekarang, lapisan Internet menambahkan tajuknya sendiri. Header ini harus menyertakan alamat IP sebenarnya dari tujuan pesan dan sumbernya. Informasi di header ini berbeda antara paket IPv4 dan IPv6. Jadi di sini, kita melihat paket IPv4.

Paket IPv4 akan berisi header lapisan Internet yang terdiri dari setidaknya 13 bidang dan bidang ke-14 opsional (berisi opsi, jika ada). Bidang-bidang ini panjangnya bervariasi dari beberapa bit hingga 32 bit. Kami mengidentifikasi masing-masing bidang tajuk pada Tabel 5.7 dan selanjutnya membahas beberapa bidang di bawah ini.

Ada beberapa item dalam tabel yang perlu kita diskusikan lebih detail. Pada lapisan transport, pesan dibagi menjadi paket-paket berdasarkan ukuran datagram yang diperbolehkan. TCP memiliki ukuran terbatas, tetapi UDP tidak. Aplikasi yang menggunakan UDP sebagai gantinya mengelompokkan data ke dalam paket-paket untuk memastikan bahwa tidak ada terlalu banyak data yang dikirim dalam satu paket jika paket tersebut terjatuh atau

rusak. Namun, ini tidak berarti bahwa jaringan itu sendiri dapat menangani ukuran paket, seperti yang ditentukan oleh lapisan transport. Ukuran yang dapat ditangani jaringan dalam satu paket apa pun dikenal sebagai unit transmisi maksimum (MTU). Ini berbeda berdasarkan teknologi jaringan. Misalnya, MTU Ethernet adalah 1500 byte, sedangkan LAN nirkabel memiliki MTU 7981 dan Token Ring memiliki MTU 4464. Karena lapisan transport tidak mempertimbangkan lapisan di bawahnya, paket yang dihasilkannya memiliki ukuran yang ditentukan. oleh aplikasi atau oleh TCP. Pada lapisan Internet, router akan mengetahui ke mana harus meneruskan paket yang diberikan dengan melihat tabel perutean. Tabel perutean juga akan menyertakan MTU untuk cabang jaringan tersebut. Jika ukuran total paket (termasuk header) lebih besar dari MTU, maka paket ini harus disegmentasi lebih lanjut. Ini dikenal sebagai fragmen.

Tabel 5.7 Bidang Header IPv4

Nama Bidang	Ukuran dalam Bit	Gunakan / Arti
Versi: kapan	4	Jenis paket (nilai = 4 untuk IPv4).
Ukuran tajuk	4	Ukuran header ini (tidak termasuk header layer transport) dalam peningkatan 32-bit. Misalnya, 5 akan menjadi header 32 * 5 = 160 bit, atau 20 byte.
Layanan yang berbeda	6	Menunjukkan jenis data, yang digunakan, misalnya, untuk menunjukkan konten streaming.
Pemberitahuan kemacetan eksplisit	2	Seperti disebutkan dalam subbagian lapisan transport, ada kemungkinan bahwa jaringan dapat mendeteksi kemacetan dan menggunakannya untuk mengontrol aliran. Jika opsi ini sedang digunakan, kolom ini menunjukkan apakah sebuah paket harus dibuang saat terjadi kemacetan.
Panjang total	16	Ukuran dalam byte dari keseluruhan paket, termasuk header IP, header TCP/UDP, dan data. Dengan 16 bit, ukuran maksimumnya adalah 65.535 byte. Ukuran minimum adalah 20 byte.
Identifikasi	16	Jika paket ini perlu disegmentasi lebih lanjut menjadi fragmen-fragmen, kolom ini digunakan untuk mengidentifikasi fragmen-fragmen dari paket asli yang sama. Ini dibahas dalam teks.
Bendera	3	Bit pertama saat ini tidak digunakan dan harus 0. Dua bit berikutnya digunakan untuk menunjukkan apakah paket ini diizinkan untuk dipecah (a 0 di bidang kedua) dan, jika dipecah, untuk menunjukkan apakah ini bukan bagian terakhir (a 1 di bidang ketiga).
Offset fragmen	13	Jika paket ini berupa fragmen, offset ini menunjukkan posisi fragmen di dalam paket. Ini dibahas dalam teks.

Waktu untuk hidup	8	Time to live digunakan untuk menentukan apakah sebuah paket yang telah berjalan selama beberapa waktu harus terus diteruskan atau dijatuhkan. Waktu untuk hidup dijelaskan lebih detail dalam teks.
Protokol	8	Jenis khusus dari protokol lapisan Internet yang digunakan untuk paket ini. Ini dapat, misalnya, termasuk IPv4, IPv6, ICMP, IGMP, TCP, Chaos, UDP, dan DDCP, di antara banyak lainnya.
Checksum tajuk	16	Checksum untuk header ini saja (ini tidak termasuk header atau data lapisan transport).
Alamat IP sumber dan tujuan	32 masing-masing	
Opsi (jika ada)	Variabel	Opsi jarang digunakan tetapi tersedia untuk memberikan ekstensibilitas ke IPv4. Kami menghilangkan diskusi tentang opsi.

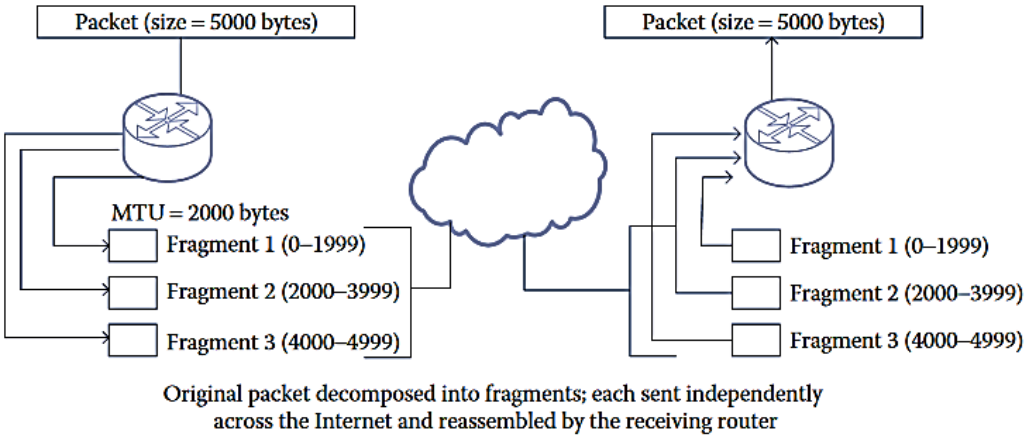
Seperti disebutkan dalam Tabel 5.7, header IPv4 memiliki tiga entri yang berkaitan dengan fragmen. Entri pertama adalah nomor identifikasi. Setiap paket yang dikirim dari router ini harus memiliki nomor identifikasi yang unik. Jumlah ini akan bertambah untuk setiap paket baru. Akhirnya, nomor identifikasi akan digunakan kembali tetapi tidak untuk beberapa waktu. Jika paket ini perlu disegmentasi menjadi fragmen, nomor ini akan digunakan oleh router tujuan untuk menyusun kembali fragmen menjadi sebuah paket. Gambar 5.9 mengilustrasikan ide ini.

Entri kedua terdiri dari tiga bendera. Bendera pertama tidak digunakan dan harus memiliki nilai 0. Bendera kedua, dilambangkan sebagai DF, digunakan untuk menunjukkan bahwa paket ini tidak boleh disegmentasi menjadi fragmen. Jika bidang ini disetel dan ada kebutuhan untuk membagi paket lebih lanjut, paket tersebut akan dibuang begitu saja. Bendera ketiga, dilambangkan sebagai MF, menunjukkan bahwa lebih banyak fragmen akan menyusul. Bit ini jelas (0) jika ini adalah fragmen terakhir dari paket tersegmentasi.

Entri ketiga tentang fragmen adalah offset fragmen. Ini adalah offset byte dari paket untuk menunjukkan urutan di mana fragmen harus dipasang kembali. Mari kita perhatikan contoh yang sangat sederhana (Gambar 5.9). Jaringan kami memiliki MTU 2000 byte. Paket yang dikirim adalah 5000 byte. Lapisan Internet harus membagi paket menjadi tiga fragmen yang offset fragmennya adalah 0, 2000, dan 4000 dalam urutan itu. Dua fragmen pertama akan memiliki medan MF yang ditetapkan dan fragmen ketiga akan memiliki medan MF yang dibersihkan.

Bidang waktu untuk hidup (TTL) digunakan sebagai sarana untuk memastikan bahwa tidak ada paket yang bertahan terlalu lama dalam upayanya melintasi Internet. Bayangkan sebuah paket telah dikirim ke Internet dan dialihkan dengan cara yang aneh, mengambil ribuan lompatan. Penerima, mungkin mengharapkan paket datagram TCP, meminta agar paket tersebut dikirim ulang. Paket dikirim ulang, dan versi terbaru ini tiba di tujuan jauh sebelum versi paket sebelumnya. Dengan menggunakan TTL, kita dapat membatasi waktu

hidup paket, sehingga, setelah berlalu, router akan membuang paket tersebut. TTL ditentukan dalam hitungan detik. Namun, ini menyebabkan komplikasi. Sebuah router dapat mengurangi beberapa jumlah dari TTL, tetapi bagaimana ia mengetahui jumlah waktu yang dibutuhkan paket untuk mencapainya sejak pertama kali dikirim? Sebagai gantinya, router cukup mengurangi TTL dengan 1, sehingga TTL benar-benar diperlakukan sebagai jumlah hop maksimum. Oleh karena itu, saat menerima paket, jika router menurunkan TTL menjadi 0, paket tersebut akan dibuang daripada diteruskan ke router berikutnya.



Gambar 5.9 Fragmenting dan reassembling paket.

Protokol internet versi 6

Protokol Internet versi 6 pertama kali dibahas pada tahun 1996, dengan standar yang menjelaskan IPv6 dirilis pada tahun 1998. Meski begitu, sistem operasi pertama yang mendukung IPv6 tidak tersedia hingga tahun 2000, ketika BSD Unix, Solaris, dan Windows 2000 membuatnya tersedia. Pada tahun 2005, Pemerintah A.S. mewajibkan agen federal mana pun yang mempertahankan bagian apa pun dari tulang punggung Internet agar sesuai dengan IPv6. Namun, di tahun-tahun berikutnya, Internet masih sangat bergantung pada IPv4. Mengapa demikian? Sebagian besar sistem operasi telah ditingkatkan untuk mengakomodasi IPv6, seperti halnya DNS. Namun, ada banyak perangkat Internet yang tidak mendukung IPv6, yaitu beberapa router lama yang masih mengisi jaringan. Tes bulanan diadakan oleh organisasi ipv6-test.com. Hasil dari Desember 2016 menunjukkan bahwa lebih dari separuh situs yang diuji memenuhi IPv6. Karena itu, banyak situs terus menggunakan IPv4 sebagai default. Statistik dari Google per Januari 2017 memperkirakan bahwa kurang dari 18% pengguna Google menggunakan IPv6. Banyak negara masih bekerja untuk membuat infrastruktur mereka berkemampuan IPv6. Hingga Internet lengkap siap menggunakan IPv6, Internet harus menggunakan keduanya.

Ini membawa kita ke masalah. Kedua versi, IPv4 dan IPv6, tidak kompatibel. Anda tidak dapat mengambil paket IPv4 dan membuatnya menjadi paket IPv6 untuk ditransmisikan melalui Internet. Sebaliknya, kedua protokol, meskipun serupa, tidak dapat dioperasikan. Jadi, sebuah paket adalah IPv4 atau IPv6. Router mungkin dapat menangani keduanya, tetapi jika router tidak dapat menangani salah satunya, itu adalah IPv6. Jadi, kita harus memastikan bahwa sistem operasi dan perangkat jaringan kita dapat menangani IPv4 minimal dan

sebaiknya keduanya. Perbedaan utama antara IPv4 dan IPv6 adalah ukuran alamatnya. Menjadi 128 bit, ruang alamat IPv6 jauh lebih besar (2128 vs 232). Untuk kenyamanan, kami menulis alamat 128-bit sebagai 32 digit heksadesimal. 32 digit heksadesimal ini dikelompokkan menjadi empat dan dipisahkan oleh titik dua (bukan empat oktet alamat IPv4 yang dipisahkan oleh titik). Misalnya, kita mungkin melihat alamat seperti 1234:5678:90ab:cdef:fedc:ba09:8765:4321.

Kita mungkin berharap banyak digit heksadesimal dari alamat IPv6 menjadi 0 (karena banyaknya alamat yang tidak terpakai di ruang yang sangat besar ini). Ada dua aturan singkatan yang diperbolehkan. Pertama, jika ada grup yang terdiri dari empat digit heksadesimal yang memiliki awalan 0, maka dapat dihilangkan. Misalnya, bagian 1234:0056 dapat ditulis sebagai 1234:56. Kedua, jika ada grup bagian yang berurutan dengan hanya 0, mereka dapat dihilangkan, hanya menyisakan dua titik dua yang berurutan. Anda diperbolehkan melakukan ini hanya sekali dalam nomor IPv6 mana pun, jadi urutan 0 diikuti oleh bukan 0 diikuti urutan 0 hanya akan mengizinkan satu bagian dari 0 untuk dihapus. Selain itu, satu kelompok yang terdiri dari empat 0 tidak dapat dihilangkan. Kasus-kasus khusus ini cenderung tidak muncul. Sebagai gantinya, kami berharap menemukan sebagian besar 0 terjadi setelah kelompok pertama digit non-0 dan sebelum digit non-0 lainnya. Berikut beberapa alamat IPv6 dan bentuk singkatannya. Perhatikan bahwa Anda dapat menghapus awalan 0 dari grup yang terdiri dari empat digit heksadesimal di titik mana pun di alamat. tetapi, seperti yang ditunjukkan pada contoh terakhir, Anda dapat mengeliminasi grup empat 0 hanya sekali.

2001:052a:0000:0000:0000:0001:0153:f1cd = 2001:52a::1:153:f1cd
fe80:0000:0000:0000:0123:b12f:ceaa:830c = fe80::123:b12f:ceaa:830c
1966:012e:0055:000f:0000:0000:0000:0c88 = 1966:12e:55:f::c88
2001:0513:0000:0000:5003:0153:0000:07b5 = 2001:513::5003:153:0000:07b5

Alamat IPv6, seperti alamat IPv4, dibagi menjadi dua bagian: alamat jaringan dan host. Namun, pada IPv6, pembagian ini dilakukan di tengah-tengah 128 bit. Artinya, alamat jaringan selalu 64 bit pertama dan alamat perangkat selalu 64 bit terakhir. Oleh karena itu, di IPv6, tidak perlu memiliki netmask karena setiap alamat jaringan akan menjadi 64 bit pertama.

Selain ukuran alamat dan cara menentukan jaringan versus alamat host, ada banyak perbedaan signifikan lainnya. Header IPv6 disederhanakan dari apa yang dikembangkan untuk IPv4. Misalnya, header IPv4 mengizinkan opsi yang membuat header memiliki panjang variabel. Sebaliknya, header IPv6 memiliki panjang 40 byte yang seragam. Jika Anda ingin menyertakan opsi, Anda menambahkannya ke ekstensi tajuk. Header IPv6 ini terdiri dari nomor versi 4-bit (sama seperti header IPv4), kelas lalu lintas 8-bit, label aliran 20-bit, panjang muatan 16-bit, header berikutnya 8-bit, 8- batas bit hop, dan dua bidang alamat 128-bit (16 byte) (sumber dan tujuan).

Kelas lalu lintas menggabungkan nomor layanan 8-bit yang dibedakan dan bidang 2-bit dari bendera pemberitahuan kemacetan eksplisit (ini sama dengan header IPv4, kecuali bahwa layanan yang dibedakan header IPv4 adalah 6 bit). Label aliran digunakan untuk streaming paket data untuk menunjukkan ke router bahwa semua paket dengan label aliran yang sama, jika memungkinkan, harus diarahkan di sepanjang jalur yang sama agar tetap bersama. Dengan cara ini, paket-paket harus tiba secara berurutan dan bukan rusak, karena

router akan meneruskan paket-paket tersebut dalam urutan yang sama saat menerimanya. Panjang muatan adalah ukuran paket lengkap, termasuk header ekstensi apa pun. Bidang header berikutnya menunjukkan bahwa bagian berikutnya dari paket adalah header ekstensi atau menyediakan jenis protokol yang digunakan untuk bagian paket lapisan transport (misalnya, apakah itu datagram TCP atau UDP). Batas hop menggantikan TTL dan digunakan secara ketat untuk menghitung jumlah hop router yang tersisa sebelum paket harus dijatuhkan.

Perhatikan bahwa satu bidang yang tidak berisi paket ini adalah checksum. Perubahan lain dalam IPv6 adalah mengandalkan lapisan transport dan lapisan tautan untuk menangani paket yang salah. Karena mekanisme di kedua lapisan lebih dari cukup untuk mendeteksi dan menangani kesalahan (kerusakan data), memiliki checksum lapisan Internet telah dijatuhkan di IPv6.

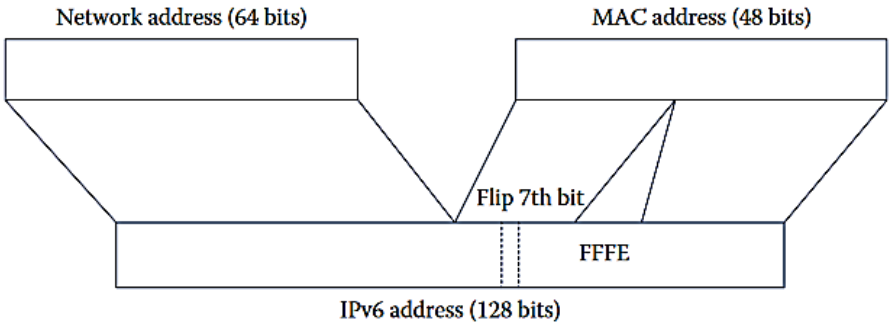
Opsi masih tersedia dengan menggunakan header ekstensi opsional. Seperti opsi IPv4, ini dipandang sebagai pengecualian lebih dari biasa. Di antara opsi yang tersedia adalah untuk menentukan opsi tujuan (opsi hanya diperiksa oleh router di tujuan), sarana untuk menentukan rute yang disukai di seluruh jaringan, memesan fragmen, verifikasi data untuk tujuan otentikasi, dan informasi enkripsi jika data dienkripsi.

Dengan sifat header yang disederhanakan, tugas router juga disederhanakan. Ini berasal dari empat perubahan. Pertama, header tidak lagi memiliki bidang opsional, sehingga sebagian besar pemrosesan dapat dilakukan dengan menganalisis 40 byte header. Kedua, seperti yang disebutkan, tidak ada checksum untuk ditangani, dan oleh karena itu, router tidak harus menghitung checksum atau menguji checksum untuk kesalahan. Ketiga, router tidak perlu melakukan fragmentasi, karena sekarang akan ditangani secara ketat di ujung sumber sebelum mengirim paket ke jaringan. Dengan demikian, sistem operasi untuk perangkat sumber sekarang memiliki beban tambahan: ia harus menemukan MTU untuk setiap potensi lompatan komunikasi dan membagi paket menjadi ukuran unit yang sesuai. Keempat, penerapan netmask tidak lagi diperlukan karena sifat alamat IPv6.

Perbedaan lain antara IPv4 dan IPv6 adalah bagaimana alamat ditetapkan dalam dua versi. Alamat IPv4 diberikan baik secara statis oleh administrator jaringan (manusia) atau secara dinamis oleh beberapa bentuk server (biasanya server DHCP, baik yang berjalan di router atau komputer). IPv6 menggunakan apa yang disebut stateless address autoconfiguration (SLAAC). Ini memungkinkan perangkat untuk menghasilkan alamat IPv6 sendiri dan kemudian menguji untuk melihat apakah alamat tersebut tersedia. Hanya jika tidak tersedia, perangkat harus meminta alamat dari server, seperti server DHCPv6.

Untuk menghasilkan alamat, bagian jaringan (64 bit pertama) akan sama dengan perangkat apa pun di jaringan. Perangkat harus menemukan bagian alamat ini, yang dapat dilakukan dengan mengalamatkan perangkat siaran subnet. Respons apa pun akan menyertakan alamat jaringan 64-bit. Alamat host (64 bit terakhir) dihasilkan dengan menyempurnakan alamat MAC 48-bit perangkat ke dalam format EUI-64. Anda mungkin ingat bahwa ini ditangani dengan memasukkan FFFE nilai heksadesimal ke tengah 48 bit. Misalnya, jika alamat MAC 48-bit adalah 00-01-02-03-04-05, maka menjadi 00-01-02- FF-EE-03-04-05. Sekarang, jika alamat ini dianggap sebagai alamat universal (yaitu, unik di Internet), bit ke-7

dari EUI-64 dibalik. Di alamat di atas, dua digit heksadesimal pertama dari 00 sebenarnya adalah 00000000 dalam biner. Membalik bit ke-7 memberi kita 00000010, atau 02. Jadi, alamat EUI-64 yang direvisi adalah 02-01-02-FF-EE-04-05. Ini menjadi 64 bit terakhir dari alamat IPv6.



Gambar 6.10 Membentuk alamat IPv6 dari alamat MAC.

Gambar 6.10 menunjukkan bagaimana kita dapat membuat alamat IPv6 secara otomatis. Setelah alamat terbentuk, perangkat mengirimkan pesan, termasuk alamat IPv6, meminta tanggapan duplikat. Jika tidak ada yang diterima, perangkat dapat berasumsi bahwa ia memiliki alamat yang unik. Jika tidak, mungkin akan menghubungi host DHCPv6 lokal.

Sebelumnya, kami mengatakan bahwa router IPv6 tidak akan memecah paket. Namun, perangkat asli yang mengirimkan paket mungkin harus memecah-mecah paket. Bagian label aliran dari header IPv6 menunjukkan jika ini adalah bagian dari sebuah fragmen di mana semua fragmen akan berbagi label aliran yang sama. Porsi payload (data) dari paket terfragmentasi akan berisi dua bagian. Yang pertama adalah header IPv6 yang akan sama untuk semua fragmen, diikuti dengan header ekstensi.

Variasi lain untuk paket IPv6 adalah jumbogram. Menentukan, melalui header ekstensi, bahwa paket tersebut adalah jumbogram memungkinkan paket menjadi jauh lebih besar. Tanpa ini, bagian data IPv6 berukuran terbesar adalah 65.535 byte. Namun, jumbogram memungkinkan bagian data sebesar 4 GB.

Karena IPv4 dan IPv6 tidak kompatibel satu sama lain, tanggung jawab untuk mendukung interoperabilitas harus terletak pada lapisan Internet. Lapisan ini harus mampu menangani paket IPv4 dan IPv6. Sayangnya, hal ini tidak selalu terjadi karena ada banyak server dan bahkan beberapa sistem operasi lama yang hanya menangani IPv4. Komunikasi IPv6 titik-ke-titik dimungkinkan tetapi hanya jika dua titik akhir dan semua lompatan di antaranya dapat menangani IPv6. Untuk mengatasi masalah ini, ada beberapa kemungkinan pilihan. Pendekatan pertama dan paling umum saat ini adalah implementasi dual IP stack. Dalam kasus seperti itu, IPv4 dan IPv6 tersedia. Sebagian besar sistem operasi (terutama sistem operasi yang lebih baru) dapat menangani kedua versi tersebut, seperti kebanyakan router yang lebih baru. Ketika sumber daya menghubungi yang lain melalui Internet untuk membuat koneksi pada lapisan transport, jika kedua titik akhir dan semua titik di antaranya dapat menangani IPv6, maka pengirim akan menggunakan ini. Jika tidak, pengirim mungkin kembali menggunakan IPv4. Pilihan lain yang mungkin, yang kurang efisien, adalah

menggunakan IPv4 untuk membuka koneksi dan, dari dalam terowongan IPv4 yang sudah ada, gunakan IPv6.

Membangun alamat protokol internet: secara statis dan dinamis

Kami menyebutkan bahwa alamat IPv4 dapat dibuat baik secara statis maupun dinamis. Alamat IP statis diberikan oleh manusia (administrator jaringan atau sistem) dari kumpulan alamat IP yang tersedia untuk organisasi. Alamat IP yang diberikan harus sesuai dengan skema pengalamatan organisasi (yaitu, alamat harus diizinkan, mengingat subnet perangkat). Sebagian besar alamat IP di masa-masa awal Internet adalah alamat statis. Saat ini, sebagian besar server menerima alamat IP statis sehingga mereka dapat dihubungi secara terprediksi, tetapi sebagian besar nonserver menerima alamat dinamis. Alamat IP dinamis hanya diberikan untuk sementara. Ini dikenal sebagai sewa. Jika sewa alamat IP Anda kedaluwarsa, perangkat Anda harus mendapatkan alamat IP baru.

Dalam subbagian ini, kami memeriksa cara menyetel komputer Anda agar memiliki alamat IP statis dan dinamis. Kami melihat dua dialek Linux dan Windows. Mari kita mulai dengan melihat Red Hat Linux. Kami akan berasumsi bahwa antarmuka Anda adalah kartu Ethernet. Dengan antarmuka seperti itu, Anda akan memiliki dua file konfigurasi penting dalam direktori /etc/sysconfig/network-scripts bernama ifcfg-lo dan ifcfg-eth0 (jika untuk antarmuka, cfg untuk config, lo adalah perangkat loopback, dan eth adalah antarmuka Ethernet Anda, biasanya bernama eth0). Anda tidak perlu mengubah ifcfg-lo karena antarmuka loopback Anda selalu memiliki alamat IP yang sama, 127.0.0.1. Untuk alamat IP statis, Anda harus memodifikasi

ifcfg-eth0 secara manual. File ini akan terdiri dari sejumlah arahan dalam bentuk VARIABLE=VALUE, di mana VARIABLE adalah variabel lingkungan dan VALUE adalah string yang Anda tetapkan ke variabel tersebut. Untuk menetapkan alamat IP statis, Anda harus menetapkan nilai ke satu set variabel. Jika Anda ingin mendapatkan alamat dinamis, Anda akan menetapkan variabel lain.

Untuk alamat IP statis, gunakan IPADDR=alamat dan BOOTPROTO="statis", di mana alamat adalah alamat yang Anda tetapkan untuk antarmuka ini. Penugasan DEVICE=eth0 seharusnya sudah ada. Selain itu, tetapkan HOSTNAME ke nilai nama host mesin (aliasnya). Ini tidak termasuk domain. Misalnya, jika domain Anda adalah somecompany.com dan mesin Anda adalah computer1, maka Anda akan menggunakan HOSTNAME="computer1", bukan HOSTNAME="computer1.somecompany.com". Anda juga perlu menyediakan netmask untuk subnet mesin ini dengan menggunakan variabel NETMASK. Terakhir, tetapkan ONBOOT="yes" untuk menunjukkan bahwa alamat IP tersedia saat boot. Anda juga dapat menentukan alamat NETWORK; namun, jika dihilangkan, sistem operasi Anda akan mendapatkan ini dari entri IPADDR dan NETMASK.

Anda juga harus memodifikasi file /etc/sysconfig/network untuk menetapkan variabel HOSTNAME dan GATEWAY. Gateway adalah koneksi jaringan lokal Anda ke jaringan lain. Nilai yang diberikan untuk GATEWAY akan menjadi alamat IP perangkat penghubung ini. Setelah file-file ini disimpan, perangkat Anda akan memiliki alamat IP dan aliasnya sendiri. Perangkat Anda juga perlu mengetahui server namanya. Ini harus ditentukan secara otomatis oleh gateway komputer Anda. Saat menemukan server nama ini, file /etc/resolv.conf diisi. Jika

Anda mengoperasikan komputer di rumah, server nama kemungkinan besar dihosting oleh ISP Anda.

Jika Anda ingin mengubah alamat IP statis, Anda akan memodifikasi file ifcfg-eth0 untuk memperbarui entri IPADDR dan mungkin NETMASK. Anda juga harus mengubah entri server DNS otoritatif apa pun untuk menunjukkan bahwa HOSTNAME memiliki catatan A baru. Jika Anda tidak memodifikasi server DNS, Anda mungkin menemukan bahwa respons terhadap pesan keluar Anda tidak berhasil kembali ke komputer Anda. Kami akan mengeksplorasi DNS di Bab 5 dan 6, jadi, kami menunda diskusi lebih lanjut untuk saat ini. Terakhir, setiap kali Anda memodifikasi file ifcfg-eth0, Anda perlu memulai ulang layanan jaringan. Tanpa ini, komputer Anda terus menggunakan alamat IP lama. Layanan jaringan dapat dimulai ulang dari baris perintah dengan menggunakan perintah layanan jaringan restart di Red Hat 6 dan sebelumnya atau systemctl restart jaringan di Red Hat 7.

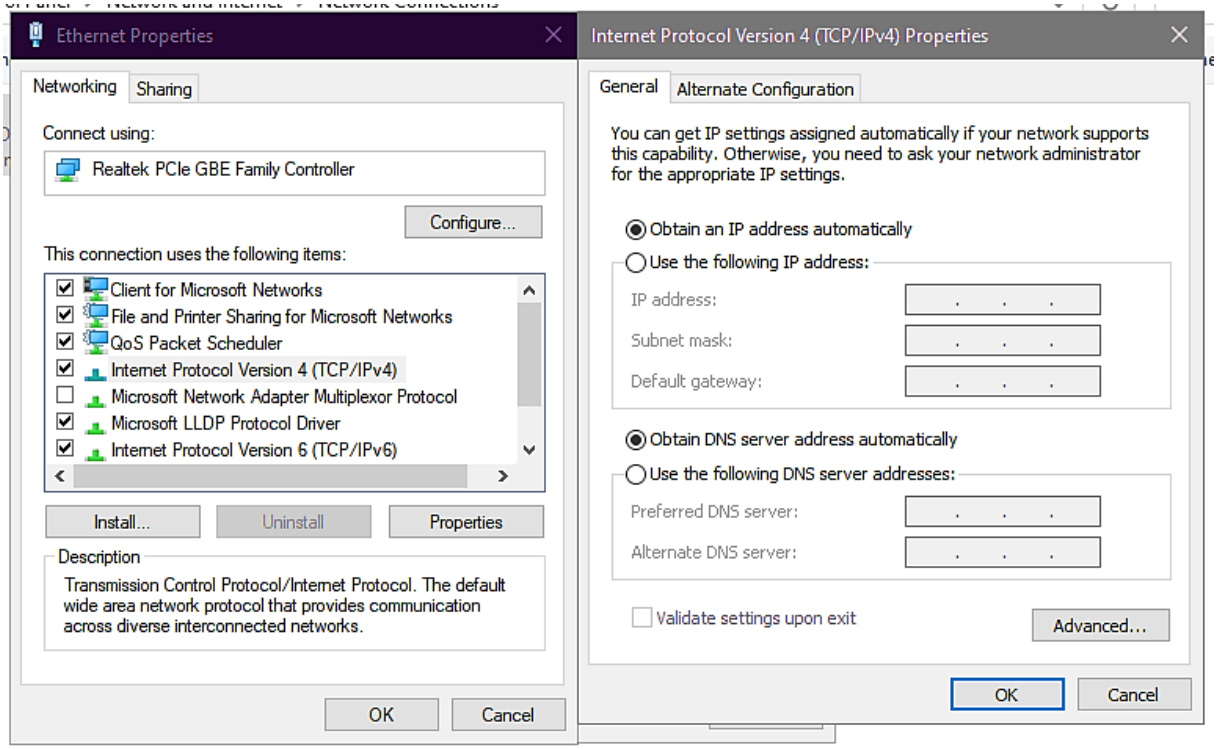
Jika Anda ingin menetapkan alamat IP statis untuk mesin Ubuntu, prosesnya sedikit berbeda. Daripada mengedit file ifcfg, Anda akan memodifikasi /etc/network/interfaces dengan menentukan alamat IP, alamat jaringan, netmask, alamat broadcast, dan alamat gateway. Berikut ini adalah contohnya. Anda akan melihat variabel yang sama tercantum di sini, seperti yang kita lihat di Red Hat, kecuali bahwa GATEWAY (dan dalam hal ini BROADCAST) disertakan dalam file tunggal.

```
iface eth0 inet static
       address 10.11.12.13
       network 10.11.0.0
       netmask 255.255.128.0
       broadcast 10.11.51.1
       gateway 172.83.11.253
```

Windows 7, 8, dan 10 menggunakan konfigurasi yang sama untuk alamat IP statis. Pertama, buka Jaringan dan Pusat Berbagi Anda. Anda dapat memperoleh ini melalui GUI Panel Kontrol atau dengan mengetik Jaringan dan Berbagi di kotak Pencarian menu tombol Mulai. Dari jendela Network and Sharing Center, pilih Change adapter settings. Jendela ini akan memberikan daftar semua antarmuka Anda. Anda mungkin akan memiliki satu perangkat berlabel Local Area Connection atau Wireless Network Connection. Ini akan menjadi NIC komputer Anda. Klik kanan pada koneksi yang sesuai, dan pilih Properties. Ini memunculkan jendela pop-up yang mirip dengan yang ditunjukkan di sisi kiri Gambar 3.11. Pilih Internet Protocol Version 4 (TCP/IPv4) dan tombol Properties di bawah daftar opsi dan jendela properti IPv4 muncul. Ini ditunjukkan di sisi kanan Gambar 3.11. Pilih Gunakan alamat IP berikut, dan isi kotak dengan menggunakan banyak informasi yang sama yang harus Anda tentukan di Linux: alamat IP statis, subnet mask, gateway, dan setidaknya satu alamat IP server DNS. Perhatikan, tidak seperti Linux, tidak ada ruang untuk memasukkan HOSTNAME atau apakah alamat IP tersedia saat boot.

DHCP dikembangkan untuk menggantikan Protokol Bootstrap yang sudah ketinggalan zaman pada awal 1990-an. DHCP memiliki fungsionalitas yang tidak dimiliki Bootstrap dan jauh lebih umum. Jika perangkat Anda menggunakan DHCP, perangkat tidak hanya akan memperoleh alamat IP secara dinamis, tetapi juga akan memperoleh informasi konfigurasi lain seperti alamat router/gateway jaringan lokal dan server DNS jaringan.

Klien, ketika membutuhkan alamat IP (biasanya saat boot), mengirimkan permintaan pada subnet lokalnya. Jika tidak ada server DHCP di subnet, router lokal meneruskan permintaan selanjutnya. Akhirnya, server DHCP akan merespons dengan tawaran. Klien akan menerima penawaran (atau penawaran jika ada beberapa penawaran) dengan permintaan. Pengakuan dari server akan mencakup tidak hanya alamat IP tetapi juga informasi jaringan lainnya, termasuk subnet mask, nama domain (alias), dan alamat perangkat siaran lokal. Di sini, kita melihat secara singkat bagaimana memastikan bahwa komputer Anda akan menggunakan DHCP di Red Hat Linux, Ubuntu, dan Windows.



Gambar 5.11 Mengkonfigurasi Windows dengan alamat IP statis.

Untuk Red Hat Linux, kami membuat beberapa modifikasi kecil pada file yang sama yang kami gunakan untuk menetapkan alamat statis: /etc/sysconfig/ifcfg-eth0. Di sini, BOOTPROTO diberi nilai dhcp, dan variabel IPADDR dan NETMASK dihilangkan. Demikian pula, jika Anda memiliki entri untuk NETWORK, itu juga akan dihapus. Untuk Ubuntu, ubah baris pertama menjadi iface eth0 inet dhcp, dan hapus baris yang tersisa dari file /etc/network/interfaces. Untuk Windows, gunakan pendekatan yang sama seperti yang disebutkan di atas, kecuali Anda akan memilih Dapatkan alamat IP secara otomatis (lihat sisi kanan Gambar 5.11).

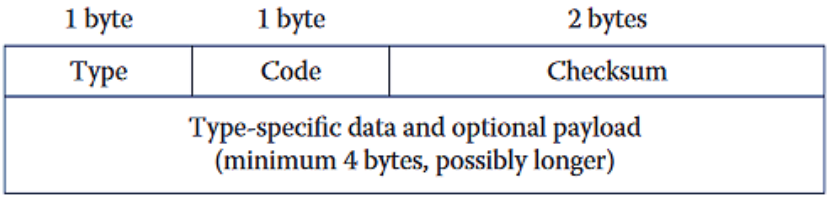
Untuk menggunakan DHCP, komputer Anda harus menjalankan layanan yang sesuai. Di Linux, ini adalah dhclient (terletak di /sbin). Layanan ini berjalan secara otomatis setiap kali Anda me-restart layanan jaringan Anda jika Anda telah menentukan bahwa antarmuka akan menggunakan dhcp (misalnya, dengan menentukan BOOTPROTO=dhcp). Jika Anda juga menentukan ONBOOT=yes, maka layanan ini berjalan selama proses inisialisasi sistem setelah booting. Variabel catatan lain untuk file ifcfg Anda adalah PERSISTENT_DHCLIENT. Dengan

menyetel variabel ini ke 1, Anda memberi tahu sistem operasi Anda untuk meminta router lokal atau DHCP hingga alamat IP diberikan. Tanpa ini, jika permintaan pertama Anda tidak mencapai DHCP atau, jika karena alasan tertentu, Anda tidak diberi alamat IP, maka Anda harus mencoba lagi. Anda akan me-reboot komputer Anda untuk mendapatkan alamat dinamis atau memulai ulang layanan jaringan Anda. Di Windows, layanan ini disebut Dhcp, dan secara default, harus diatur agar berjalan saat boot.

Kami akan menjelajahi server DHCP di Bab 6. Secara khusus, kami akan melihat cara mengkonfigurasi server DHCP di Linux. Server berpotensi berjalan di komputer Linux; namun, lebih umum untuk menjalankan server DHCP Anda pada perangkat router atau gateway.

Protokol pesan pengendalian internet dan protokol manajemen kelompok internet

Meskipun lapisan Internet menggunakan IPv4 dan IPv6 untuk mengirim pesan, ada dua protokol lain yang dapat digunakan lapisan ini: Internet Control Message Protocol (ICMP) dan Internet Group Management Protocol (IGMP). ICMP terutama digunakan oleh perangkat siaran untuk mengirim permintaan dan pesan kesalahan terkait aksesibilitas perangkat. Permintaan digunakan untuk menentukan apakah perangkat merespons, dan pesan kesalahan digunakan untuk menunjukkan bahwa perangkat saat ini tidak dapat diakses. ICMP tidak bertukar data. Dengan demikian, tidak ada porsi muatan untuk paket ICMP. Format paket ICMP ditunjukkan pada Gambar 5.12. Anda dapat melihat bahwa ini jauh lebih sederhana daripada header paket TCP, UDP, IPv4, dan IPv6.



Gambar 5.12 format paket ICMP.

Dalam header ICMP, tipe 8-bit menunjukkan tipe pesan kontrol. Beberapa nomor tipe dicadangkan tetapi tidak digunakan atau eksperimental, dan yang lainnya telah ditinggalkan. Di antara jenis yang perlu diperhatikan adalah gema (cukup balas), tujuan tidak dapat dijangkau (peringatan yang memberi tahu perangkat bahwa tujuan yang ditentukan tidak dapat diakses), pesan pengalihan (untuk menyebabkan pesan mengambil jalur lain), permintaan gema (cukup tanggapi), router iklan (umumkan router), permintaan router (temukan router), waktu terlampaui (TTL kedaluwarsa), header IP buruk, stempel waktu, atau respons stempel waktu. Kode 8-bit mengikuti jenis di mana kode menunjukkan penggunaan pesan yang lebih spesifik. Misalnya, tujuan tidak dapat dijangkau berisi 16 kode berbeda untuk menunjukkan alasan mengapa tujuan tidak dapat dijangkau seperti jaringan tujuan tidak diketahui, host tujuan tidak diketahui, jaringan tujuan tidak dapat dijangkau, dan host tujuan tidak dapat dijangkau. Ini diikuti oleh checksum 16-bit dan bidang 32-bit yang isinya bervariasi berdasarkan jenis dan kode. Sisa dari paket ICMP akan menyertakan header IPv4 atau IPv6, yang tentu saja akan menyertakan alamat IP sumber dan tujuan, dan kemudian sejumlah byte dari datagram asli (TCP atau UDP), termasuk alamat port dan informasi checksum.

Ada juga ICMPv6, yaitu untuk ICMP apa itu IPv6 untuk IPv4. Namun, ICMPv6 memainkan peran yang jauh lebih penting karena sebagian besar IPv6 adalah penemuan komponen jaringan. Perangkat IPv6 harus memanfaatkan ICMPv6 sebagai bagian dari proses penemuan tersebut. Format paket ICMPv6 sama dengan ICMP kecuali ada lebih banyak jenis pesan yang diimplementasikan dan semua jenis yang tidak digunakan lagi telah dihapus atau diganti. Secara khusus, ICMPv6 telah menambahkan tipe untuk mengizinkan laporan masalah parameter, resolusi alamat untuk IPv6 (kita akan membahas ini di Bagian 3.5 di bawah Address Resolution Protocol [ARP]), ICMP untuk IPv6, penemuan router, dan pengalihan router. Laporan masalah parameter termasuk mendeteksi alamat duplikat dan perangkat yang tidak dapat dijangkau (*Neighbor Unreachability Decision* [NUD]).

Seperti disebutkan di atas, ICMP terutama digunakan oleh perangkat siaran saat menjelajahi jaringan untuk melihat apakah node lain dapat dijangkau. Pengguna akhir jarang memperhatikan paket ICMP, kecuali saat menggunakan ping atau traceroute. Kedua program ini tersedia bagi pengguna akhir untuk secara eksplisit menguji ketersediaan perangkat. Kami akan mengeksplorasi kedua program ini di Bab 4.

IGMP memiliki kegunaan yang sangat spesifik: untuk membuat grup perangkat untuk keperluan multicast. Namun, IGMP sendiri tidak digunakan untuk multicast tetapi hanya untuk membuat atau menambah grup, menghapus dari grup, atau menanyakan keanggotaan grup. Implementasi sebenarnya untuk multicasting terjadi dalam protokol yang berbeda, seperti Simple Multicast Protocol, Multicast Transport Protocol, dan Protocol-Independent Multicast (protokol terakhir ini adalah keluarga dengan empat varian, berurusan dengan grup berpenduduk jarang, grup berpenduduk padat, grup multicast dua arah, dan multicast sumber-spesifik). Atau, multicasting dapat ditangani pada lapisan aplikasi. Jaringan IPv6 memiliki kemampuan built-in untuk melakukan multicasting. Oleh karena itu, IGMP hanya ditargetkan pada IPv4. Tidak ada yang setara dengan IGMPv6 untuk IGMP, tidak seperti IPv6 dan ICMPv6.

IGMP telah ada dalam tiga versi berbeda (dilambangkan sebagai IGMPv1, IGMPv2, dan IGMPv3). IGMPv2 mendefinisikan struktur paket. IGMPv2 dan IGMPv3 menentukan kueri dan laporan keanggotaan. Semua paket IGMP beroperasi pada lapisan Internet dan tidak melibatkan lapisan transport. Paket IGMP disusun sebagai berikut. Byte pertama adalah jenis pesan: untuk bergabung dengan grup tertentu, kueri keanggotaan, laporan keanggotaan IGMPv1, laporan keanggotaan IGMPv2, laporan keanggotaan IGMPv3, atau permintaan untuk keluar dari grup yang ditentukan. Byte berikutnya adalah waktu respons, yang hanya digunakan untuk kueri keanggotaan. Nilai ini menunjukkan batas waktu sebelum kueri habis. Berikutnya adalah checksum 16-bit untuk paket tersebut. Alamat grup 4-byte mengikuti, yang merupakan alamat unik yang didedikasikan untuk grup multicast. Jika pertanyaannya adalah untuk bergabung atau keluar dari grup, hanya ini yang diperlukan. Kueri dan laporan berisi informasi tambahan. Kueri berisi tanda tambahan untuk menentukan apa yang sedang ditanyakan. Laporan mencakup daftar semua anggota grup saat ini.

Ingat dari diskusi kita tentang kelas jaringan bahwa jaringan kelas D dicadangkan untuk alamat multicast. Alamat multicast akan berkisar dari 224/8 hingga 239/8. Dua alamat spesifik dalam rentang ini dicadangkan untuk grup semua host (224.0.0.1) dan grup semua router

(224.0.0.2). Jika perangkat Anda mampu melakukan multicasting, maka ping 224.0.0.1 akan membuat perangkat Anda merespons. Artinya, setiap dan setiap perangkat multicast harus menanggapi upaya ping ke alamat ini. Jika sebuah router mampu menangani multicast, ia harus bergabung dengan grup 224.0.0.2.

Penerjemahan alamat jaringan

Peran lain dari lapisan Internet adalah melakukan NAT. Dengan menggunakan NAT, alamat yang digunakan dalam jaringan area lokal tidak terlihat oleh dunia luar. Sebaliknya, saat paket berpindah dari LAN ke Internet, alamat internal diterjemahkan ke alamat yang digunakan di dunia luar. Saat paket masuk dari Internet, alamat diterjemahkan ke alamat internal. Biasanya, alamat internal menggunakan salah satu ruang alamat pribadi yang disisihkan oleh IANA. Misalnya, setiap alamat IPv4 yang dimulai dengan 10 adalah alamat pribadi (yang lain termasuk yang berada dalam kisaran 172.16–172.31 dan yang dimulai dengan 192.168).

Ada dua bentuk NAT: satu-ke-banyak dan satu-ke-satu. Formulir ini menunjukkan apakah ada satu alamat eksternal untuk semua perangkat internal atau ada satu alamat eksternal untuk setiap alamat internal. NAT satu-ke-satu memberikan anonimitas untuk perangkat di LAN. Anonimitas bukan hanya masalah kenyamanan bagi pengguna jaringan yang mungkin tidak ingin diidentifikasi (dan, faktanya, karena alamat IP internal mereka diketahui oleh perangkat apa pun yang melakukan penerjemahan, pengguna mungkin tidak seanonim mereka. berpikir), tetapi juga menyediakan lapisan keamanan untuk jaringan. Jika alamat IP internal tidak dapat diketahui dari luar LAN, itu membuat penyerang jauh lebih sulit untuk dapat menyerang perangkat tertentu itu. NAT satu-ke-satu juga dapat digunakan jika ada dua jaringan yang menggunakan bentuk pengalamatan yang berbeda, misalnya Token Ring, yang kemudian terhubung ke Internet. Token Ring tidak menggunakan TCP/IP.

Yang lebih berguna dari kedua bentuk itu adalah satu-ke-banyak. Dengan menggunakan one-to-many NAT, sebuah organisasi dapat memiliki lebih banyak alamat IP yang tersedia secara internal daripada yang telah ditetapkan ke organisasi. Misalnya, bayangkan sebuah situs hanya diberi dua alamat IP: satu untuk server webnya dan yang lainnya untuk server NAT-nya. Secara eksternal, seluruh dunia memandang organisasi ini dengan dua alamat IP. Namun, secara internal, berkat terjemahan satu-ke-banyak, bisa ada lusinan, ratusan, atau ribuan alamat IP yang digunakan. Terserah perangkat yang melakukan terjemahan dari satu alamat eksternal ke alamat internal yang benar untuk memastikan bahwa pesan dirutekan dengan benar ke internal organisasi.

Ada dua masalah saat berhadapan dengan NAT. Yang pertama adalah pemetaan alamat eksternal ke internal yang tepat. Kami menggunakan tabel terjemahan untuk ini; namun, mekanisme di balik tabel ini berbeda antara NAT satu-ke-satu dan NAT satu-ke-banyak. Kami menjelaskan ini sebentar. Masalah lainnya berkaitan dengan checksum. Lapisan Internet akan menambahkan alamat IP (tujuan dan sumber) ke header IP. Seluruh header IP kemudian digunakan untuk menghitung checksum. Pesan tersebut kemudian diteruskan dari komputer lokal ke beberapa router internal dan akhirnya ke perangkat yang akan melakukan terjemahan NAT. Itu menukar alamat IP internal untuk alamat IP eksternal. Pada titik ini, checksum header tidak valid karena dihitung dengan alamat IP internal. Oleh karena itu, alat

penerjemah juga harus menghitung checksum baru, menggantikan yang lama dengan yang baru.

Mari kita pertimbangkan tabel terjemahan untuk NAT, seperti yang dikelola oleh perangkat penerjemah. Jika kami menjalankan NAT satu-ke-satu, maka satu-satunya persyaratan untuk tabel ini adalah menyimpan alamat IP internal dan alamat eksternal yang ditetapkan padanya. Misalnya, bayangkan kita memiliki tiga perangkat internal dengan alamat IP internal masing-masing 10.11.12.1, 10.11.12.15, dan 10.11.13.6. Saat paket dari salah satu perangkat ini mencapai perangkat penerjemah, perangkat menambahkan entri ke tabel dan memilih alamat eksternal yang tersedia. Tabelnya mungkin terlihat seperti ini:

10.11.12.1 → 192.114.63.14
10.11.12.15 → 192.114.63.22
10.11.13.6 → 192.114.63.40

Setiap kali sebuah paket masuk dari Internet dengan salah satu alamat yang ditunjukkan di atas di sebelah kanan, itu hanya dipetakan kembali ke alamat internal, header IP diubah untuk mencerminkan alamat IP internal yang tepat, dan checksum diperbarui. Paket melanjutkan perjalanannya.

Dalam NAT satu-ke-banyak, kami memiliki masalah: masing-masing dari tiga alamat internal di atas dipetakan ke alamat eksternal yang sama. Jika kami mencatat ini dalam sebuah tabel, mungkin terlihat seperti berikut:

10.11.12.1 → 192.114.63.14
10.11.12.15 → 192.114.63.14
10.11.13.6 → 192.114.63.14

Sekarang, paket eksternal tiba dengan alamat tujuan 192.114.63.14. Alamat IP internal mana yang kami gunakan? Jadi, mari kita pertimbangkan ini lebih jauh. Ketika 10.11.12.15 mengirim paket keluar, alamat tujuannya adalah 129.51.26.210. Jadi, tabel kita mungkin terlihat seperti berikut:

Source Mappings	Sent to
10.11.12.1 → 192.114.63.14	Some address
10.11.12.15 → 192.114.63.22	129.51.26.210
10.11.13.6 → 192.114.63.40	Some address

Sekarang, paket yang diterima dari 129.51.26.210 hingga 192.114.63.22 diidentifikasi sebagai pesan dari 10.11.12.15, dan karenanya, 10.11.12.15 menjadi alamat tujuan paket.

Apa yang terjadi jika dua perangkat internal atau lebih mengirimkan paket ke alamat IP eksternal yang sama (misalnya, www.google.com)? Solusi di atas dengan sendirinya tidak memadai. Solusi untuk masalah ini adalah dengan menggunakan port yang berbeda pada perangkat terjemahan. Jadi, sekarang, kami menambahkan informasi lain ke tabel.

Source Mappings	Sent to	Source Port
10.11.12.1 → 192.114.63.14	Some address	1025
10.11.12.15 → 192.114.63.22	129.51.26.210	1026
10.11.13.6 → 192.114.63.40	129.51.26.210	1027

Paket yang berasal dari 129.51.26.210 tetapi tiba di port 1026 diidentifikasi sebagai paket yang ditujukan untuk 10.11.12.15. Perhatikan bahwa dalam solusi untuk NAT ini, kami tidak hanya

menerjemahkan alamat tetapi juga porta. Tabel juga harus menyimpan port sumber asli yang digunakan oleh 10.11.12.15; jika tidak, pesan akan dialihkan kembali ke 10.11.12.15 tetapi ke port yang salah (port 1026 dalam kasus ini).

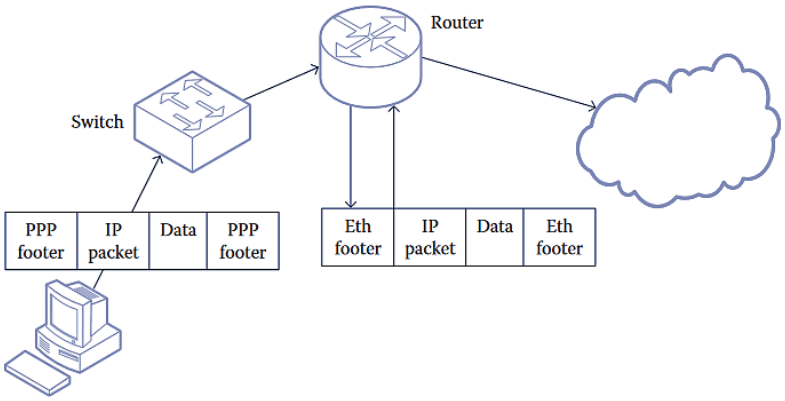
NAT adalah sarana yang sangat nyaman bagi organisasi untuk menambahkan keamanan ke sumber daya internalnya dengan melindungi alamat IP-nya dan memperluas jumlah alamat IP yang tersedia. Namun, IPv6 tidak memerlukan NAT, karena begitu banyak alamat IP yang tersedia. Jadi, meskipun NAT bisa ada di IPv6, itu tidak umum, setidaknya pada saat ini.

5.5 LAPISAN TAUTAN

Lapisan tautan secara kasar mencakup dua lapisan terbawah OSI: lapisan tautan data dan lapisan fisik. Kami menggunakan kata itu secara kasar karena ada beberapa pengecualian penting untuk perbandingan apa pun. Pertama, model OSI bukanlah sebuah implementasi, sehingga data link dan physical layer menjelaskan apa yang harus terjadi, sedangkan pada TCP/IP, link layer menjelaskan apa yang akan terjadi. Selain itu, banyak yang menganggap lapisan fisik OSI bukan bagian dari definisi TCP/IP. Artinya, TCP/IP terletak di atas implementasi fisik jaringan. Dengan demikian, beberapa administrator jaringan menganggap lapisan tautan, pada kenyataannya, menjadi dua lapisan terpisah, disebut sebagai lapisan tautan dan lapisan fisik. Kami menjelajahi lapisan tautan sebagai satu lapisan, di mana kami akan membatasi analisis kami hanya untuk melihat dua protokol: ARP dan *Network Discovery Protocol* (NDP). Kami tidak memeriksa salah satu mekanisme jaringan fisik (yang sudah dibahas dalam Bab 1).

Dalam TCP/IP, pemisahan antara lapisan Internet dan lapisan tautan kabur dalam beberapa hal. Router beroperasi pada lapisan Internet tetapi harus terlibat dengan lapisan tautan untuk melakukan terjemahan alamat perangkat keras ke jaringan. Ini ditangani di IPv4 dengan menggunakan ARP dan di IPv6 dengan menggunakan NDP. Ini bukan hanya domain switch atau hub, karena ada kalanya router harus mengetahui alamat perangkat keras.

Kedua, router ada di dalam dan di antara jaringan tetapi bukan merupakan titik akhir dari jaringan tersebut. Komputer akan mengemas paket TCP/IP bersama-sama, termasuk header (dan footer) lapisan tautan. Isi dari link layer header/footer dikhususkan untuk jenis jaringan yang ada pada level jaringan tersebut. Misalnya, komputer mungkin memiliki koneksi point-to-point, dan oleh karena itu, header dan footer akan menggunakan Point-to-Point Protocol (PPP). Titik akhir dari koneksi ini mungkin adalah router yang kemudian terhubung ke jaringan Ethernet. Header/footer PPP harus diganti dengan header/footer Ethernet yang sesuai. Ini adalah router yang menangani ini. Router, mengetahui jenis jaringan apa yang ada pada rute yang dipilih, memodifikasi paket TCP/IP dengan tepat sebelum meneruskan paket. Gambar 5.13 mengilustrasikan ide ini di mana switch jaringan komputer berkomunikasi dengan router melalui PPP. Router terhubung ke Internet melalui Ethernet. Oleh karena itu, router menukar header/footer PPP dengan header/footer Ethernet.



Gambar 5.13 Router saling menukar link layer header/footer.

Apa header dan footer yang kita lihat pada Gambar 5.13? Lapisan tautan, seperti lapisan transportasi dan Internet, menambahkan detailnya sendiri ke paket. Pada level ini, paket atau fragmen dari sebuah paket dikenal sebagai frame. Frame memiliki header prepended yang menyertakan informasi spesifikasinya sendiri untuk jenis jaringan tertentu (misalnya, Ethernet, PPP, dan Token Ring). Footer, juga disebut trailer, berisi checksum dalam bentuk bit CRC, sehingga perangkat penerima dapat segera menghitung jika ada kesalahan dalam pengiriman/penerimaan melalui media jaringan. Selain itu, padding bit dapat ditambahkan, sehingga seluruh paket memiliki ukuran yang seragam.

Sakelar di subnet lokal digunakan untuk mengirimkan pesan yang masuk dari jaringan ke sumber daya tujuan di subnet lokal. Switch tidak beroperasi pada lapisan Internet dan karena itu tidak menyadari alamat jaringan (misalnya, alamat IPv4). Alih-alih, sakelar beroperasi pada alamat perangkat keras. Bentuk alamat perangkat keras yang paling umum adalah alamat MAC. Meskipun alamat MAC pertama kali digunakan dalam jaringan Ethernet, banyak jenis jaringan lain yang sekarang menggunakan alamat MAC. Jadi, jaringan kami memerlukan mekanisme untuk menerjemahkan alamat IP menjadi alamat MAC saat pesan masuk dari jaringan dan alamat MAC menjadi alamat IP saat pesan keluar ke jaringan. ARP menangani ini.

Terjemahan alamat untuk ARP ditangani dengan menggunakan tabel ARP. Paling sederhana, tabel ARP mencantumkan alamat IP dan alamat MAC yang sesuai dari semua perangkat yang diketahuinya. Informasi tambahan mungkin direkam, jika tersedia, seperti jenis alamat IP (dinamis atau statis), jenis perangkat antarmuka (misalnya kartu Ethernet), nama host (jika diketahui), dan bendera. Bendera menunjukkan apakah suatu entri sudah lengkap, permanen, dan diterbitkan. Bahkan lebih banyak informasi dapat digunakan seperti usia dan kedaluwarsa alamat IP, jika dinamis, dan di mana informasi itu dipelajari (secara eksternal, terprogram, dan lainnya).

Untuk memanfaatkan ARP, pesan ARP harus dikirim dari satu perangkat ke perangkat lainnya. Biasanya, satu perangkat meminta informasi terjemahan dari yang lain. Misalnya, router mungkin meminta terjemahan dari sakelar atau komputer. Perhatikan bahwa pesan ARP tidak disebarkan ke seluruh jaringan. Artinya, mereka berada di dalam jaringan atau subnet. Pesan ARP itu sendiri terdiri dari beberapa bidang 2-byte. Ini termasuk jenis perangkat keras (misalnya, kartu Ethernet), protokol jaringan alamat (misalnya, IPv4), apakah pesannya

adalah permintaan atau tanggapan, dan perangkat keras sumber dan tujuan serta alamat jaringan. Jelas, untuk permintaan, alamat jaringan akan kosong. Satu bidang 2-byte tambahan mencantumkan panjang alamat perangkat keras dan alamat jaringan. Alamat MAC adalah 6 byte, dan alamat IPv4 adalah 4 byte, jadi entri ini biasanya masing-masing berisi 6 dan 4.

Tabel ARP disimpan di switch tetapi juga dapat disimpan di router. Namun, tabel ARP juga dapat disimpan setidaknya untuk sementara di komputer. Komputer yang berkomunikasi dengan perangkat lokal lain di jaringannya sendiri dapat menyimpan hasil ARP sehingga tidak perlu terus menggunakan ARP untuk komunikasi tersebut. Saat menerima respons ARP, informasi dalam respons ditambahkan ke tabel ARP lokalnya. Hal ini dapat menyebabkan situasi di mana data ARP yang di-cache secara lokal menjadi usang (misalnya, perangkat memperoleh alamat IP baru atau kartu antarmukanya diubah, dan dengan demikian, alamat MAC-nya diubah). Dari waktu ke waktu, sebaiknya bersihkan tabel ARP.

```
Interface:  10.15.8.19 --- 0xb
  Interface Address        Physical Address       Type
  10.15.11.235             78-2b-cb-b3-96-e8      dynamic
  192.168.56.255           ff-ff-ff-ff-ff-ff      static
  224.0.0.2                01-00-5e-00-00-02      static
  224.0.0.22               01-00-5e-00-00-16      static
  239.255.255.255          01-00-t3-7f-ff-fa      static
```

Di Unix/Linux, perintah arp menampilkan cache ARP yang di-cache. Perintah ini telah diganti dengan ip neigh (untuk tetangga). Di Windows, untuk menampilkan cache ARP saat ini, gunakan arp –s dari prompt baris perintah. Berikut ini adalah kutipan dari tabel ARP.

Perhatikan bahwa ff-ff-ff-ff-ff-ff adalah alamat MAC untuk pesan siaran yang akan dikirim ke semua perangkat di jaringan lokal.

Selain permintaan dan balasan ARP, ada dua fungsi ARP lain yang perlu diperhatikan. Yang pertama adalah probe ARP. Probe ARP dapat digunakan oleh perangkat yang ingin menguji alamat IP-nya untuk melihat apakah sudah ada di dalam jaringan. Ini berguna apakah perangkat diberi alamat IP statis atau dinamis, karena kecelakaan dapat terjadi dalam kedua kasus tersebut. Probe ARP memiliki alamat IP pengirim semua 0, menunjukkan bahwa ini adalah permintaan probe. Pesan tersebut akan berisi alamat MAC perangkat seperti biasa dan alamat IP yang diusulkan sebagai alamat target. Jika waktu yang cukup berlalu tanpa tanggapan, perangkat mengetahui bahwa ia dapat mulai menggunakan alamat IP ini.

Fungsi ARP lainnya dikenal sebagai pengumuman. Pengumuman ini dibuat untuk memperingatkan perangkat lain di jaringan lokal tentang keberadaannya. Ada dua alasan untuk pengumuman tersebut. Salah satunya adalah saat perangkat baru ditambahkan. Misalnya, saat menambahkan perute baru, perute harus diketahui keberadaannya sehingga komputer dan perute lain dapat menggunakannya. Alasan lainnya adalah untuk me-refresh tabel ARP yang di-cache jika nilainya berubah. Ini terkadang disebut sebagai pesan ARP serampangan. Saat pengumuman ARP dibuat, alamat jaringan perangkat ditempatkan di bidang alamat jaringan target, sedangkan alamat perangkat keras target diatur ke 0. Ini menunjukkan bahwa pesan ini tidak ditujukan untuk perangkat tertentu. Tidak ada tanggapan yang diharapkan dari pengumuman ARP.

ARP memiliki dua variasi protokol yang dikenal sebagai inverse ARP (InARP) dan reverse ARP (RARP). Peran kedua protokol ini adalah untuk melakukan pemetaan mundur.

Sementara ARP menerjemahkan dari alamat IP ke alamat MAC, InARP dan RARP menerjemahkan dari alamat MAC ke alamat IP. RARP tidak lagi berguna, karena DHCP dapat menangani permintaan semacam itu. InARP, meskipun berdasarkan pesan yang sama dengan ARP, jarang digunakan. Untuk IPv6, pendekatan serupa namun baru telah dibuat. Daripada mengandalkan ARP, IPv6 menggunakan NDP untuk mendapatkan alamat perangkat keras dari alamat jaringan. Namun, NDP melakukan lebih dari ini. Tabel 5.8 menjelaskan banyak fungsi NDP.

Tabel 5.8 Fungsi NDP

Nama	**Arti**
Alamat konfigurasi otomatis	Hasilkan alamat IPv6 tanpa menggunakan server DHCP
Resolusi alamat	Lakukan pemetaan alamat IPv6 ke MAC
Deteksi alamat duplikat	Ini sejalan dengan konfigurasi otomatis alamat; setelah alamat IPv6 dibuat, host perlu menentukan apakah ini alamat unik
Komunikasi tetangga	Router dapat berbagi alamat perangkat keras dengan router lain melalui pesan ajakan dan pesan iklan
Penemuan parameter	Perangkat dapat memperoleh informasi tentang sumber daya lain di jaringan seperti MTU router
Penemuan router	Perangkat dapat menemukan router apa yang dapat dijangkau dalam jaringannya. Perangkat juga dapat menentukan router mana yang akan digunakan sebagai hop pertama atau berikutnya dalam komunikasi tertentu melalui pengalihan pesan yang disediakan oleh router

BAB 6
ALAT PROTOKOL INTERNET

Transmission Control Protocol/Internet Protocol (TCP/IP) terdiri dari beberapa protokol yang berbeda, masing-masing digunakan untuk mengkomunikasikan jenis pesan yang berbeda. Sebagai pengguna Internet, rincian yang menyusun TCP/IP dan protokol individual ini biasanya tidak penting. Namun, bagi siapa saja yang berencana menggunakan server Internet atau mengimplementasikan perangkat lunak yang berkomunikasi melalui Internet, memahami TCP/IP sangatlah penting. Mendapatkan pemahaman tentang cara kerja protokol atau melihatnya dalam tindakan dapat menjadi tantangan jika Anda hanya membaca tentang protokol. Untungnya, tersedia alat yang memungkinkan Anda menangkap dan melihat paket TCP/IP, berkomunikasi langsung melalui port tertentu, memeriksa port dan antarmuka jaringan, dan memeriksa aspek lain dari jaringan lokal atau Internet Anda. Kami fokus pada beberapa alat yang berguna dalam bab ini.

Bab ini dibagi menjadi empat bagian. Pertama, kami menjelajahi alat penangkap paket. Wireshark adalah program penganalisis protokol jaringan sumber terbuka berbasis antarmuka pengguna grafis (GUI). Wireshark tersedia di Windows, Mac OS X, Linux, dan banyak versi Unix. TShark adalah versi berbasis teks dari Wireshark yang menawarkan fungsionalitas serupa. tcpdump, tersedia di Linux dan Unix, berbasis teks seperti TShark. Namun, tidak seperti Wireshark dan TShark, ini dibatasi untuk menangkap paket hanya dari tiga protokol. Ini juga memiliki kemampuan penyaringan yang lebih sedikit. Program lain disebut dumpcap, yang merupakan program penangkap paket yang digunakan Wireshark (kami tidak akan memeriksa TShark atau dumpcap di bab ini). Bagian kedua berfokus pada netcat (atau nc), yang sering disebut sebagai pisau komunikasi jaringan tentara Swiss. Ini adalah program berbasis teks yang memungkinkan Anda membaca dan menulis langsung melalui koneksi jaringan dengan menggunakan TCP dan User Datagram Protocol (UDP). Netcat/nc dan varian yang lebih baru bernama ncat tersedia di Linux/Unix dan Windows. Bagian ketiga berfokus pada program jaringan Unix/Linux tertentu yang berhubungan dengan memperoleh, menampilkan, dan mengubah alamat IP. Secara khusus, kami melihat ip di Linux/Unix dan ipconfig di Windows (dan program Linux/Unix yang lebih lama ifconfig tercakup dalam bacaan online). Kami juga melihat program jaringan lainnya: ping, traceroute, netstat, dan melihat fungsi ip lainnya. ping, traceroute, dan netstat tersedia di Linux/Unix, Mac OS X, dan Windows. ip tidak tersedia di Windows. Terakhir, kami melihat program yang menggunakan jaringan daripada yang menguji jaringan. Kami akan fokus pada ssh, ftp, dan sftp. Kami juga melihat program pencarian DNS nslookup (juga tersedia di Windows), whois, host, dan dig.

6.1 PROGRAM PENANGKAPAN PAKET

Dalam kebanyakan kasus, komunikasi jaringan tidak jelas bagi pengguna. Daripada melihat paket-paket jaringan, pengguna disajikan dengan produk akhir, dan paket-paket tersebut disusun menjadi satu pesan dan ditampilkan kepada pengguna di lapisan aplikasi.

Meskipun ini lebih dari cukup untuk sebagian besar pengguna komputer, administrator jaringan dan pemrogram soket jaringan mungkin perlu menjelajahi paket itu sendiri. Misalnya, administrator jaringan mungkin perlu menganalisis paket untuk melihat mengapa jaringan menjatuhkan beberapa paket dan bukan yang lain atau untuk mencari masalah keamanan seperti intrusi dan keberadaan malware. Administrator jaringan mungkin juga ingin menjelajahi lalu lintas jaringan untuk meningkatkan efisiensi atau untuk mengumpulkan statistik. Pemrogram soket mungkin perlu menjelajahi paket untuk melihat bagaimana protokol digunakan dan/atau untuk memanfaatkan protokol secara lebih efisien. Pada bagian ini, kami memeriksa dua program penangkap paket: Wireshark dan tcpdump.

Wireshark

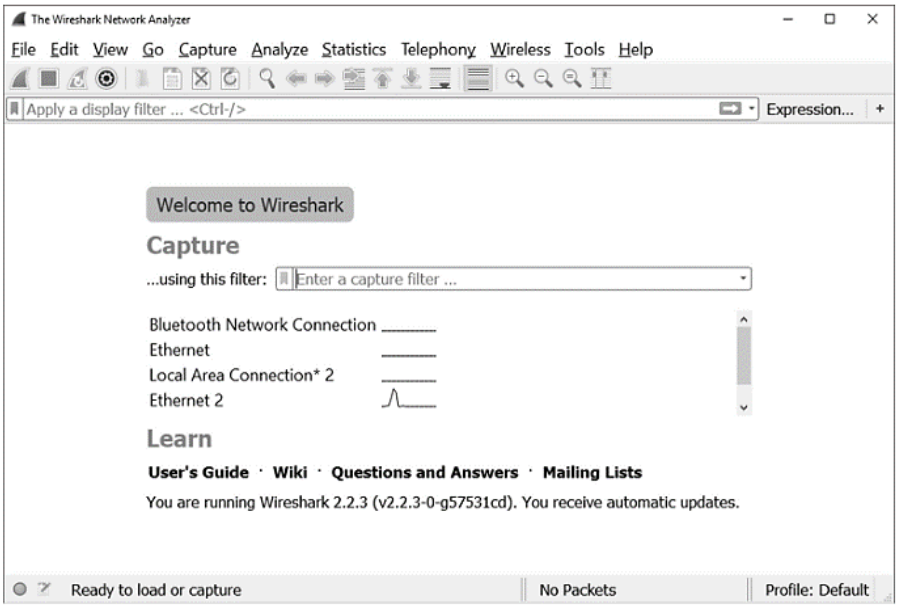
Program Wireshark awalnya dikembangkan oleh Gerald Combs dengan nama Ethereal. Rilis pertama adalah pada bulan Juli 1998. Pada tahun 2006, Combs mengganti nama perangkat lunak tersebut menjadi Wireshark, dengan versi lengkap pertama dirilis pada tahun 2008. Ratusan orang telah terlibat dalam pengembangan Ethereal/Wireshark, banyak di antaranya memiliki keahlian dengan TCP/IP . Wireshark adalah perangkat lunak sumber terbuka yang diterbitkan di bawah Lisensi Publik Umum GNU, artinya Anda bebas mengunduh, menginstal, memodifikasi, dan menggunakan Wireshark (namun, Anda tidak bebas menggunakan Wireshark sebagai bagian dari produk komersial yang Anda mungkin menghasilkan). Meskipun Anda dapat menginstal Wireshark dari kode sumber, kecuali jika Anda berencana untuk memodifikasi kode, yang terbaik adalah mengunduh versi yang dapat dieksekusi agar program penginstalan dapat menginstal semua komponen yang sesuai. Dapat dieksekusi ada untuk sebagian besar platform.

Kunjungi wireshark.org untuk mengunduh versi. Saat tulisan ini dibuat, versi Wireshark saat ini adalah versi 2.4.0. Perhatikan bahwa saat menginstal Wireshark di Windows, Anda akan diminta untuk menginstal WinPcap. Ini adalah pustaka libpcap versi Windows, diperlukan untuk mengambil data jaringan langsung.

Wireshark dianggap sebagai penganalisa protokol jaringan. Jadi, meskipun mampu menangkap paket jaringan, ini adalah alat yang menyediakan platform untuk analisis yang lebih mendalam. Gambar 6.1 menunjukkan antarmuka awal Wireshark. Pada gambar ini, Anda dapat menemukan tiga area utama. Di sepanjang bagian atas terdapat bilah menu, bilah tombol, dan kotak untuk memasukkan ekspresi yang akan difilter (ini akan dijelaskan nanti). Di bawahnya, jendela utama dibagi menjadi dua area: Tangkap dan Pelajari. Di bawah Capture, Anda menemukan berbagai antarmuka yang dapat didengarkan Wireshark. Dalam hal ini, hanya Ethernet 2 yang menunjukkan aktivitas apa pun. Di bawah Belajar, terdapat tautan ke berbagai sumber daya bantuan.

Kami akan membatasi pemeriksaan Wireshark kami untuk menangkap dan menganalisis paket. Apa yang dimaksud dengan menangkap? Artinya, semua komunikasi jaringan, dalam bentuk paket, akan disimpan untuk Anda periksa. Ingatlah bahwa komunikasi jaringan terdiri dari paket-paket yang dikirimkan dari komputer Anda ke perangkat jarak jauh dan dari perangkat jarak jauh kembali ke komputer Anda. Perangkat jarak jauh dapat berupa komputer lain (klien atau server) atau perangkat siaran seperti gateway dan router. Jadi,

penangkapan adalah proses mendapatkan dan menyimpan masing-masing paket yang dipertukarkan.



Gambar 6.1 Wireshark.

Untuk memulai penangkapan, Anda harus menentukan antarmuka terlebih dahulu. Komputer Anda mungkin memiliki satu antarmuka, yaitu perangkat yang mengarah ke koneksi Internet Anda. Ini mungkin berupa kartu antarmuka jaringan nirkabel atau kabel (NIC) (biasanya kartu Ethernet). Pada Gambar 6.1, antarmuka diberi label sebagai Ethernet 2. Anda dapat memilih antarmuka Anda dengan mengklik label di bawah bagian Capture. Dari bilah tombol atau menu, Anda kemudian dapat mengontrol memulai dan menghentikan pengambilan paket. Tombolnya adalah simbol Wireshark ( ), atau Anda dapat memilih Mulai dari menu Tangkap.

Berapa banyak paket yang kita bicarakan? Itu tergantung pada proses dan layanan apa yang sedang berjalan di komputer Anda. Pertukaran Hypertext Transfer Protocol (HTTP) tunggal dapat terdiri dari lusinan, ratusan, atau ribuan paket. Permintaan HTTP pendek dan mungkin dibagi di antara beberapa paket, tetapi responsnya bervariasi berdasarkan jumlah data yang dikembalikan dan secara harfiah dapat mengambil ribuan paket. Selain itu, sumber daya web mungkin menyertakan referensi ke file lain (mis., file gambar). Saat menerima file asli, klien akan membuat banyak permintaan tambahan.

Apakah paket akan ditukar jika saat ini Anda tidak melakukan akses jaringan? Hampir pasti, karena walaupun Anda mungkin tidak aktif menggunakan aplikasi, komputer Anda memiliki layanan latar belakang yang akan berkomunikasi dengan perangkat jaringan. Layanan ini mungkin mencakup, misalnya, klien Sistem Nama Domain (DNS) yang berkomunikasi dengan server nama DNS lokal Anda, klien Protokol Konfigurasi Host Dinamis (DHCP) yang berkomunikasi dengan router lokal, program firewall Anda, jaringan lokal yang disinkronkan dengan klien, dan aplikasi yang menggunakan jaringan seperti aplikasi jam atau cuaca. Selain itu, jika Anda tersambung ke utilitas penyimpanan cloud seperti Dropbox atau OneDrive,

komputer Anda akan terus berkomunikasi dengan server cloud untuk memastikan koneksi terbuka.

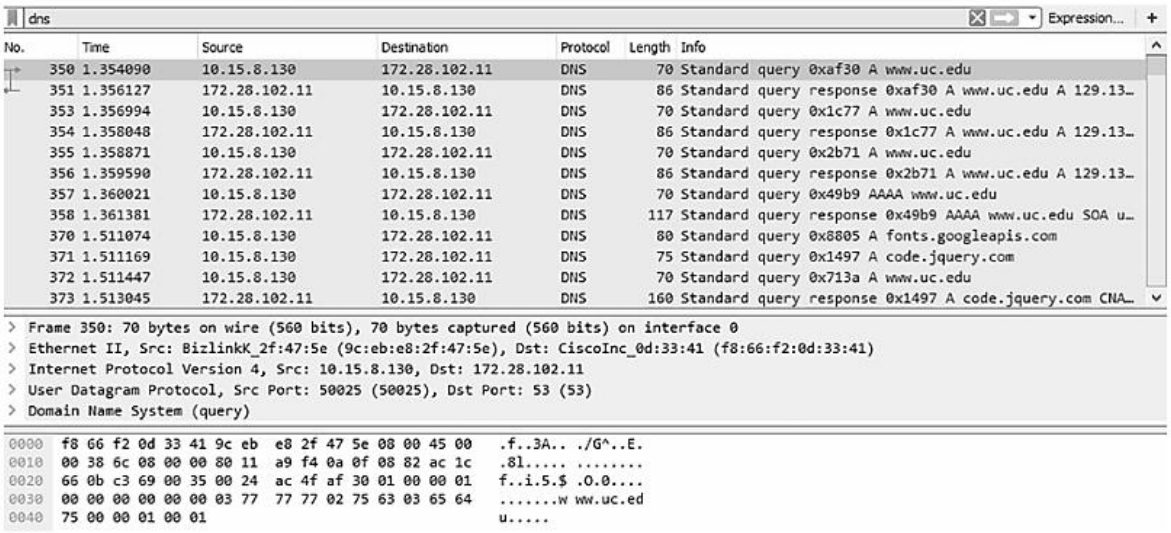
Saat Anda menangkap paket, GUI Wireshark berubah menjadi tiga area. Area teratas adalah daftar paket. Daftar ini berisi deskripsi singkat dari setiap paket yang diambil. Secara khusus, paket diberi nomor identifikasi (dimulai dari 1 setiap kali Anda memulai sesi penangkapan baru), waktu (sejak awal penangkapan), alamat IP perangkat sumber dan perangkat tujuan (IPv6 jika tersedia), protokol, ukuran paket dalam byte, dan informasi tentang paket seperti jenis operasi yang menyebabkan paket dikirim atau diterima. Pada Gambar 6.2, kita melihat daftar singkat paket yang ditangkap oleh Wireshark. Dalam hal ini, tidak ada pesan yang secara khusus disebabkan oleh aplikasi pengguna (mis., pengguna saat ini tidak menjelajahi web atau mengakses server email). Perhatikan beragam protokol paket dalam tangkapan singkat ini (TLSv1.2, TCP, DHCPv6, dan lainnya tidak ditampilkan di sini).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::c93d:c212:aff…	ff02::1:2	DHCPv6	163	Solicit XID: 0x7730d9 CID: 000100011f491a5d782bcb…
2	0.030178	CiscoInc_e6:78:82	PVST+	STP	64	Conf. Root = 24576/22/f8:66:f2:0d:33:41 Cost = 1…
3	0.035847	CiscoInc_e6:78:82	PVST+	STP	64	Conf. Root = 24576/91/f8:66:f2:0d:33:41 Cost = 1…
4	0.167032	10.15.8.208	10.15.11.255	BJNP	60	Printer Command: Unknown code (2)
5	0.167033	10.15.8.208	224.0.0.1	BJNP	60	Printer Command: Unknown code (2)
6	0.251486	Dell_59:b7:d7	Broadcast	ARP	60	Who has 10.15.8.221? Tell 10.15.8.54
7	0.345226	10.15.9.161	255.255.255.255	UDP	304	41794→41794 Len=262
8	0.428455	10.15.8.126	255.255.255.255	DB-LSP…	254	Dropbox LAN sync Discovery Protocol
9	0.428628	10.15.8.126	10.15.11.255	DB-LSP…	254	Dropbox LAN sync Discovery Protocol
10	0.537965	10.15.9.161	255.255.255.255	UDP	304	41794→41794 Len=262
11	0.625879	Dell_9a:6c:bb	Broadcast	ARP	60	Who has 10.15.9.37? Tell 10.15.9.90
12	0.737679	10.15.9.161	255.255.255.255	UDP	304	41794→41794 Len=262
13	1.190921	10.15.8.111	10.15.11.255	UDP	86	57621→57621 Len=44
14	1.214888	10.15.8.178	255.255.255.255	UDP	82	65175→1947 Len=40
15	1.219536	Dell_9a:6c:bb	Broadcast	ARP	60	Who has 10.15.9.37? Tell 10.15.9.90
16	1.324106	fe80::887e:6466:db3…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
17	1.330580	fe80::887e:6466:db3…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
18	1.330847	fe80::887e:6466:db3…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
19	1.331768	Dell_54:30:bc	Broadcast	ARP	60	Who has 10.15.8.1? Tell 10.15.8.34

Gambar 6.2 Contoh paket.

Panel tengah GUI Wireshark memberikan detail paket dari setiap paket yang dipilih, sedangkan panel bawah berisi data aktual paket (bita paket) yang tercantum dalam heksadesimal dan Kode Standar Amerika untuk Pertukaran Informasi (ASCII) (jika data berkaitan dengan cetak - Mampu karakter ASCII). Untuk memeriksa paket tertentu, klik paket tersebut di daftar paket. Saat memilih paket, paket akan diperluas di detail paket dan panel data paket. Detail paket akan mencakup ringkasan, data Ethernet (kontrol akses media [MAC] alamat perangkat sumber dan tujuan, serta jenis atau pabrikan perangkat), data IP, data datagram (TCP atau UDP), dan mungkin protokol- informasi spesifik. Untuk salah satu dari informasi ini, Anda dapat mengembangkannya untuk mendapatkan lebih banyak informasi.

Mari kita telusuri contoh tindakan Wireshark dan informasi yang dapat kita peroleh. Dalam hal ini, kami akan mempertimbangkan komunikasi tipikal yang terlibat saat Anda membuat permintaan di browser web Anda (klien web) ke server web. Secara khusus, kami akan mengeluarkan permintaan HTTP melalui kotak alamat browser web kami dengan menggunakan URL www.uc.edu. Kami berasumsi bahwa alias IP ini tidak di-cache di tabel DNS lokal atau tabel host komputer klien dan bahwa data situs ini tidak di-cache di cache browser web.

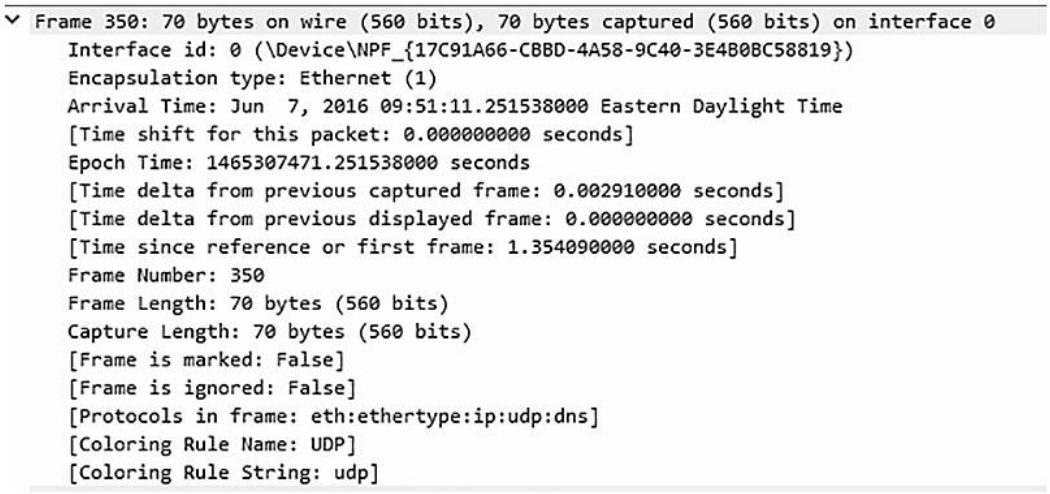


Gambar 6.3 Paket DNS.

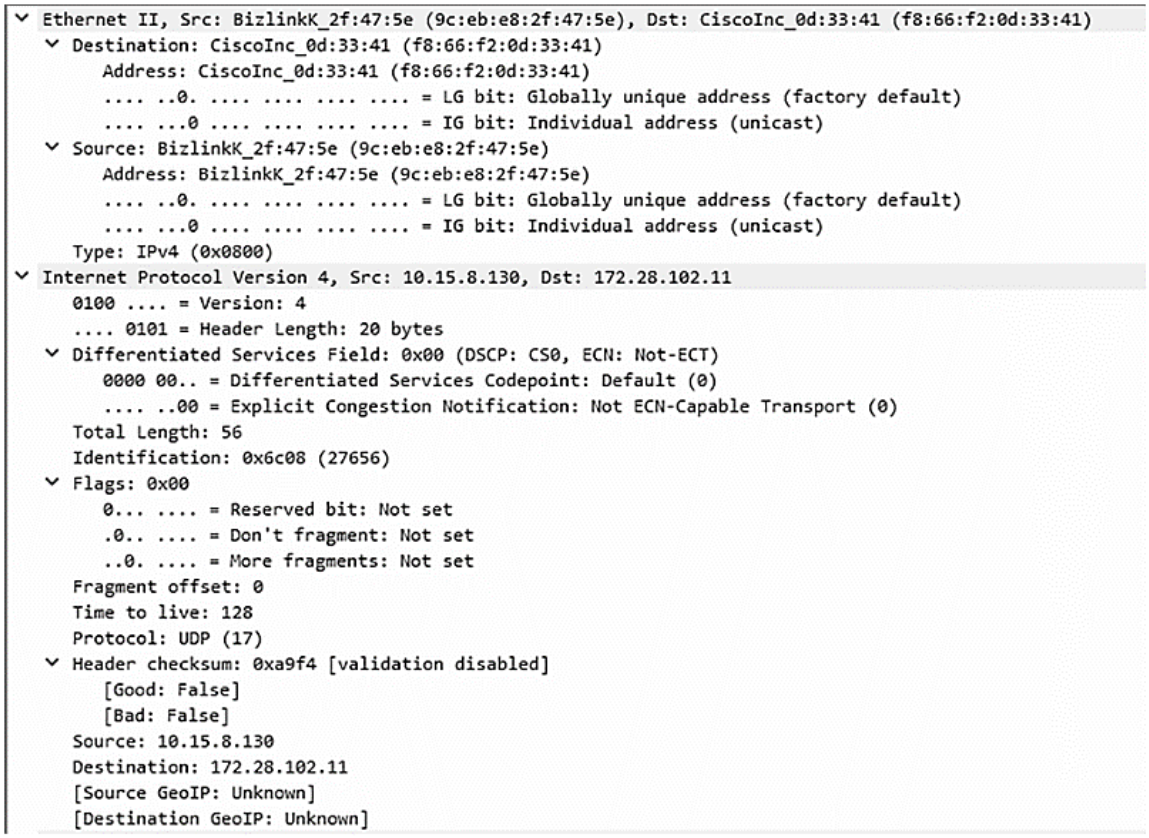
Saat mengeluarkan permintaan HTTP, langkah pertama adalah klien web menyelesaikan alamat IP. Ini membutuhkan permintaan DNS. Meskipun di Wireshark kita akan melihat lusinan atau ratusan paket, jika kita ingin memeriksa kueri dan respons DNS, kita dapat menggunakan fitur filter untuk membatasi paket yang ditampilkan di panel atas agar sesuai dengan kriteria yang dimasukkan. Di kotak filter, kami memasukkan "dns." Ini menyebabkan hanya paket DNS yang ditampilkan. Pada Gambar 6.3, kita melihat daftar singkat ini di bagian daftar paket jendela, bersama dengan rincian paket.

Kami memilih entri pertama dalam daftar paket (kueri dari 10.15.8.130 hingga 172.28.102.11 untuk kueri IP alias www.uc.edu). Dalam memilih paket ini, panel tengah diisi dengan detail. Kelima entri di panel rincian paket memiliki > di sebelahnya. Memperluas ini memberi kita lebih banyak detail.

Yang pertama dari lima detail paket adalah deskripsi dari frame: nomor frame, ukuran, dan nomor antarmuka. Detail yang diperluas ditunjukkan pada Gambar 6.4. Dalam memperluas informasi pada perangkat antarmuka (Ethernet II), kami menemukan alamat sumber (komputer kami) dan tujuan (server DNS). Informasi yang lebih menarik diperoleh saat memperluas detail yang tercantum di bawah IPv4. Di sini, kita melihat beberapa detail yang ditemukan dalam bagian IP dari paket: panjang header, waktu aktif, jenis protokol, dan checksum header. Detail tentang Ethernet dan IPv4 ditunjukkan pada Gambar 6.5.



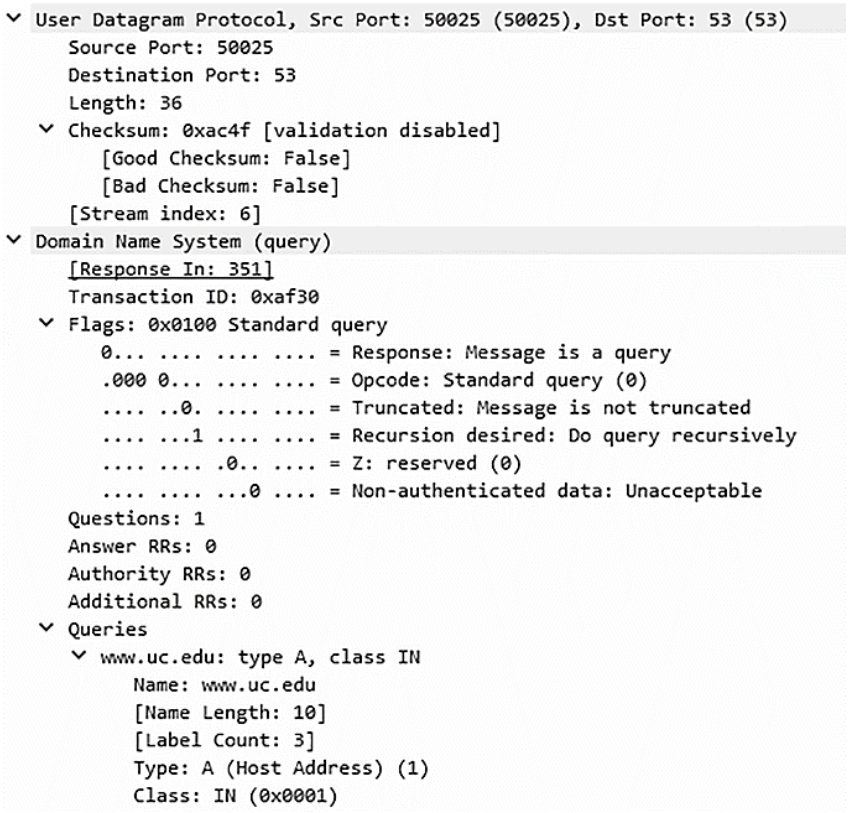
Gambar 6.4 Detail bingkai.



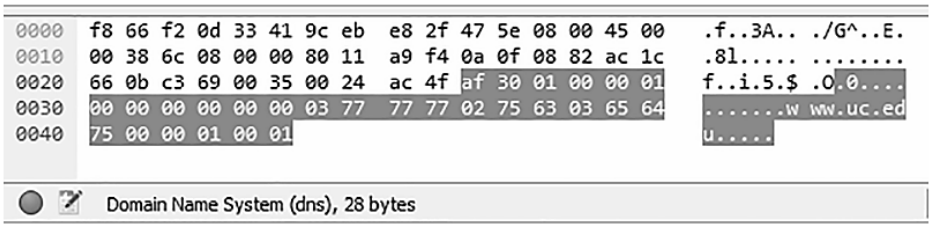
Gambar 6.5 Rincian Ethernet II dan IPv4.

Item berikutnya dalam detail paket berisi data dari datagram UDP. Saat memperluas ini, kami menemukan port sumber dan tujuan serta checksum. Checksum adalah seluruh paket (inilah mengapa tidak cocok dengan checksum dari bagian IPv4). Item terakhir dari detail paket berisi informasi tentang berisi informasi tentang data paket, yang dalam hal ini adalah kueri DNS. Kami melihat berbagai tanda kueri yang digunakan oleh server DNS, bagian-bagian dalam paket DNS. Karena ini adalah permintaan, ini berisi pertanyaan tetapi tidak ada

jawaban, bagian otoritas, dan catatan sumber daya tambahan (RR). Akhirnya, kita melihat kueri. Kedua bagian ini ditunjukkan pada Gambar 6.6.



Gambar 6.6 Detail kueri UDP dan DNS.

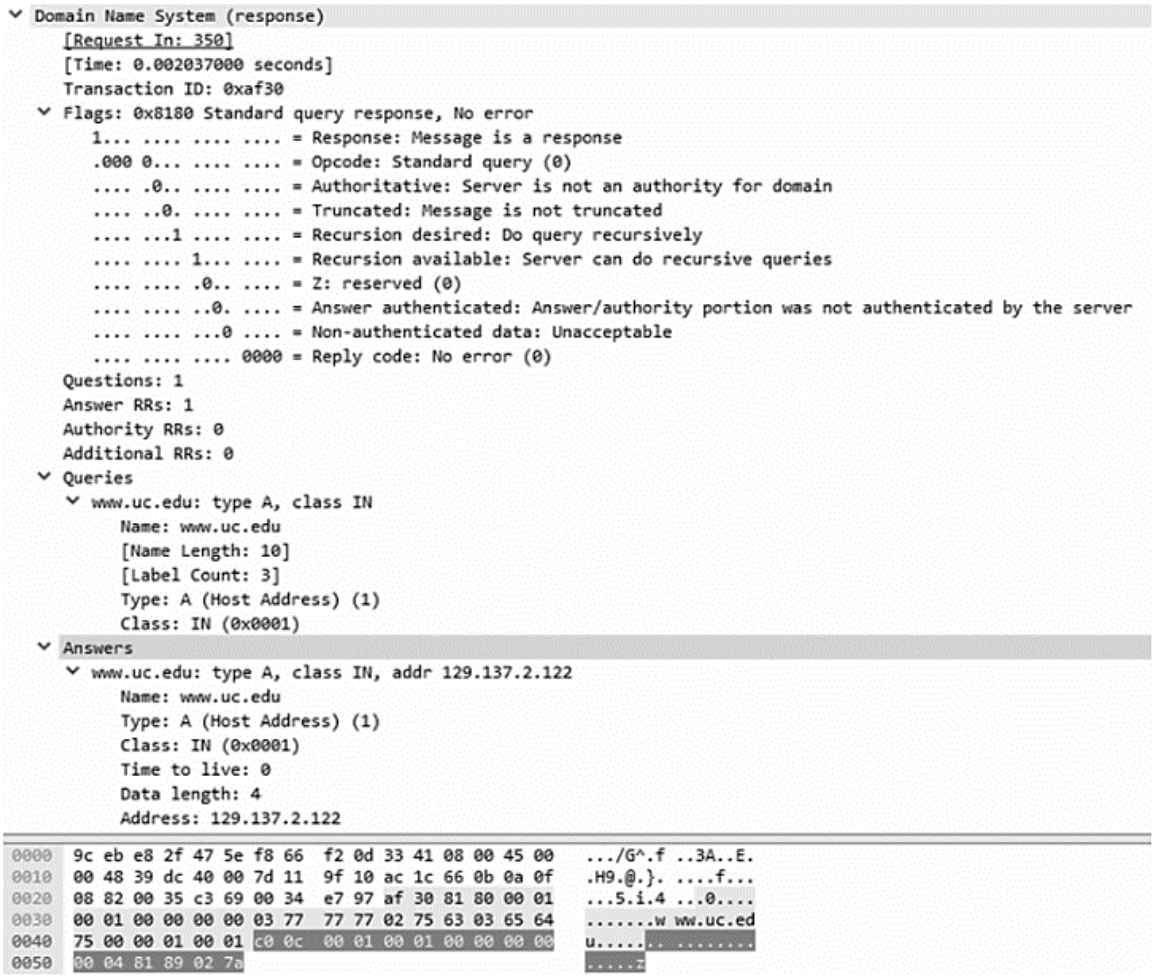


Gambar 6.7 Data paket untuk kueri DNS.

Bagian byte paket dari Wireshark menampilkan seluruh paket sebagai data. Untuk kueri DNS yang kita lihat pada Gambar 6.6, datanya ditunjukkan pada Gambar 6.7. Secara default, ini ditampilkan dalam heksadesimal (namun, Anda juga dapat melihatnya dalam biner) dan padanan ASCII (jika dapat dicetak). Bagian DNS (kueri) dari detail paket disorot, jadi, kita melihat bahwa bagian dari byte paket juga disorot. Artinya, 28 byte terakhir (atau 28 pasang digit heksadesimal terakhir) adalah kueri DNS itu sendiri. Ini terdiri dari alias IP (www.uc.edu), tipe (A), dan kelas (IN).

Anda dapat melihat bagian dalam ASCII yang dapat dicetak untuk www.uc.edu, yang dinyatakan dalam heksadesimal sebagai 77 77 77 02 75 63 03 65 64 75, yang secara harfiah menyatakan ekuivalen heksadesimal untuk setiap karakter di www.uc.edu. Perhatikan bahwa

dua periode memiliki nilai yang berbeda, 02 dan 03. Sebelum bagian data ini, kita dapat melihat bagian lain dari kueri kita. Misalnya, bendera direpresentasikan sebagai 01 00 dalam heksadesimal, yang sebenarnya adalah 0000 0001 dalam biner. Bit-bit ini adalah flag individu dari kueri dengan jenis kueri (0 pertama), operasi yang diminta (000 0), jangan potong (0), rekursi diinginkan (1), dicadangkan (0), dan data yang tidak diautentikasi tidak dapat diterima (0). Kita juga bisa melihat nilai jumlah pertanyaan (1), RR jawaban, RR otoritas, dan RR tambahan (semuanya 0) di data. Sebelumnya dalam data, kita dapat menemukan, misalnya, checksum (ac 4f), alamat, panjang, dan sebagainya.



Gambar 6.8 Bagian respons DNS dan data paket.

Paket DNS berikutnya, bernomor 351 pada Gambar 6.3, memberikan respons dari server DNS kami. Anda mungkin memperhatikan bahwa alamat IP dibalik pada Gambar 6.3 daripada pesan dari 10.15.8.130 (komputer klien kami) ke 172.28.102.11 (server DNS lokal), paket diteruskan dari 172.28.102.11 ke 10.15.8.130 . Detail paket dicantumkan dengan menggunakan lima area yang sama: Frame, Ethernet, IPv4, UDP, dan DNS, tetapi dalam kasus ini, ini adalah respons DNS, bukan kueri. Kami mengabaikan detail untuk empat area pertama, karena satu-satunya perbedaan yang signifikan adalah alamat (terbalik dari apa yang telah kami jelajahi sebelumnya): waktu dan checksum. Namun, respons DNS berisi informasi yang

berbeda termasuk bagian Jawaban. Demikian pula, data paket berbeda. Kita lihat perbedaan respon DNS, seperti terlihat pada Gambar 6.8.

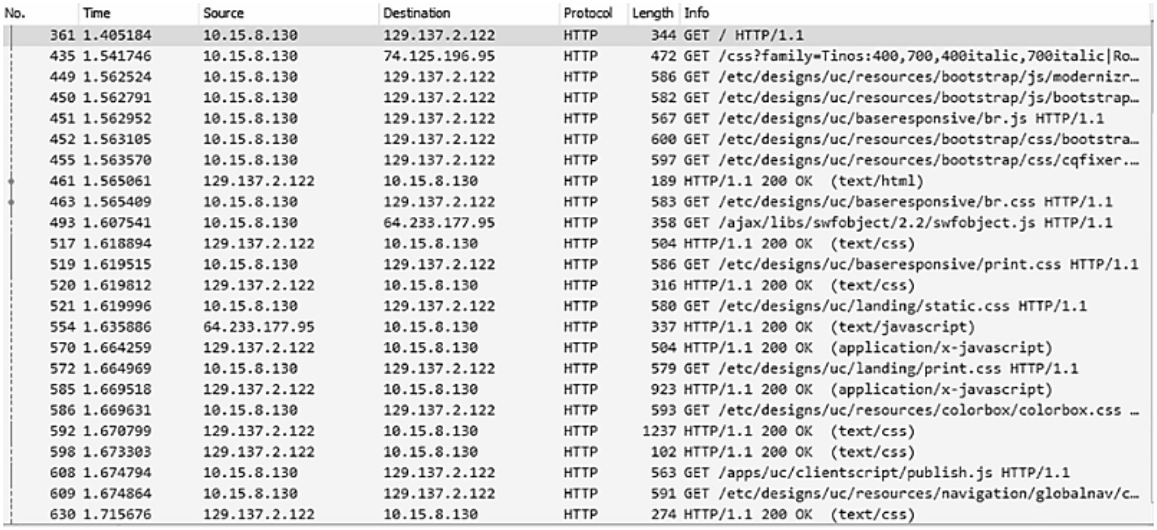
Hal pertama yang perlu diperhatikan di bagian DNS (respons) adalah menyertakan tautan ke kueri DNS (bingkai 38). Serupa dengan permintaan, kita melihat bendera kueri dan isi paket: sebuah pertanyaan dan dua tanggapan RR. Pertanyaannya hanyalah pernyataan ulang dari permintaan awal.

Bagian Jawaban berisi dua bagian: pertama adalah catatan sumber daya untuk www.uc.edu. RR kedua adalah tipe A, yang memetakan dari alias IP ke alamat IP. Di sini, kita melihat bahwa www.uc.edu adalah alias untuk 129.137.2.122. Perhatikan bahwa catatan sumber daya memiliki waktu untuk masuk langsung. Jika Anda bingung tentang beberapa perincian tentang DNS ini, kami akan menjelajahi DNS secara terperinci, termasuk RR, di Bab 5 dan 6.

Sekarang komputer klien kami memiliki alamat IP URL dari browser web, itu dapat mengirimkan permintaan HTTP. Gambar 6.3 hanya menampilkan filter DNS. Kami hanya dapat melihat daftar paket HTTP dengan mengubah filter dari dns ke http. Sebagian dari tampilan terfilter ini ditunjukkan pada Gambar 6.9. Perhatikan bahwa ada dua bentuk dasar pesan yang ditunjukkan pada gambar ini: yang dimulai dengan GET, yang merupakan permintaan HTTP, dan yang menyatakan HTTP/1.1 200 OK, yang merupakan tanggapan.

Mengapa ada begitu banyak paket untuk satu permintaan HTTP? Banyak halaman web mereferensikan sumber lain. Dengan halaman yang dibuat secara dinamis (misalnya, melalui eksekusi skrip sisi server), sumber daya ini ditambahkan ke halaman web pada saat halaman diambil dan/atau dibuat oleh server web. Namun, dalam banyak kasus, sumber daya ini direferensikan menggunakan tag HTML. Dalam kasus seperti itu, halaman web dikembalikan ke klien Anda, dan browser Anda mengirimkan permintaan HTTP tambahan untuk mendapatkan sumber daya tersebut.

Gambar 6.10 memberikan detail paket untuk paket HTTP pertama (seperti yang disorot pada Gambar 6.9). Paket ini berisi permintaan HTTP yang menentukan GET / HTTP/1.1 atau baris pertama permintaan HTTP awal kami. Permintaan ini mengatakan "dapatkan file yang URL-nya / menggunakan HTTP versi 1.1." URL / dipetakan ke direktori teratas server web. Biasanya, jika tidak ada nama file yang diberikan, defaultnya adalah file yang namanya index, seperti pada index.html atau index.php. Server web akan mengetahui cara menangani permintaan ini. Pada Gambar 6.10, kita melihat bahwa ini adalah paket TCP dan bukan paket UDP. Mengapa? Ini karena TCP dapat diandalkan, sedangkan UDP tidak. Protokol DNS menggunakan UDP, sedangkan HTTP menggunakan TCP.



No.	Time	Source	Destination	Protocol	Length	Info
361	1.405184	10.15.8.130	129.137.2.122	HTTP	344	GET / HTTP/1.1
435	1.541746	10.15.8.130	74.125.196.95	HTTP	472	GET /css?family=Tinos:400,700,400italic,700italic|Ro…
449	1.562524	10.15.8.130	129.137.2.122	HTTP	586	GET /etc/designs/uc/resources/bootstrap/js/modernizr…
450	1.562791	10.15.8.130	129.137.2.122	HTTP	582	GET /etc/designs/uc/resources/bootstrap/js/bootstrap…
451	1.562952	10.15.8.130	129.137.2.122	HTTP	567	GET /etc/designs/uc/baseresponsive/br.js HTTP/1.1
452	1.563105	10.15.8.130	129.137.2.122	HTTP	600	GET /etc/designs/uc/resources/bootstrap/css/bootstra…
455	1.563570	10.15.8.130	129.137.2.122	HTTP	597	GET /etc/designs/uc/resources/bootstrap/css/cqfixer.…
461	1.565061	129.137.2.122	10.15.8.130	HTTP	189	HTTP/1.1 200 OK (text/html)
463	1.565409	10.15.8.130	129.137.2.122	HTTP	583	GET /etc/designs/uc/baseresponsive/br.css HTTP/1.1
493	1.607541	10.15.8.130	64.233.177.95	HTTP	358	GET /ajax/libs/swfobject/2.2/swfobject.js HTTP/1.1
517	1.618894	129.137.2.122	10.15.8.130	HTTP	504	HTTP/1.1 200 OK (text/css)
519	1.619515	10.15.8.130	129.137.2.122	HTTP	586	GET /etc/designs/uc/baseresponsive/print.css HTTP/1.1
520	1.619812	129.137.2.122	10.15.8.130	HTTP	316	HTTP/1.1 200 OK (text/css)
521	1.619996	10.15.8.130	129.137.2.122	HTTP	580	GET /etc/designs/uc/landing/static.css HTTP/1.1
554	1.635886	64.233.177.95	10.15.8.130	HTTP	337	HTTP/1.1 200 OK (text/javascript)
570	1.664259	129.137.2.122	10.15.8.130	HTTP	504	HTTP/1.1 200 OK (application/x-javascript)
572	1.664969	10.15.8.130	129.137.2.122	HTTP	579	GET /etc/designs/uc/landing/print.css HTTP/1.1
585	1.669518	129.137.2.122	10.15.8.130	HTTP	923	HTTP/1.1 200 OK (application/x-javascript)
586	1.669631	10.15.8.130	129.137.2.122	HTTP	593	GET /etc/designs/uc/resources/colorbox/colorbox.css …
592	1.670799	129.137.2.122	10.15.8.130	HTTP	1237	HTTP/1.1 200 OK (text/css)
598	1.673303	129.137.2.122	10.15.8.130	HTTP	102	HTTP/1.1 200 OK (text/css)
608	1.674794	10.15.8.130	129.137.2.122	HTTP	563	GET /apps/uc/clientscript/publish.js HTTP/1.1
609	1.674864	10.15.8.130	129.137.2.122	HTTP	591	GET /etc/designs/uc/resources/navigation/globalnav/c…
630	1.715676	129.137.2.122	10.15.8.130	HTTP	274	HTTP/1.1 200 OK (text/css)

Gambar 6.9 paket HTTP.

Memperluas detail TCP, kami menemukan lebih banyak informasi daripada yang kami lihat dengan paket UDP. Misalnya, kami memiliki informasi pengurutan, sehingga paket-paket akan disusun dalam urutan yang benar. Kami juga melihat lebih banyak flag yang digunakan di TCP daripada yang kami lakukan di UDP. Selain itu, analisis jabat tangan tiga arah TCP (SEQ/ACK) ditampilkan di sini. UDP tidak menggunakan jabat tangan tiga arah. Bagian terakhir memberikan detail data paket, yang dalam hal ini adalah pesan HTTP. Kami melihat permintaan lengkap. Baris GET hanyalah baris pertama dari permintaan HTTP. Baris berikut berisi argumen Host, *User-Agent*, *Accept, Accept-Language, Accept-Encoding*, dan *Connection. Accept, Accept-Language,* dan *Accept-Encoding* digunakan untuk negosiasi konten. Kami akan menjelajahi semua argumen ini (dikenal sebagai header HTTP) di Bab 7 dan 8.

Menanggapi permintaan ini, server web mengembalikan 51 paket berbeda (Anda dapat melihat beberapa di antaranya dalam daftar yang ditunjukkan pada Gambar 6.9). 51 paket ini berisi halaman web yang dikembalikan serta potongan informasi lain yang akan diminta browser web Anda sebagai respons terhadap halaman web seperti file skrip java dan file css. Halaman web yang dikembalikan juga akan memiliki referensi ke objek web lain yang mungkin tidak dihosting oleh www.uc.edu, dan karenanya, Anda juga akan melihat permintaan HTTP ke tujuan lain seperti 64.233.177.95.

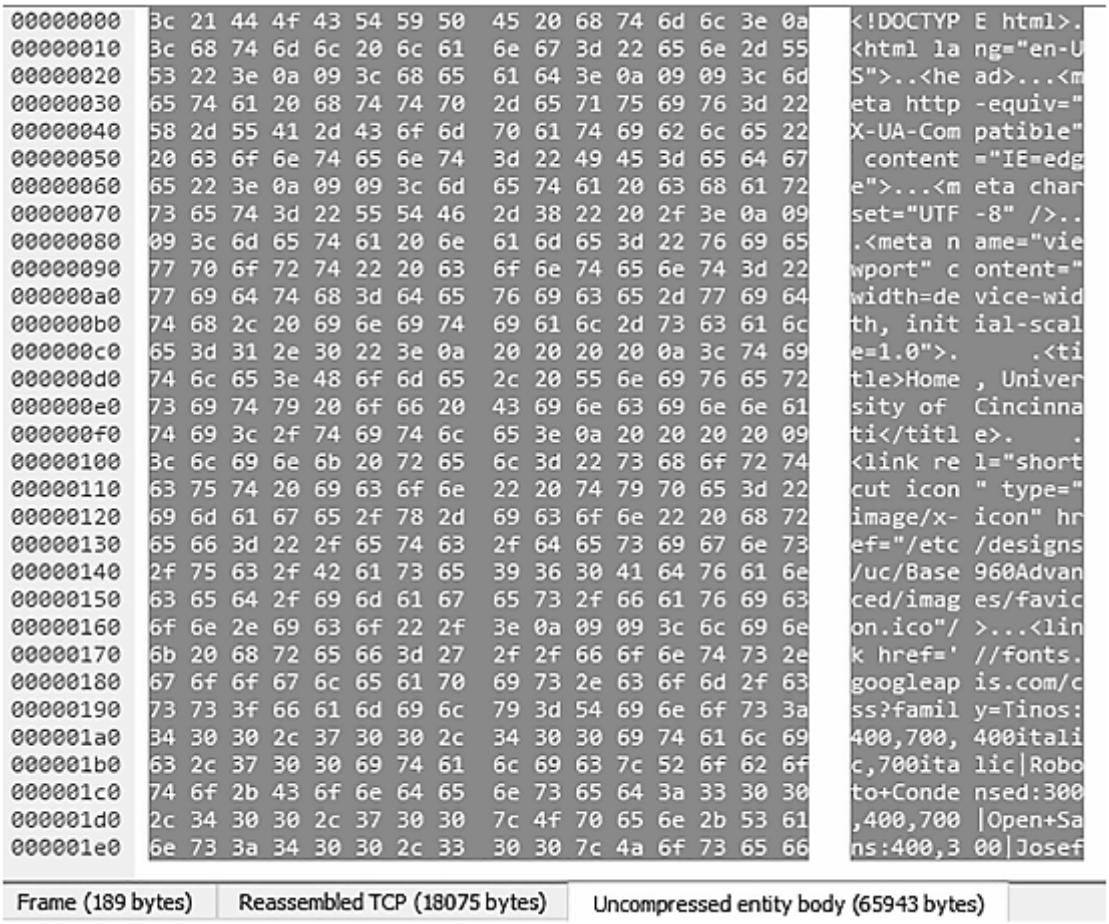
Sekarang, mari kita pertimbangkan paket pertama yang dikembalikan oleh www.uc.edu. Ini adalah paket 461 pada Gambar 6.9. Gambar 6.11 memberikan kutipan singkat dari bagian data dari respon ini. Anda harus memperhatikan bahwa padanan ASCII terdiri dari beberapa kode HTML, termasuk, misalnya, tag meta, tag judul, dan tautan untuk memuat gambar seperti file favicon.ico (gambar kecil yang muncul di bilah judul web). peramban). Beberapa item yang ditautkan tidak ditangani oleh HTTP tetapi oleh TCP, sehingga item ini tidak akan muncul di Gambar 6.9 (yang, seperti yang mungkin Anda ingat, adalah daftar paket HTTP yang difilter). Mungkin juga akan ada beberapa permintaan dan tanggapan DNS jika halaman web merujuk ke objek di server lain dengan alias IP, bukan alamat IP.

Gambar 6.10 Paket permintaan HTTP.

Sekarang kita telah menjelajahi urutan paket tertentu, mari kita lihat secara singkat apa lagi yang bisa dilakukan Wireshark untuk kita. Di sisi kiri Gambar 6.12, kita melihat menu Statistik Wireshark. Melalui alat ini, kami dapat memperoleh informasi berguna tentang paket yang dipilih atau semua paket yang diambil. Bagian tengah Gambar 6.12 menunjukkan hasil pemilihan Properti File Tangkap, yang menyediakan ringkasan dari paket yang ditangkap sesi saat ini.

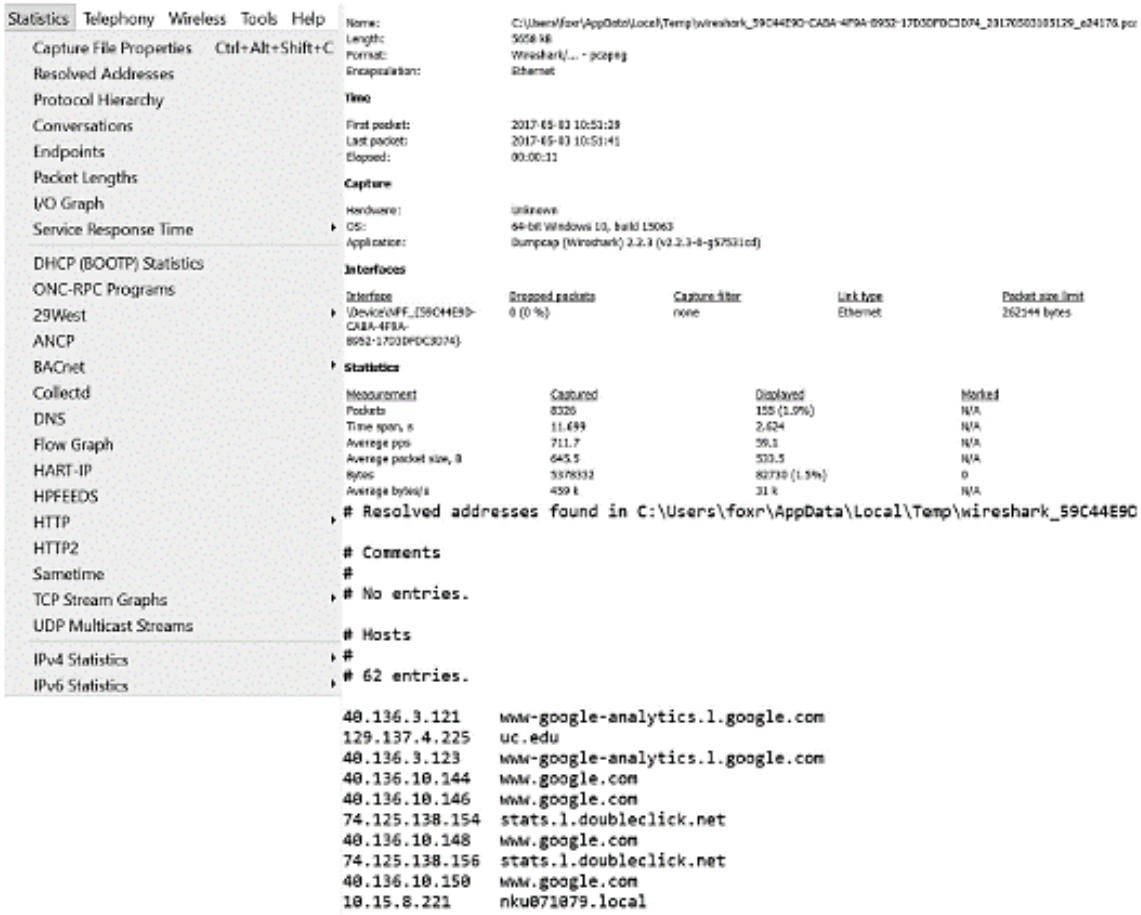
Ringkasan ini mencakup tanggal dan waktu paket pertama dan terakhir yang diambil selama sesi ini dan statistik paket tersebut. Dalam kasus ini, kita melihat bahwa 8326 paket telah ditangkap dan 155 paket sedang ditampilkan dari jendela tangkapan 2,624 detik. Kami melihat rata-rata ukuran paket, jumlah paket yang ditangkap per detik, dan jumlah byte per detik. Bagian bawah Gambar 6.12 menampilkan statistik Alamat Terselesaikan, yang

menunjukkan alias IP yang kami selesaikan selama sesi ini melalui DNS (di sini, kami telah mengutip keluaran ini untuk menunjukkan hanya beberapa alias IP teratasi).



Gambar 6.11 Porsi data paket TCP dari respons HTTP.

Ringkasan sebenarnya dari jumlah berbagai jenis paket tersedia dengan memilih Statistik IPv4 atau Statistik IPv6. Keduanya memiliki pilihan submenu tujuan dan pelabuhan. Kami melihat sebagian daftar dari sesi kami di Gambar 6.13. Misalnya, tujuan 64.215.193.195 memiliki total 433 paket, semuanya adalah paket TCP melalui port 443. Tujuan 40.136.3.117 memiliki lima paket TCP, semuanya melalui port 80. Sesi terdiri dari 3888 paket, jadi jelas, kami hanya melihat sebagian kecil dari output ini pada Gambar 6.13. Ada banyak fitur lain yang perlu dijelajahi dengan Wireshark yang kami hilangkan di sini karena ruang.



Gambar 6.12 Menu statistik Wireshark dan output ringkasan.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Destinations and Ports	3888				0.9687	100%	7.1300	2.603
74.125.196.95	7				0.0017	0.18%	0.0500	1.514
TCP	7				0.0017	100.00%	0.0500	1.514
80	7				0.0017	100.00%	0.0500	1.514
64.233.177.95	6				0.0015	0.15%	0.0600	1.579
TCP	6				0.0015	100.00%	0.0600	1.579
80	6				0.0015	100.00%	0.0600	1.579
64.215.193.195	433				0.1079	11.14%	0.6100	0.273
TCP	433				0.1079	100.00%	0.6100	0.273
443	433				0.1079	100.00%	0.6100	0.273
52.84.0.146	12				0.0030	0.31%	0.1000	1.641
TCP	12				0.0030	100.00%	0.1000	1.641
443	12				0.0030	100.00%	0.1000	1.641
40.136.3.117	5				0.0012	0.13%	0.0500	2.180
TCP	5				0.0012	100.00%	0.0500	2.180
80	5				0.0012	100.00%	0.0500	2.180
255.255.255.255	5				0.0012	0.13%	0.0200	2.413
UDP	5				0.0012	100.00%	0.0200	2.413
7765	1				0.0002	20.00%	0.0100	3.650
68	2				0.0005	40.00%	0.0200	2.413
17500	2				0.0005	40.00%	0.0100	0.721
23.111.9.31	2				0.0005	0.05%	0.0200	1.514
TCP	2				0.0005	100.00%	0.0200	1.514
80	2				0.0005	100.00%	0.0200	1.514

Gambar 6.13 Resolusi alamat selama sesi ini.

tcpdump

Program tcpdump adalah program standar di sistem Linux dan Unix. Tujuannya adalah untuk melaporkan lalu lintas jaringan. tcpdump hanya akan berjalan di bawah root. Di bagian ini, kami mengeksplorasi cara menggunakan tcpdump dan informasi apa yang dapat Anda peroleh darinya.

Secara default, program akan berjalan hingga terputus (misalnya, dengan mematikan program dengan mengetik ctrl+c), mengirimkan semua hasilnya ke jendela terminal. Output menampilkan 96 byte pertama dari setiap paket yang diambil. Anda dapat mengarahkan output ke file jika diinginkan dengan menambahkan –w filename. Anda masih akan keluar dari tcpdump dengan mengetik ctrl+c. Hasil pengalihan bukanlah file teks; namun, untuk membaca file, gunakan tcpdump –r filename. Anda juga dapat membaca file keluaran tcp-dump melalui Wireshark.

Seperti halnya Wireshark, Anda dapat membatasi keluaran yang dilaporkan oleh tcpdump dengan menyertakan filter. Filter harus berupa ekspresi Boolean yang menguji setiap paket dengan beberapa kriteria. Jika melewati kriteria, paket ditampilkan. Sintaks untuk menentukan ekspresi Boolean adalah melalui pemanggilan fungsi pcap-filter. Sintaks hukum adalah untuk menentukan satu atau lebih primitif diikuti dengan nilai-nilai tertentu. Misalnya, Anda dapat memfilter berdasarkan alamat IP atau alias, menggunakan host, src, atau dst. Dengan host dst, tcpdump hanya menampilkan pesan di mana alamat IP tujuan (atau alias) adalah host. Dengan host src, tcpdump hanya menampilkan pesan yang ditujukan untuk host alamat IP. Untuk menunjukkan bahwa perangkat sumber bernama host, gunakan host host. Jadi, misalnya, Anda dapat menentukan yang berikut ini:

```
tcpdump dst 10.11.12.13
tcpdump src 10.11.12.13
tcpdump host 10.11.12.13
```

Anda juga dapat menggunakan dan (atau &&) atau atau (||) untuk menentukan src dan/atau dst. Kriteria yang lebih rumit dapat ditentukan dengan menggunakan dan atau atau bersama dengan tanda kurung, di mana tanda kurung harus diawali dengan \. Berikut ini akan cocok dengan paket apa pun yang nama hostnya mencakup nku atau uc dan edu.

```
tcpdump host edu and \( nku or uc \)
```

Anda juga dapat menggunakan tidak atau!. Ini akan meniadakan kriteria, sehingga kondisi salah menyebabkan filter berlaku.

Anda juga dapat menentukan alamat jaringan untuk setiap lalu lintas pesan yang dikirim melalui jaringan yang diberikan. Seperti halnya alamat jaringan apa pun, Anda hanya menentukan bagian jaringan, diikuti oleh /n, di mana n adalah jumlah bit untuk menunjukkan jaringan. Misalnya, jika alamat IP Anda adalah 10.11.12.13 dan netmask Anda adalah 255.255.248.0, maka alamat jaringan Anda adalah 10.11.8.0/21 (21 bit pertama dari alamat IP menjadi alamat jaringan, dan 11 bit sisanya menjadi up alamat host di jaringan). Untuk membatasi tcpdump ke pesan yang dikirim melalui jaringan ini, Anda akan menggunakan tcpdump net 10.11.8.0/21. Kriteria penyaringan lainnya ditunjukkan pada Tabel 6.1.

Selain kriteria pemfilteran, ada beberapa opsi lain yang tersedia. Tabel 6.2 membahas beberapa opsi yang lebih bermanfaat. Anda dapat melihat semua opsi melalui halaman manual tcpdump.

Secara default, output dari tcpdump dibatasi hingga 96 byte per paket. Dengan cara ini, hasilnya kurang informatif dibandingkan dengan Wireshark, di mana Anda dapat menjelajahi detail tambahan dari setiap paket yang ditangkap. Namun, dengan menggunakan berbagai opsi, Anda dapat membuat laporan tcpdump lebih detail, seperti through –s atau salah satu mode verbose. Dengan –A atau –X, Anda dapat melihat data aktual paket.

Tabel 6.1 Filter Lain untuk tcpdump

Jenis Filter dan Argumen	Arti
jumlah yang lebih besar	Memfilter paket yang ukurannya lebih besar dari nilai yang ditentukan. Anda juga dapat menggunakan >. >= dapat digunakan untuk lebih dari atau sama dengan.
nomor kurang	Memfilter paket yang ukurannya kurang dari nilai yang ditentukan. Anda juga dapat menggunakan <. <= dapat digunakan untuk kurang dari atau sama dengan. Misalnya, tcpdump less 1024 tcpdump < 1024
nomor pelabuhan	Hanya menampilkan pesan yang dikirim atau diterima melalui nomor port yang diberikan. Anda juga dapat menambahkan dst atau src untuk menunjukkan nomor port tujuan atau sumber, seperti pada tcpdump dst port 8080.
portrange angka1–angka2	Untuk melihat paket dalam rentang port, seperti pada 21-23.
protokol proto	Hanya menampilkan pesan dari protokol yang diberikan, salah satunya tcp, udp, dan icmp. Anda tidak menyertakan kata proto dalam pernyataan tcpdump, seperti dalam tcpdump icmp.

Tabel 6.2 Opsi tcpdump yang signifikan

Pilihan	Arti
–c number	Berhenti setelah paket nomor ditangkap.
–e	Juga menampilkan header Ethernet.
–G number	Digunakan dengan –w untuk memutar ke file keluaran baru setiap detik. Nama file ditambahkan dengan format waktu (jika tidak ada format waktu yang ditentukan, setiap file yang berurutan akan menimpa file sebelumnya). Jika Anda menambahkan –C, itu menambahkan penghitung di akhir nama file.
–i interface	Hanya menangkap paket yang dikirim atau diterima oleh perangkat antarmuka yang ditentukan; kata any dapat digunakan untuk menunjukkan semua antarmuka. Tanpa opsi ini, tcpdump hanya mendengarkan pada antarmuka bernomor terendah (tidak termasuk loopback/localhost).
–n	Tidak menyelesaikan nama (hanya menampilkan alamat IP).
–q	Mode cepat, menampilkan lebih sedikit informasi protokol.
–s number	Menampilkan jumlah byte dari paket daripada ukuran default (96).

–v, –vv, and –vvv	Tiga mode verbose yang berbeda (–vvv adalah yang paling verbose).
–X	Menampilkan teks ASCII dan heksadesimal untuk konten pesan paket. Perhatikan bahwa –A menampilkan teks ASCII dari konten pesan.

4.2 NETCAT

Netcat, atau nc, program secara substansial berbeda dari program penangkapan atau analisis paket, sebagai gantinya menyediakan sarana untuk mengirim komunikasi langsung melalui port jaringan. Dengan nc, Anda dapat menguji koneksi jaringan dan koneksi ke perangkat jaringan, mengirim pesan langsung ke server dan melihat responsnya, atau membuat instruksi baris perintah (atau skrip) yang berfungsi seperti berbagai bentuk program jaringan. Perlu dicatat bahwa versi nc yang ditingkatkan tersedia, disebut ncat. Program nc telah dimodifikasi, sehingga banyak fitur yang ditemukan di ncat juga tersedia sekarang di nc, tetapi ncat juga menyediakan perantara koneksi, pengalihan, koneksi *Secure Sockets Layer* (SSL), dan kemampuan untuk menggunakan proxy. Karena nc dan ncat adalah alat yang menantang, kami membatasi pemeriksaan kami pada nc dan hanya beberapa penggunaan yang lebih mendasar.

Untuk menggunakan nc, pertama, Anda harus menginstalnya (mungkin asli atau tidak asli dari sistem operasi yang Anda instal). Untuk Windows, program ini disebut sebagai netcat dan tersedia dari sejumlah sumber. Di Unix/Linux, pastikan nc belum diinstal dengan menggunakan which nc atau which netcat. Cara termudah untuk menginstalnya adalah dengan menggunakan yum di Red Hat, seperti di yum install nc, atau apt-get di Debian, seperti di apt-get install netcat. Program netcat juga tersedia dalam kode sumber, tetapi tidak perlu menginstalnya dari kode sumber, karena diragukan bahwa Anda perlu melakukan perubahan apa pun.

Dengan nc tersedia, Anda menggunakannya untuk membuka koneksi jaringan. Koneksi memerlukan alamat IP (atau nama host) dan nomor port untuk digunakan. Misalnya, jika Anda ingin mengirim pesan ke port 80 dari server web www.somesite.org, gunakan nc www.somesite.org 80. Saat menekan <enter>, Anda akan dimasukkan ke buffer nc. Tidak ada prompt, jadi sepertinya nc telah hang. Sekarang, Anda memasukkan pesan. Pesan dikirim ke port tujuan yang ditentukan dari host, dan tanggapan, jika ada, ditampilkan di jendela terminal. Biasanya, koneksi ke host tetap terbuka, sehingga Anda dapat terus mengirim pesan sampai Anda mengakhiri koneksi (menggunakan ctrl + d atau ctrl + c) atau koneksi diakhiri oleh host (misalnya, jika koneksi habis).

Kami akan memulai eksplorasi nc kami dengan membuat koneksi ke server web. Dalam hal ini, mari kita asumsikan bahwa kita ingin berkomunikasi dengan www.somesite.org (server fiktif) melalui port HTTP standar, 80. Kita membuat koneksi dengan menggunakan perintah nc dari paragraf sebelumnya. Dengan koneksi terjalin, nc sedang menunggu Anda untuk memasukkan pesan. Pesan, dalam hal ini, akan menjadi permintaan HTTP. Pesan tersebut terdiri dari metode HTTP, nama file yang diminta, dan protokol khusus yang akan digunakan. Nama file akan memerlukan jalur lengkap, mulai dari DocumentRoot server (jika Anda tidak terbiasa dengan DocumentRoot, kami akan menjelajahinya saat melihat server web Apache di Bab 8). Protokolnya mungkin HTTP/1.1. Pesan HTTP diharapkan diakhiri dengan dua jeda

baris, jadi saat menyelesaikan permintaan kami, kami menekan <enter> dua kali. Pesan dikirim ke www.somesite.org, dan tanggapan apa pun yang dikembalikan kemudian ditampilkan.

Ada beberapa metode HTTP yang berbeda. Dua yang paling umum adalah GET, untuk mengambil halaman web, dan HEAD, untuk mendapatkan header file yang dihasilkan dari server web, tetapi bukan file itu sendiri. Metode lainnya adalah PILIHAN, untuk menerima metode yang tersedia yang diterapkan untuk server web ini, TRACE, untuk menerima dari server web salinan permintaan yang telah Anda kirim, dan POST dan PUT untuk mengunggah konten ke server web. Banyak server melarang POST dan PUT, karena dapat membuat server tidak aman.

Mari kita lihat contoh spesifik. Situs web www.time.gov menampilkan waktu saat ini, berdasarkan zona waktu Anda. Anda dapat mengirimkan permintaan HTTP melalui browser web Anda. Di sini, kita melihat bagaimana melakukan ini di nc. Pertama, buat koneksi. Setelah dibuat, kirim permintaan HTTP.

```
$ nc www.time.gov 80
GET / HTTP/1.1 <enter><enter>
```

Server web menerima permintaan HTTP sederhana ini dan, jika mungkin, mematuhi respons HTTP. Respons akan terdiri dari header respons HTTP dan halaman web. Di bawah ini, kita melihat tajuk respons. Daripada menampilkan konten halaman web (yang panjangnya beberapa halaman), kami menampilkan halaman web itu sendiri, seperti yang Anda lihat di browser Anda, pada Gambar 6.14.

```
HTTP/1.1 200 OK
Date: Wed, 18 Feb 2015 18:03:11 GMT
Server: Apache
Last-Modified: Wed, 31 Dec 2014 17:09:41 GMT
ETag: "704062-2888-50b862c575f40"
Accept-Ranges: bytes
Content-Length: 10376
Vary: Accept-Encoding
NIST: g4
Connection: close
Content-Type: text/html
```

Gambar 6.14 Halaman web yang sesuai dengan pesan nc kami.

Kita akan menjelajahi HTTP dan header yang menyusun permintaan HTTP dan respons HTTP secara lebih rinci di Bab 7. Sekarang mari kita fokus pada beberapa header yang ditemukan dalam respons di atas. Baris pertama dalam respons menunjukkan versi HTTP (1.1) dan kode status permintaan (200 berarti berhasil). Item seperti Date, Server, Last-Modified, Content-Length, dan Content-Type harus cukup jelas. Header ETag menunjukkan pengidentifikasi untuk file ini di cache lokal. Sambungan menunjukkan bahwa sambungan telah ditutup saat mengirim tanggapan ini.

Meskipun berguna untuk melihat tajuk respons, nc memungkinkan kita melakukan lebih banyak. Kami dapat menyediakan tajuk kami sendiri dalam permintaan HTTP. Di antara header yang umum adalah Content-Language dan Content-Type untuk menegosiasikan versi file mana yang akan dikirim (jika file tersedia dalam beberapa bahasa dan tipe). Mari kita perhatikan satu lagi, penggunaan Range. Dengan Range, kita dapat menentukan byte mana dari halaman yang akan dikirim. Kami mengubah permintaan kami sebagai berikut:

```
$ nc www.time.gov 80
GET / HTTP/1.1 <enter>
Range: bytes=0-100 <enter><enter>
```

Perhatikan bahwa kita menekan <enter> hanya sekali setelah baris pertama permintaan karena kita memiliki baris kedua permintaan untuk ditentukan. Kami menekan <enter> dua kali setelah baris kedua untuk menunjukkan bahwa permintaan HTTP selesai. Respons dari server sama seperti sebelumnya, hanya saja sekarang konten halaman web dibatasi hingga 101 byte pertama. Header juga sedikit berubah karena Content- Length sekarang 101 bukannya 10376.

Kami juga dapat berkomunikasi dengan server web dengan menyalurkan permintaan HTTP khusus ke nc daripada menggunakan buffer. Dengan cara ini, kita tidak harus berinteraksi langsung dengan antarmuka nc melainkan menggunakan skrip shell. Kami akan menggunakan perintah gema Unix/Linux untuk menampilkan permintaan HTTP kami dan kemudian menyalurkan hasilnya ke nc. Karena permintaan HTTP diakhiri dengan dua penekanan tombol <enter>, kita harus mensimulasikannya dalam gema. Untuk melakukannya, kami menggunakan \r\n\r\n dan menambahkan opsi –en ke perintah echo untuk mengaktifkan penggunaan \, sekaligus menonaktifkan keluaran baris baru. Perintah kami sebelumnya dengan header Range sekarang terlihat seperti berikut:

```
echo -en "HEAD / HTTP/1.1\r\n\Range: bytes=0-100\r\n\r\n"
    | nc www.time.gov 80
```

Ini adalah contoh yang cukup sepele. Sebagai gantinya, kami mungkin menggunakan nc dengan cara ini untuk menjelajahi beberapa halaman web untuk melihat apakah semuanya masih tersedia. Ingatlah bahwa baris pertama dari respons menyertakan kode status. Angka 200 menunjukkan halaman yang berhasil. Apa pun selain 200 menunjukkan kesalahan. Bayangkan kita memiliki file yang berisi nama dan URL server web. Kami ingin menentukan URL yang masih valid dan yang tidak. Sebuah skrip dapat menangani ini untuk kita. Inti dari skrip adalah gema kami | instruksi nc. Hasilnya dapat disalurkan ke program grep untuk mendapatkan hanya baris pertama dari keluaran (kami menggunakan HTTP untuk ekspresi reguler kami), dan kemudian, hasilnya dapat dikurangi dengan menyalurkan hasil tersebut ke

perintah awk '{print $2}' ke dapatkan item baris kedua saja. Ini hanya sesuai dengan kode status. Instruksi kami sekarang terlihat seperti berikut:

```
echo -en "HEAD $File HTTP/1.1\r\n\r\n" | nc $server 80 |
    grep HTTP | awk '{print $2}'
```

Ingatlah bahwa ini ada dalam skrip. $File adalah variabel yang menyimpan nama file saat ini, dan $server adalah variabel yang menyimpan alamat IP atau alias server. Sekarang setelah kita memiliki kode status, kita dapat mengeluarkan nama file, nama server, dan kode status ke file keluaran. Ini skrip lengkapnya, ditulis dalam bahasa skrip Bash. Kami menjalankan skrip shell ini, mengalihkan file nama file/nama server ke skrip, seperti pada ./url_checker < list.txt > url_checker_results.txt.

```
#!/bin/bash
while read file server; do
      status=`echo -en "HEAD $File HTTP/1.1\r\n\r\n" |
      nc $server 80 | grep HTTP | awk '{print $2}'`
      echo $file $server $status
done
```

Mari kita lihat menggunakan nc untuk mendengarkan port. Luncurkan nc dengan menggunakan port nc –l, seperti pada nc –l 80. Sekarang jendela terminal menggunakan nc untuk mendengarkan port TCP/IP yang diberikan, koneksi apa pun yang dibuat ke port tersebut akan membuang pesan masuk ke jendela terminal tersebut hingga koneksi terputus.

Sebagai percobaan, buka dua jendela terminal di Unix/Linux. Di jendela pertama, su untuk root dan ketik nc –l 80. Di jendela lain, ketik nc 127.0.0.1 80 dan tekan <enter>. Jendela terminal pertama diberitahu untuk mendengarkan port 80. Jendela terminal kedua telah membuka koneksi ke host lokal Anda melalui port 80. Sekarang, apa pun yang Anda ketik di jendela terminal kedua akan muncul di jendela pertama. Selain itu, perhatikan bahwa meskipun jendela terminal pertama bertindak sebagai pendengar port, ia bekerja dua arah. Jika Anda mengetik sesuatu di jendela terminal pertama, itu akan muncul di jendela terminal kedua. Anda akan keluar dari ini dengan menghentikan input menggunakan ctrl+d atau menghentikan program nc menggunakan ctrl+c dari salah satu jendela. Jika Anda ingin melakukan komunikasi semacam itu di seluruh jaringan ke perangkat kedua, gunakan nama host atau alias IP perangkat alih-alih 127.0.0.1. Perhatikan bahwa percobaan ini tidak akan bekerja jika firewall Anda memblokir paket masuk melalui port 80, dan Anda mungkin mendapatkan hasil yang aneh jika Anda menjalankan server web yang menangani permintaan melalui port 80.

Dengan mengalihkan output dari pendengar port kami ke file, kami dapat menggunakan nc untuk menyalin file di jaringan. Dalam hal ini, pada perangkat penerima, kami menggunakan instruksi nc –l port > nama file, di mana port adalah port yang ingin Anda dengarkan dan nama file akan menjadi file yang disimpan. Pada mesin pengirim, gunakan nc host port < filename. File dari komputer asli dikirim ke host melalui port port. Sambungan secara otomatis diakhiri dari kedua ujungnya ketika perintah nc kedua selesai.

Kita dapat menggunakan nc dengan cara yang lebih berani daripada pengalihan file sederhana melalui jaringan. Dengan sebuah pipa, kami dapat mengirimkan segala jenis informasi dari satu mesin ke mesin lainnya. Misalnya, kami mungkin ingin mengirim laporan

tentang siapa yang saat ini masuk ke komputer 1 ke komputer 2. Di komputer 2, dengarkan port yang diketahui oleh pengguna komputer 1 yang sedang Anda dengarkan. Selanjutnya, di komputer satu, kirimkan perintah who ke perintah nc. Dengan asumsi komputer ini diberi nama computer1.someorg.net dan computer2.someorg.net dan kita menggunakan port 301, kita dapat menggunakan perintah berikut:

```
Computer 2: nc -l 301
Computer 1: who | nc computer2.someorg.net 301
```

Kita dapat menggunakan nc sebagai alat untuk memeriksa port yang tersedia di perangkat jarak jauh. Untuk menentukan rentang port, gunakan nc –z host first-last, di mana first dan last adalah nomor port. Instruksi nc –z www. someserver.org 0-1023 akan menguji port server 0 hingga 1023. Perintah akan menampilkan daftar koneksi yang berhasil. Untuk melihat koneksi yang berhasil dan tidak berhasil, tambahkan –v (verbose). Output mungkin terlihat seperti berikut:

```
nc: connect to www.someserver.org port 0 (tcp) failed: Connection refused
nc: connect to www.someserver.org port 1 (tcp) failed: Connection refused
nc: connect to www.someserver.org port 2 (tcp) failed: Connection refused
...
Connection to www.someserver.org 22 port [tcp/ssh] succeeded!
...
```

Perhatikan bahwa perintah ini dapat menjadi masalah keamanan untuk server karena kami melihat port mana dari server yang dapat melewati firewall server. Jangan kaget jika server tidak menanggapi pertanyaan seperti itu.

Untuk penggunaan netcat tambahan, Anda mungkin ingin menjelajahi beberapa situs web yang didedikasikan untuk alat tersebut, termasuk situs resmi netcat di http://nc110.sourceforge.net/.

6.3 PROGRAM JARINGAN LINUX/UNIX

Di bagian ini, kami fokus pada program Linux dan Unix yang mungkin Anda gunakan untuk menjelajahi konektivitas jaringan Anda. Jika Anda bukan pemula di jaringan Linux/Unix, kemungkinan besar Anda telah melihat banyak atau sebagian besar perintah ini.

Pertama, kami mempertimbangkan layanan jaringan. Layanan ini harus dimulai secara otomatis oleh sistem operasi. Merupakan tanggung jawab layanan ini untuk memberikan alamat IP ke setiap antarmuka jaringan Anda. Nama layanan ini berbeda menurut distribusinya. Di Red Hat, itu disebut jaringan. Di Debian, itu disebut jaringan. Di Slackware/SLS/SuSE, ini disebut rc.inet1. Jika layanan telah berhenti, Anda dapat memulainya (atau memulai ulang) dengan mengeluarkan perintah yang tepat seperti systemctrl start network. service atau systemctrl restart network.service di Red Hat 7 atau /sbin/service network start atau /sbin/service network restart di Red Hat 6. Saat memulai/memulai ulang, layanan jaringan mengubah file konfigurasi jaringan. Di Red Hat 6, ada file untuk setiap antarmuka yang disimpan di bawah /etc/sysconfig/network-scripts dan diberi nama ifcfg-dev, di mana dev adalah nama perangkat seperti eth0 untuk perangkat Ethernet atau lo untuk perangkat loopback (host lokal).). Anda juga dapat secara langsung mengontrol antarmuka individual Anda melalui serangkaian skrip. Misalnya, ifup-eth0 dan ifdown-eth0 dapat

digunakan untuk memulai dan menghentikan antarmuka eth0. Alternatifnya, Anda dapat memulai atau menghentikan antarmuka, perangkat, dengan menggunakan perangkat ifup dan perangkat ifdown, seperti pada ifup eth0.

Perintah IP linux/unix

Anda dapat memperoleh alamat IP antarmuka Anda atau mengubah alamat tersebut melalui perintah ip. Perintah ip dapat digunakan lebih dari berurusan dengan alamat IP, jadi kami akan kembali lagi nanti di bagian ini. Kami juga secara singkat menyebutkan ipconfig, sebuah program Windows.

Untuk mendapatkan alamat IP antarmuka, gunakan ip addr atau ip a. Ini menampilkan semua alamat antarmuka. Untuk membatasi ip agar hanya menampilkan alamat IPv4 atau IPv6 tetapi tidak keduanya, tambahkan -4 atau -6 antara ip dan addr. Untuk menentukan alamat antarmuka tunggal, gunakan ip addr show name, seperti pada ip addr show eth0 atau ip addr show lo. Anda juga dapat memeriksa alamat IP dari antarmuka yang saat ini diaktifkan dengan menggunakan ip link show up atau antarmuka tertentu dengan menggunakan ip link show dev name, di mana name adalah nama antarmuka yang diberikan.

Untuk menyetel alamat IP, gunakan ip addr add address dev name, di mana nama adalah nama antarmuka dan alamat adalah alamat yang disediakan untuk antarmuka tersebut. Jika antarmuka sudah memiliki alamat, perintah ini memberi antarmuka alamat tambahan. Anda dapat menambahkan netmask bersamaan dengan alamat IP dengan menggunakan notasi IP_address/netmask untuk alamat, seperti pada 10.11.12.13/255.255.248.0. Atau, Anda dapat menentukan jumlah bit di netmask yang relevan dengan menggunakan IP_address/prefix_number, seperti pada 10.11.12.13/21 untuk contoh di atas (21 bit pertama di 255.255.248.0 adalah 1 bit).

Perintah ip addr add juga dapat digunakan untuk menambahkan informasi lain ke alamat IP yang sudah ada. Menambahkan antarmuka dev alamat brd memungkinkan Anda menetapkan alamat sebagai alamat siaran untuk antarmuka. Kata broadcast dapat digunakan untuk menggantikan brd. Demikian pula, Anda dapat menentukan alamat siaran mana pun dengan menggunakan alamat siaran mana pun, alamat untuk titik akhir koneksi titik-ke-titik menggunakan alamat peer, atau label yang dapat ditambahkan ke alamat sehingga alamat tersebut dapat dirujuk. Disebabkan oleh label ini daripada oleh alamat tertentu, menggunakan alamat label. Contoh berikut menetapkan alamat untuk kartu Ethernet kami bersama dengan netmask, alamat broadcast, dan label.

```
ip addr add 10.11.12.13/21 brd 10.11.12.0 dev eth0
     label ethernet
```

Anda dapat dengan mudah menghapus alamat antarmuka yang sudah ditetapkan dengan mengganti add dengan del. Anda harus menyertakan alamat yang Anda hapus jika antarmuka memiliki banyak alamat. Terakhir, dengan ip, Anda juga dapat menaikkan atau menurunkan antarmuka. Perintahnya adalah ip link atur nama dev ke atas/bawah. Di Windows, program untuk melihat alamat IP disebut ipconfig. Ini jauh lebih sederhana daripada ip (dan lebih sederhana daripada ifconfig yang tercakup dalam bacaan online), karena memiliki opsi yang jauh lebih sedikit dan fungsionalitas yang jauh lebih sedikit. Perintah ipconfig dengan sendirinya akan menampilkan semua interface. Untuk setiap antarmuka, Anda akan melihat

nama domain yang ditetapkan ke perangkat, alamat IPv4, alamat IPv6 (jika tersedia), netmask, dan gateway. Informasi tambahan dapat disajikan seperti adaptor terowongan apa pun. Menggunakan parameter /all memberikan detail lebih lanjut, termasuk deskripsi perangkat, nama host yang ditetapkan ke antarmuka (atau komputer yang menaungi antarmuka), dan apakah DHCP dan konfigurasi otomatis diaktifkan, dan jika demikian, informasi tentang alamat IP sewaan, alamat server DNS, dan alamat MAC untuk perangkat. Program ipconfig menerima parameter /renew untuk memperbarui alamat IP dinamis yang kedaluwarsa untuk semua antarmuka (atau /renew nama untuk memperbarui hanya alamat IP untuk antarmuka bernama nama) dan/nama rilis atau nama /release6 untuk merilis IPv4 atau IPv6 sewaan alamat. Parameter /displaydns menampilkan konten cache DNS lokal Anda (jika ada), /flushdns membersihkan cache DNS lokal, dan /registerdns mencoba menyegarkan semua sewa saat ini pada alamat IP dinamis. Perhatikan bahwa Anda tidak dapat menetapkan alamat IP baru secara langsung melalui ipconfig, tidak seperti ip dan ifconfig. Anda dapat mempelajari lebih lanjut tentang ipconfig dengan menggunakan parameter /h.

Mac OS X dibangun di atas sistem operasi mirip Unix (Mach) dan begitu juga dengan ifconfig. Itu juga memiliki ipconfig, memungkinkan pengguna untuk menggunakan salah satu perintah.

Sumber daya jaringan lainnya

Administrator jaringan harus memastikan bahwa jaringan tidak hanya dapat diakses tetapi juga berjalan secara efisien dan aman. Semua sistem operasi menyediakan segudang alat untuk menjelajahi jaringan. Di bagian ini, kami fokus pada beberapa program populer. Semua program ini tersedia di Linux/Unix, dan beberapa program tersedia dalam format yang sama atau serupa di Windows.

Pertama, mari kita lihat program untuk menguji konektivitas jaringan dan ketersediaan sumber daya jaringan. Dua program yang paling umum untuk penggunaan ini adalah ping dan traceroute (tracert di Windows). Kedua program ini menggunakan Internet Control Message Protocol (ICMP). Sesuai namanya, protokol ini digunakan agar perangkat dapat mengeluarkan pesan kontrol. Pesan permintaan dapat mencakup mengulangi permintaan, mendapatkan informasi status dari node jaringan (seperti apakah node tujuan atau port dari node tujuan tertentu dapat dijangkau dan apakah jaringan dapat dijangkau), meminta pengalihan pesan, dan menanyakan untuk informasi lainnya. Meskipun kami menyebutnya sebagai kontrol, banyak dari jenis permintaan ini digunakan untuk diagnosis.

Program ping menggunakan ICMP untuk mengirim permintaan gema, dengan hasil menerima paket respons dari sumber daya tujuan. Kemudian, ping menampilkan informasi yang diperoleh dari paket respons sebagai ringkasan ukuran paket dalam byte, alamat IP sumber (alamat perangkat yang di-ping), waktu hidup paket (ttl), dan waktu dibutuhkan agar paket dapat diterima di seluruh jaringan. Kecuali ditentukan lain, ping berjalan hingga dimatikan (misalnya, dengan mengetik ctrl+c). Anda dapat membatasi jumlah paket yang dikirim dan ditampilkan dengan menggunakan –c count, seperti pada ping –c 10. Pada akhirnya, ping juga menampilkan ringkasan jumlah paket yang dikirimkan, jumlah yang diterima, persentase yang hilang, dan total waktu untuk yang ping berlari. Ini diikuti oleh waktu pulang pergi minimum, waktu pulang pergi rata-rata, waktu pulang pergi maksimum,

dan standar deviasi (dikenal sebagai mdev). Semua waktu diberikan dalam milidetik. Kami melihat contoh sesi dengan ping:

```
$ ping www.google.com
PING www.google.com (74.125.196.105) 56(84) bytes of data.
64 bytes from 74.125.196.106: icmp_seq1 ttl=44 time=38.5 ms
64 bytes from 74.125.196.106: icmp_seq2 ttl=44 time=38.3 ms
64 bytes from 74.125.196.106: icmp_seq3 ttl=44 time=38.3 ms
64 bytes from 74.125.196.106: icmp_seq4 ttl=44 time=38.1 ms
64 bytes from 74.125.196.106: icmp_seq5 ttl=44 time=38.2 ms
^C
--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4459ms
rtt min/avg/max/mdev = 38.188/38.332/38.584/0.138 ms
```

Selain –c, ping memiliki opsi lain yang mungkin layak digunakan, tergantung pada keadaan. Tabel 6.3 memberikan deskripsi beberapa opsi yang lebih relevan di Linux.

Ping versi Windows mirip dengan versi Unix/Linux; namun, keluarannya sedikit berbeda dan pilihannya berbeda. Misalnya, di Windows, ping hanya mengeluarkan sejumlah paket dan berhenti secara otomatis, kecuali diberikan opsi –t. Selain itu, opsi penghitungan adalah –n angka daripada –c. Meskipun ping berguna untuk menetapkan jika beberapa titik akhir dapat dijangkau, traceroute berguna untuk menampilkan rute yang diambil ke titik akhir tersebut. Dengan cara ini, Anda tidak hanya dapat menentukan apakah suatu tujuan dapat dijangkau, tetapi juga bagaimana tujuan tersebut dapat dijangkau. Jalur yang di hasilkan oleh traceroute juga dapat memberi tahu Anda jika jalur tertentu ke rute tersebut tidak tersedia (jika jalur tersebut dicoba). Melalui traceroute, Anda juga dapat menentukan seberapa efisien jaringan Anda dapat mencapai titik akhir tertentu. Output dari traceroute adalah satu baris per percobaan hop. Setiap hop adalah jalur yang ditempuh oleh paket antara perangkat lokal dan tujuan.

Tabel 6.3 Opsi Ping di Linux

Pilihan	Arti
–A	Interval paket yang ditransmisikan disesuaikan berdasarkan waktu yang dibutuhkan untuk paket respons pertama tiba.
–b	Ping perangkat siaran.
–f	Mengeluarkan periode sebagai pengganti informasi respons biasa (periode dikeluarkan untuk setiap transmisi dan dihapus untuk setiap tanda terima).
–i number	Setel interval antara ping ke angka detik; standarnya adalah 1 detik. Hanya root yang dapat mengeluarkan interval kurang dari 0,2 detik.
–l number	Dalam mode ini, ping mengirimkan paket nomor sekaligus. Hanya root yang dapat mengeluarkan angka 4 atau lebih besar.
–q	Mode diam, hanya menghasilkan ringkasan akhir. Ini berguna jika Anda menggunakan ping dalam skrip dan hanya perlu mendapatkan informasi ringkasan.
–R	Menampilkan rute (hingga 9 rute). Opsi ini sering diabaikan oleh perangkat di Internet sebagai potensi masalah keamanan.
–t number	Mengatur waktu hidup untuk paket-paket ini ke nomor.
–w number	Minta ping keluar setelah angka detik.

Cara kerja traceroute adalah mengirimkan banyak paket, yang disebut probe. Untuk satu lompatan, tiga probe dikirim. Probe diinisialisasi dengan dua TTL: TTL pertama dan TTL maksimum. TTL pertama diatur ke 1 secara default. Nilai ini menunjukkan jumlah lompatan pertama yang diizinkan untuk mencapai router atau gateway dari perangkat sumber. Jika jumlah hop yang dibutuhkan lebih sedikit dari TTL pertama, maka probe akan dijatuhkan. Ini mungkin terjadi, misalnya, jika TTL pertama diatur ke 2 dan probe mencapai gateway perantara dalam satu hop daripada gateway jaringan. Namun, probe biasanya tidak boleh dijatuhkan karena alasan ini karena defaultnya adalah 1, artinya hanya probe yang mencapai router/gateway dalam waktu kurang dari satu lompatan yang dijatuhkan. TTL maksimum default ke 30 tetapi dapat diubah oleh pengguna. Nilai ini berkurang saat probe mencapai router berikutnya dalam perjalanannya. Jadi, TTL maksimum sebenarnya menentukan jumlah lompatan maksimum yang diizinkan. Setelah nilai ini mencapai 0, probe dijatuhkan daripada diteruskan.

Jika probe dijatuhkan, * akan ditampilkan di rute. Jika ketiga probe dijatuhkan, lompatan itu menunjukkan bahwa router berikutnya tidak dapat dijangkau. Output untuk kasus seperti itu adalah * * *. Jika tersedia router lain, rute alternatif dapat dicoba. Jika 5 detik berlalu tanpa lompatan yang berhasil, traceroute berakhir, menunjukkan bahwa tidak ada jalur yang berhasil dari sumber ke tujuan.

Pada setiap hop yang berhasil, router (atau tujuan) mengembalikan pesan ICMP echo reply untuk menunjukkan bahwa probe berhasil mencapai lokasi jaringan saat ini. Informasi ini adalah keluaran dan mencakup waktu yang diperlukan untuk mencapai titik ini dari sumbernya (ini adalah total kumulatif). Kemudian, output dari traceroute adalah urutan hop (lokasi yang dicapai) dan waktu yang dibutuhkan masing-masing probe untuk mencapai lokasi tersebut. Contoh keluaran traceroute dari komputer di jaringan area lokal ke www.google.com ditunjukkan pada Gambar 6.15. Perhatikan bahwa lompatan 11 sampai 12, 11 sampai 13, dan 14 sampai 15 tidak berhasil. Selain itu, baris 1 menunjukkan jalur yang diambil dari perangkat sumber ke titik kontak subnetnya (router atau gateway). Jumlah total lompatan adalah 13, dengan total waktu sedikit lebih dari 38 ms untuk setiap probe.

Program traceroute memiliki sejumlah pilihan; namun, Anda jarang membutuhkannya. Secara default, traceroute menggunakan UDP untuk probe, karena UDP tidak memerlukan pengakuan. Namun, -Saya akan memaksa traceroute untuk mengirim permintaan gema ICMP untuk probe dan –T akan menggunakan probe TCP dan menggunakan flag SYN untuk menunjukkan tanda terima. Anda dapat membatasi traceroute untuk hanya menggunakan IPv4 atau IPv6 dengan -4 dan -6, masing-masing. Anda dapat mengubah TTL pertama dari 1 ke nilai lain dengan menggunakan nilai –f, memodifikasi TTL maksimum dari 30 ke beberapa nilai dengan menggunakan nilai –m, dan memodifikasi waktu tunggu default untuk setiap probe dari 5 detik ke beberapa nilai lain dengan menggunakan nilai –w.

```
traceroute to www.google.com (74.125.196.147), 30 hops max, 60 byte packets
 1  10.2.56.1 (10.2.56.1)  0.435 ms  0.420 ms  0.426 ms
 2  10.2.254.5 (10.2.254.5)  0.767 ms  0.725 ms  0.691 ms
 3  172.31.100.1 (172.31.100.1)  0.724 ms  0.703 ms  0.746 ms
 4  st03-cr6-03.nku.edu (192.122.237.10)  2.166 ms  4.052 ms  4.024 ms
 5  10.1.250.6 (10.1.250.6)  3.982 ms  2.026 ms  3.566 ms
 6  10.1.250.9 (10.1.250.9)  3.856 ms  3.004 ms  3.207 ms
 7  173.191.112.169 (173.191.112.169)  8.407 ms  9.342 ms  9.321 ms
 8  216.249.136.157 (216.249.136.157)  9.251 ms  9.168 ms  8.910 ms
 9  64.57.21.109 (64.57.21.109)  37.618 ms  37.161 ms  36.683 ms
10  162.252.69.135 (162.252.69.135)  23.323 ms  23.315 ms  23.731 ms
11  209.85.143.154 (209.85.143.154)  44.460 ms  34.323 ms  23.642 ms
12  * * *
13  * * *
14  209.85.248.31 (209.85.248.31)  38.351 ms  38.5 ms  38.553 ms
15  * * *
16  74.125.196.147 (74.125.196.147)  38.482 ms  38.218 ms  38.439 ms
```

Gambar 6.15 keluaran Traceroute.

Perhatikan bahwa traceroute, secara default, menggunakan UDP untuk probenya. Namun, respon dari tujuan adalah paket ICMP, yang melaporkan rute lengkap dan waktu yang dibutuhkan. ICMP dianggap sebagai protokol yang berbahaya karena dapat digunakan untuk mendapatkan informasi tentang cara kerja internal suatu jaringan. Misalnya, pada Gambar 6.15, kita tidak hanya melihat alamat tujuan (yang mudah didapat, seperti yang akan kita jelajahi di Bagian 4.4) tetapi juga alamat lain yang mungkin internal Google. Menggunakan ping atau traceroute untuk mendapatkan alamat IP internal jaringan dikenal sebagai serangan pengintaian.

Mengapa serangan pengintaian menjadi masalah? Jika seseorang dapat memperoleh alamat IP internal, orang tersebut berpotensi meluncurkan berbagai bentuk serangan terhadap node jaringan tersebut. ICMP dapat digunakan untuk meluncurkan bentuk serangan lain yang dikenal sebagai serangan denial of service (DOS). Dalam serangan seperti itu, server benar-benar dibanjiri dengan terlalu banyak permintaan untuk ditangani. Hasilnya adalah server sangat sibuk dalam menangani permintaan palsu sehingga layanan ditolak untuk pengguna server yang sah. Meskipun serangan DOS dapat dibuat melalui sejumlah pendekatan, termasuk mengeksploitasi kelemahan TCP, HTTP, dan langsung melalui aplikasi, ICMP memiliki kelemahan tersendiri yang dapat dimanfaatkan. Ini dikenal sebagai serangan Smurf, juga disebut banjir ICMP, banjir ping, atau banjir kematian. Pertimbangkan pendekatan ini. Sebuah paket dikirim ke alamat broadcast jaringan. Paket tersebut kemudian disiarkan dari node yang diberikan ke semua node dalam jaringan. Node lain ini biasanya akan merespons dengan mengirimkan paket pengembalian mereka sendiri kembali ke sumbernya. Jika alamat IP sumber dapat dipalsukan, yang cukup mudah, maka siaran awal dikalikan dengan semua tanggapan, yang kemudian ditanggapi, yang menyebabkan lebih banyak tanggapan dihasilkan, berulang-ulang, mengisi seluruh jaringan dengan ICMP yang tidak berguna pesan.

Tidak jarang administrator jaringan melindungi jaringan dari serangan ICMP dengan salah satu dari dua cara. Pertama, semua perangkat di jaringan dapat dikonfigurasi untuk mengabaikan permintaan ICMP. Dengan cara ini, setiap paket ICMP yang diterima dapat dijatuhkan secara otomatis (misalnya, melalui firewall). Sayangnya, solusi ini menghilangkan kemampuan untuk menggunakan ping dan traceroute dan dengan demikian menghilangkan

alat yang sangat berguna. Solusi kedua adalah tidak meneruskan paket yang diarahkan ke alamat broadcast. Meskipun ini mencegah banjir ICMP, itu tidak melarang percobaan pengintaian.

Alat jaringan lain yang berguna adalah perintah netstat. Program ini menyediakan statistik penggunaan jaringan. Tujuan defaultnya adalah untuk menampilkan koneksi paket TCP yang masuk dan keluar berdasarkan nomor port. Itu juga dapat digunakan untuk melihat tabel perutean dan statistik yang berkaitan dengan antarmuka jaringan. Gambar 6.16 memberikan kutipan singkat dari keluaran netstat.

Perintah netstat, tanpa opsi, menyediakan daftar semua soket yang aktif. Pada gambar, kita melihat daftar ini dibagi menjadi dua bagian: koneksi internet dan soket domain UNIX.

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address   Foreign Address         State
tcp        1      0 10.2.56.44:55720    97.65.93.72:http        CLOSE_WAIT
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ]         DGRAM                    8709   @/org/kernel/udev/udevd
unix  2      [ ]         DGRAM                    12039  @/org/freedesktop/...
unix  9      [ ]         DGRAM                    27129  /dev/log
unix  2      [ ]         DGRAM                    338413
unix  2      [ ]         DGRAM                    337005
unix  2      [ ]         DGRAM                    336667
unix  2      [ ]         STREAM     CONNECTED     320674 /var/run/dbus/...
unix  2      [ ]         STREAM     CONNECTED     320673
unix  3      [ ]         STREAM     CONNECTED     313454 @/tmp/dbus-UVQUx4c4BV
unix  3      [ ]         STREAM     CONNECTED     313453
unix  2      [ ]         DGRAM                    313238
unix  3      [ ]         STREAM     CONNECTED     313206 /var/run/dbus/...
```

Gambar 6.16 Keluaran dari netstat.

Perbedaannya di sini adalah bahwa soket domain adalah mekanisme komunikasi antarproses yang bertindak seolah-olah itu adalah koneksi jaringan, tetapi komunikasi tersebut mungkin sepenuhnya lokal ke komputer. Pada Gambar 6.16, kita hanya melihat satu soket yang digunakan untuk komunikasi jaringan. Dalam hal ini, ini adalah pesan TCP yang diterima melalui port 55720 dari alamat IP 97.65.93.72 melalui port http (80). Status soket ini adalah CLOSED_WAIT.

Untuk soket domain, netstat memberi kami tujuh bidang informasi. Proto adalah protokolnya, dan unix menunjukkan soket domain standar. RefCnt menampilkan jumlah proses yang terpasang pada soket ini. Bendera menunjukkan bendera apa pun, kosongkan dalam hal ini karena tidak ada bendera yang ditetapkan. Jenis menunjukkan jenis akses soket. DGRAM adalah datagram tanpa koneksi, dan STREAM adalah koneksi. Jenis lainnya antara lain RAW (soket mentah), RDM (soket pesan terkirim dengan andal), dan UNKNOWN. Status akan kosong jika soket tidak terhubung, FREE jika soket tidak dialokasikan, MENDENGARKAN jika soket mendengarkan permintaan koneksi, CONNECTING jika soket akan membuat koneksi, MEMUTUSKAN jika soket terputus dari pesan, CONNECTED jika soket aktif digunakan, dan ESTABLISHED jika koneksi telah dibuat untuk suatu sesi. I-Node adalah nomor inode yang diberikan ke soket (di Unix/Linux, soket diperlakukan sebagai file yang diakses melalui struktur

data inode). Terakhir, Path adalah nama path yang terkait dengan proses. Misalnya, dua entri pada Gambar 6.16 berasal dari proses /var/run/dbus/system_bus_socket.

Opsi netstat –a menunjukkan semua soket, apakah sedang digunakan atau tidak. Menggunakan opsi –a akan memberi kita lebih banyak koneksi Internet. Opsi –t menyediakan koneksi port tcp aktif, dan –at menampilkan semua port yang mendengarkan pesan TCP. Demikian pula, -u menyediakan koneksi port udp aktif, dan –au menampilkan semua port yang mendengarkan pesan UDP.

Opsi –p ke netstat menambahkan PID dari nama proses. Opsi –i menyediakan ringkasan untuk antarmuka. Variasi netstat –ie merespons dengan informasi yang sama seperti ifconfig. Opsi –g menyediakan daftar grup multicast, yang dicantumkan berdasarkan antarmuka. Opsi –c memaksa netstat untuk berjalan terus menerus, memperbarui keluarannya setiap detik, atau memperbarui keluarannya berdasarkan interval yang disediakan, seperti –c 5 untuk setiap 5 detik.

Dengan –s, netstat merespons dengan ringkasan semua statistik jaringan. Ringkasan ini mencakup TCP, UDP, ICMP, dan pesan IP lainnya. Anda dapat membatasi ringkasan untuk jenis protokol tertentu dengan menyertakan huruf setelah –s seperti –st untuk TCP dan –su untuk UDP. Tabel 6.4 memberikan gambaran tentang jenis informasi yang diberikan dalam ringkasan ini.

Tabel 6.4 netstat –s Output

Protokol	Jenis Informasi yang Diberikan
ICMP	Pesan diterima, input gagal, tujuan tidak dapat dijangkau, batas waktu, permintaan gema, dan balasan gema. Pesan dikirim, gagal, tujuan tidak dapat dijangkau, permintaan gema, dan balasan gema.
IP	Paket diterima, diteruskan, dibuang, alamat tidak valid, dan dikirimkan. Permintaan dikirim dan dijatuhkan karena rute yang hilang.
TCP	Koneksi aktif, koneksi lewat, upaya koneksi gagal, reset koneksi, koneksi dibuat, dan reset dikirim. Segmen diterima, segmen ditransmisikan, segmen ditransmisikan ulang, dan segmen buruk diterima.
UDP	Paket diterima, ke port yang tidak diketahui, kesalahan, dan paket dikirim.

Menggunakan netstat –r memberikan output yang sama persis dengan rute perintah. Program route digunakan untuk menampilkan satu atau lebih tabel routing atau untuk memanipulasi satu atau lebih tabel routing. Perintah route telah digantikan oleh ip route. Contoh tabel routing adalah sebagai berikut:

```
Destination   Gateway     Genmask         Flags   Metric   Ref   Use   Iface
10.2.56.0     *           255.255.248.0   U       1        0     0     eth0
Default       10.2.56.1   0.0.0.0         UG      0        0     0     eth0
```

Tabel ini memberi tahu kita bahwa gateway untuk perangkat di jaringan 10.2.56.0 adalah 10.2.56.1 dan netmask untuk jaringan ini adalah 255.255.248.0. Bendera U menunjukkan bahwa perangkat aktif (aktif), dan G menunjukkan bahwa perangkat tersebut adalah gateway. Metrik menunjukkan jarak (lompatan) antara perangkat di jaringan dan gateway. Jadi, kami melihat bahwa semua perangkat di jaringan ini berjarak satu lompatan dari gateway untuk jaringan ini. Bidang Ref tidak digunakan (di Linux), dan Iface menunjukkan nama perangkat antarmuka tempat paket yang dikirim dari router ini akan diterima. Perintah rute memiliki berbagai opsi, tetapi kami tidak akan mempertimbangkannya di sini, karena kami akan melihat perintah ip berikutnya.

Perhatikan bahwa perintah netstat telah digantikan. Dalam hal ini, fungsinya ditangkap dalam program bernama ss, yang digunakan untuk menampilkan statistik soket. Kami mengakhiri bagian ini dengan mengunjungi kembali ip. Anda akan ingat bahwa ip adalah program yang dibuat untuk menggantikan ifconfig. Itu juga telah menggantikan rute. Untuk mendapatkan informasi yang sama dengan rute, gunakan daftar rute ip. Sebagai tanggapan, Anda mungkin melihat keluaran seperti berikut:

```
10.2.56.0/21 dev eth0 proto kernel scope link src 10.2.56.45 metric 1
default via 10.2.56.1 dev eth0 proto static
```

Walaupun outputnya berbeda dengan route, kita bisa melihat informasi yang sama. Misalnya, output menunjukkan alamat jaringan, 10.2.56.0/21, alamat IP gateway, dan metrik. Selain itu, ditunjukkan di sini bahwa alamat IP gateway adalah alamat statis. Dengan rute ip, Anda dapat mengubah tabel perutean. Pilihannya adalah menambahkan rute baru, menghapus rute, mengubah informasi tentang rute, atau mengosongkan tabel perutean. Anda dapat menambahkan rute jika ada jalur kedua dari komputer ini ke jaringan, misalnya dengan memiliki gateway kedua. Anda akan menghapus rute jika Anda menghapus jalur, misalnya, dari dua gerbang menjadi satu. Anda dapat mengubah rute jika informasi tentang gateway seperti prototipe, jenis perangkat, atau metriknya berubah. Perintah-perintah ini adalah sebagai berikut:

```
ip route add node [additional information]
ip route delete node
ip route change node via node2 [additional information]
ip route replace node [additional information]
```

Node nilai akan menjadi alamat/nomor IP, di mana angka tersebut adalah angka 1 di netmask, misalnya, 21 pada contoh sebelumnya. Dalam kasus perubahan, node2 akan menjadi alamat IP tetapi tanpa nomor. Informasi tambahan dapat berupa antarmuka perangkat tertentu menggunakan notasi yang kita lihat di bagian terakhir: antarmuka dev seperti dev eth0 atau protokol seperti pada proto statis atau kernel proto. Karena addr dapat disingkat sebagai a, ip juga memungkinkan singkatan untuk rute sebagai ro atau r, dan tambah, ubah, ganti, dan hapus masing-masing dapat disingkat sebagai a, chg, repl dan del.

Penggunaan lain dari ip adalah untuk memanipulasi cache Address Resolution Protocol (ARP) kernel. Protokol ini digunakan untuk menerjemahkan alamat dari lapisan jaringan TCP/IP menjadi alamat pada lapisan tautan atau untuk menerjemahkan alamat IP menjadi alamat perangkat keras fisik (MAC). Perintah ip neigh (untuk tetangga) digunakan untuk

menampilkan alamat perangkat keras antarmuka dari tetangga (gateway) ke komputer ini. Perintah ini telah menggantikan perintah arp yang lebih lama.

Misalnya, arp akan merespons dengan keluaran seperti berikut:

```
Address      HWtype    HWaddresses          Flags mask    Iface
10.2.56.1    ether     00:1d:71:f4:b0:00    C             eth0
```

Gateway komputer ini, 10.2.56.1, menggunakan kartu Ethernet bernama eth0 dan memiliki alamat MAC yang diberikan. C untuk Bendera menunjukkan bahwa entri ini saat ini ada di cache ARP komputer. M akan menunjukkan entri permanen, dan P akan menunjukkan entri yang diterbitkan. Menggunakan ip, perintahnya adalah ip neigh show, yang menghasilkan keluaran berikut:

```
10.2.56.1 dev eth0 lladdr 00:1d:71:f4:b0:00 STALE
```

Perbedaan utama dalam keluaran adalah penyertaan kata STALE untuk menunjukkan bahwa entri cache mungkin sudah usang. Simbol lladdr menunjukkan alamat perangkat keras. Kami dapat memperoleh nilai baru dengan terlebih dahulu menghapus entri ini dari cache ARP kami dan kemudian mengakses gateway kami lagi. Perintah flush adalah ip neigh flush dev eth0. Perhatikan bahwa ini hanya menghapus cache ARP yang terkait dengan antarmuka eth0 kami, tetapi sepertinya hanya itu antarmuka kami.

Dengan cache ARP kami sekarang kosong, kami tidak memiliki alamat MAC untuk gateway kami, sehingga kami dapat menggunakan ARP untuk mendapatkan alamat IP-nya. Lalu, bagaimana kita bisa berkomunikasi dengan jaringan? Kita harus mendapatkan alamat MAC gateway kita. Ini ditangani dengan menyiarkan permintaan di subnet lokal. Permintaan diterima oleh switch jaringan atau router dan diteruskan ke gateway jika perangkat ini bukan gateway kami. Gateway merespons, dan sekarang, cache kami dilengkapi dengan alamat MAC (yang kemungkinan besar sama dengan yang kami miliki sebelumnya). Namun, dalam kasus ini, ketika kami mengeluarkan kembali perintah ip neigh show, kami melihat bahwa gateway dapat dijangkau, bukan STALE.

```
10.2.56.1 dev eth0 lladdr 00:1d:71:f4:b0:00 REACHABLE
```

Seperti halnya ip route, kita dapat menambah, menghapus, mengubah, atau mengganti tetangga. Perintahnya serupa karena kami menentukan alamat/nomor IP baru, dengan opsi untuk menambahkan antarmuka perangkat melalui dev. Satu penggunaan terakhir dari ip adalah menambah, mengubah, menghapus, atau menampilkan terowongan apa pun. Terowongan dalam jaringan adalah penggunaan satu protokol dari dalam yang lain. Dalam hal Internet, kami menggunakan TCP untuk membuat koneksi antara dua perangkat. Sayangnya, TCP mungkin tidak berisi fitur yang kami inginkan, misalnya enkripsi. Jadi, di dalam koneksi TCP yang dibuat, kami membuat terowongan yang menggunakan protokol lain. Protokol kedua, atau tunneled, dapat menjadi salah satu yang memiliki fitur yang tidak ada pada IP.

Perintah ip menyediakan kemampuan berikut untuk memanipulasi terowongan:

```
ip tunnel add name [additional information]
ip tunnel change name [additional information]
ip tunnel del name
ip tunnel show name
```

Nama adalah nama yang diberikan terowongan. Informasi tambahan dapat mencakup mode untuk mengenkapsulasi pesan dalam terowongan (salah satu dari ipip, sit, isatap, gre, ip6ip6, dan ipip6, atau apa pun), satu atau dua alamat (lokal dan jarak jauh), nilai waktu untuk hidup, antarmuka untuk mengikat terowongan, apakah paket harus diserialkan atau tidak, dan apakah akan menggunakan checksum.

Kami sebenarnya baru saja menggores permukaan program ip. Ada banyak kegunaan lain dari ip, dan ada banyak pilihan yang tidak dijelaskan dalam buku pelajaran ini. Meskipun halaman utama akan menunjukkan kepada Anda semua opsi, yang terbaik adalah membaca tutorial lengkap tentang ip jika Anda ingin memanfaatkannya secara maksimal.

Program Logging

File log dibuat secara otomatis oleh sebagian besar sistem operasi untuk merekam peristiwa yang bermanfaat. Di Linux/ Unix, ada beberapa log berbeda yang dibuat dan program tersedia untuk melihat file log. Di sini, kami fokus pada file log dan mencatat pesan yang terkait dengan situasi jaringan. Ada beberapa file log berbeda yang mungkin berisi informasi tentang peristiwa jaringan. File /var/log/messages menyimpan semua jenis pesan informasi. Sebagai contoh, pembaruan file konfigurasi antarmuka (mis., ifcfg-eth0) akan dicatat.

File /var/log/secure berisi bentuk-bentuk pesan autentikasi (catat bahwa beberapa pesan autentikasi juga direkam di /var/log/messages). Beberapa entri log autentikasi akan berkaitan dengan komunikasi jaringan, misalnya, jika pengguna mencoba ssh, ftp, atau telnet ke komputer dan harus melalui proses login. Entri log berikut menunjukkan bahwa pengguna foxr telah mencoba dan kemudian berhasil melakukan ssh ke komputer ini (localhost).

```
Mar 4 11:22:56 localhost sshd[9428]: Accepted password for foxr from 10.2.56.45 port 34207 ssh2
Mar 4 11:22:56 localhost sshd[9428]: pam_unix(sshd:session): session opened for user foxr by (uid=0)
```

Hampir setiap operasi yang dilakukan Unix/Linux dicatat. Entri log ini ditempatkan di /var/log/audit dan dapat diakses menggunakan aureport dan ausearch. aureport menyediakan informasi ringkasan, sedangkan ausearch dapat digunakan untuk menemukan entri log tertentu berdasarkan berbagai kriteria. Sebagai contoh, jika Anda ingin melihat event yang terkait dengan perintah ssh, ketikkan ausearch –x sshd. Ini akan menampilkan entri log audit tempat sshd (daemon ssh) dipanggil. Setiap entri mungkin terlihat seperti berikut:

```
----
time->Fri Dec 18 08:18:36 2015
type=CRED_DISP msg=audit(1450444716.354:114280): user
pid=15798 uid=0 auid=500 ses=19042
subj=subject_u:system_r:sshd_t:s0-s0:c0.c1023
msg='op=destroy kind=server
fp=b5:88:47:0e:1a:65:89:3a:90:7e:0a:fd:34:7f:c6:44
direction=? spid=15798 suid=0 exe="/usr/sbin/sshd"
hostname=? addr=10.2.56.45 terminal=? res=success'
```

Entrinya terlihat sangat samar. Mari kita lihat lebih dekat. Pertama, tipe memberitahu kita tipe pesan; CRED_DISP, dalam hal ini, adalah pesan yang dihasilkan dari peristiwa autentikasi,

menggunakan mekanisme autentikasi bernama modul autentikasi pluggable (PAM). Selanjutnya, kita melihat ID entri audit. Kami akan menggunakan 114280 untuk menanyakan log audit untuk hal-hal spesifik seperti dengan mengeluarkan perintah ausearch –a 114280. Selanjutnya, kami melihat informasi tentang pengguna: ID proses, ID pengguna (pengguna tempat program ini dijalankan—root dalam kasus ini) , ID pengguna efektif (pengguna yang membuat permintaan—foxr dalam kasus ini), nomor sesi, dan data konteks SELinux. Selanjutnya, kita melihat pesan aktual yang dicatat oleh sshd, termasuk kunci enkripsi yang disediakan, sehingga pengguna dapat mengenkripsi pesan di bawah ssh. Item yang terkait dengan exe adalah program yang dapat dieksekusi yang diminta (sshd).

Menggunakan aureport, kami hanya dapat memperoleh ringkasan. Karena setiap permintaan ssh memerlukan autentikasi, kami dapat menggunakan aureport –au, yang mencantumkan peristiwa tersebut. Di sini, kita melihat sebagian daftar.

```
1. 08/25/2015 10:09:01 foxr ? :0 /sbin/unix_chkpwd yes 105813
2. 09/17/2015 12:44:13 zappaf ? pts/0 /bin/su yes 109207
3. 12/18/2015 08:18:36 2015 foxr 10.2.56.45 ssh /usr/sbin/sshd yes 114280
```

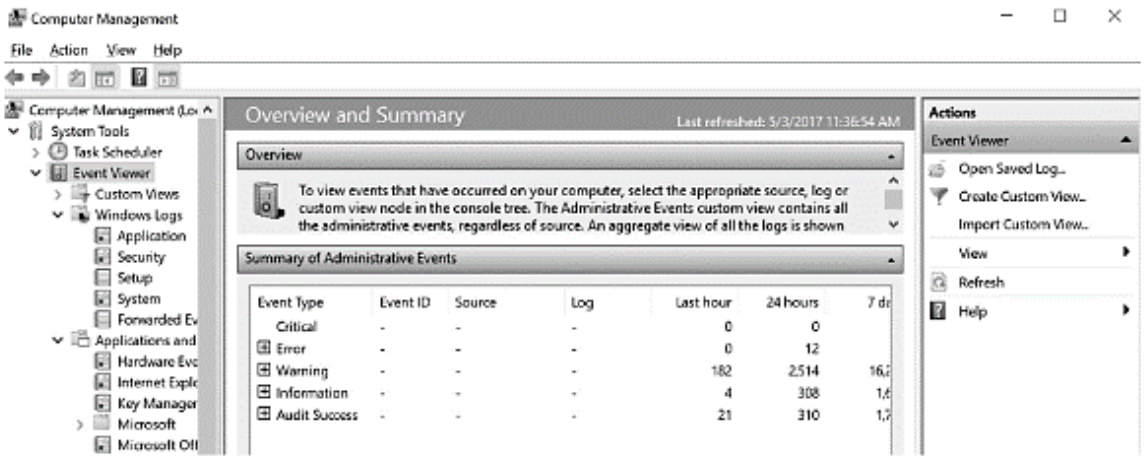
Dalam kutipan ini, ada tiga event autentikasi, hanya satu yang merupakan event ssh (yang terakhir, yang juga merupakan entri yang tercantum di atas dengan hasil ausearch).

Anda juga dapat melihat pesan log dari aktivitas yang terkait dengan layanan jaringan (memulai, menghentikan, dan mengonfigurasi ulang). Untuk melakukan ini, Anda mengatur mekanisme logging Anda sendiri melalui daemon syslog. Tidak ada aturan khusus yang dapat Anda berikan untuk mencari aktivitas layanan jaringan saja, tetapi Anda dapat menentukan aturan logging untuk semua layanan. Anda akan memodifikasi file konfigurasi syslog dengan memasukkan aturan berikut:

```
daemon.*        /var/log/daemons
```

Entri ini menyatakan bahwa setiap pesan yang dihasilkan dari layanan apa pun (daemon) harus dicatat ke dalam file /var/log/daemon. * menunjukkan tingkat pesan apa pun. Anda dapat menggantinya dengan level pesan yang lebih spesifik seperti warn (setiap peringatan atau pesan dengan prioritas lebih tinggi) atau info (informasi atau pesan dengan prioritas lebih tinggi). Lihat halaman manual syslog.conf untuk detail lebih lanjut. Perhatikan bahwa versi Linux terbaru telah mengganti nama layanan menjadi rsyslogd dan file konfigurasi menjadi /etc/rsyslog.conf.

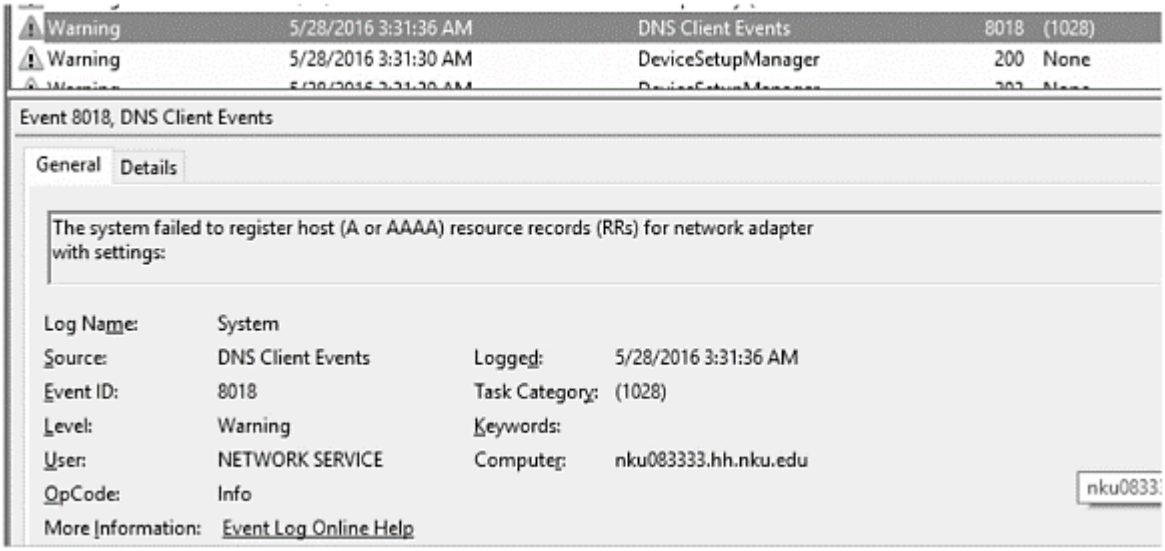
Windows memiliki mekanisme pencatatan yang serupa, dan Anda dapat memeriksa log ini melalui program Peraga Peristiwa. Jika Anda mengklik kanan ikon Komputer Anda dan memilih Manajemen, Anda akan diberi daftar alat manajemen. Memilih Peraga Peristiwa menampilkan tampilan, seperti yang ditunjukkan pada Gambar 6.17. Dari sini, Anda dapat memperluas tingkat peristiwa yang dicatat (kritis, kesalahan, peringatan, dan sebagainya). Pada gambar ini, di panel kiri, Anda dapat melihat bahwa dua subdaftar Log Windows dan Aplikasi dan Log Layanan telah diperluas. Anda dapat mencari, misalnya, Peristiwa Sistem atau Perangkat Keras untuk yang terkait dengan jaringan Anda.

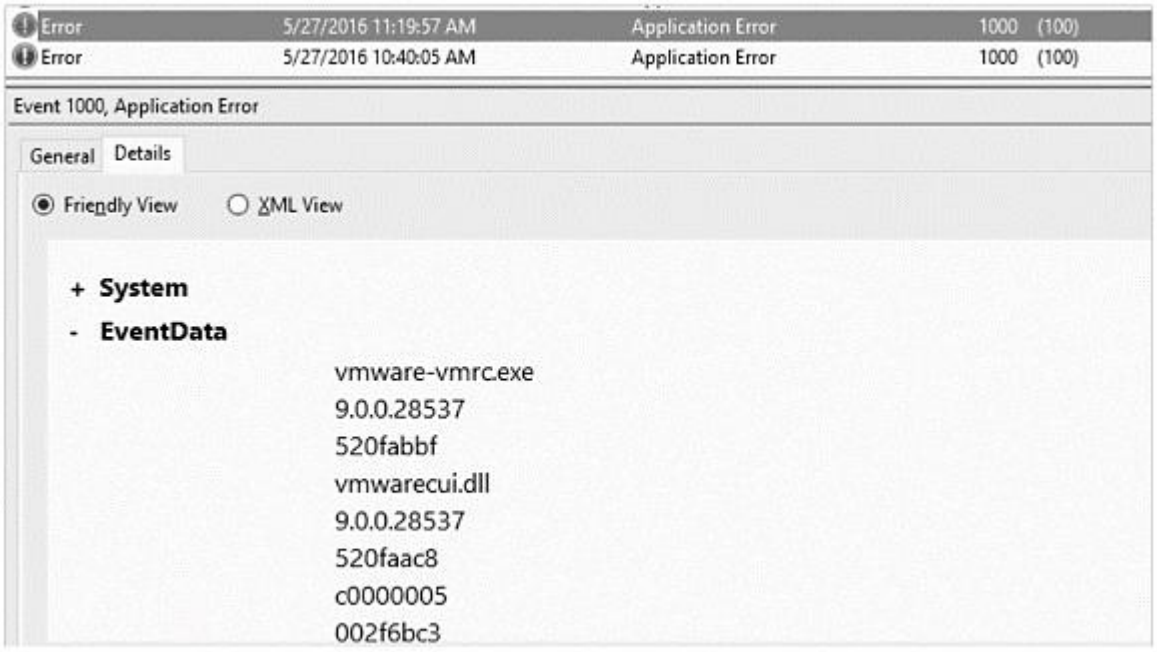


Gambar 6.17 penampil acara Windows.

Gambar 6.18 mengilustrasikan peringatan dan kesalahan yang ditemukan. Yang pertama, peringatan, adalah Peristiwa Klien DNS, di mana kita dapat melihat bahwa "Sistem gagal mendaftarkan catatan sumber daya host (A atau AAAA) (RR) untuk adaptor jaringan dengan pengaturan" (sisanya tidak terlihat tanpa menggulir). Kesalahan muncul dari menjalankan aplikasi vmware-vmrc.exe (bagian dari program startup VMware Client). Dalam hal ini, kami melihat beberapa detail data peristiwa.

Di sebelah kanan panel tengah Peraga Peristiwa terdapat daftar tindakan yang tersedia, termasuk fitur pencarian untuk mencari acara dengan kata kunci (pemilihan disebut Temukan...). Panel kanan ini tidak diperlihatkan pada Gambar 6.17. Di bawah ini adalah contoh entri jaringan yang ditemukan di bawah daftar Keamanan (di bawah Log Windows). Dalam hal ini, ini adalah acara login jaringan yang berhasil.





Gambar 6.18 Entri log untuk peringatan (DNS) dan kesalahan (VMWare).

```
Log Name:           Security
Source:                 Microsoft-Windows-Security-Auditing
Date:                   Fri 12 18 2015 4:31:11 AM
Event ID:               4648
Task Category:      Logon
Level:                  Information
Keywords:               Audit Success
User:                   N/A
Computer:               ************
Description:
      A logon was attempted using explicit credentials.

Subject:
Security ID:            SYSTEM
Account Name:           **************
Account Domain:     NKU
Logon ID:               0x3e7
Logon GUID:         {00000000-0000-0000-0000-000000000000}

Account Whose Credentials Were Used:
Account Name:           ************
Account Domain:     HH.NKU.EDU
Logon GUID:         {3e21bdee-71d8-74d5-79f7-0c00a0cbdeb4}

Target Server:
Target Server Name:     ************
Additional Information:     **************
Process Information:
Process ID:             0x4664
Process Name:           C:\Windows\System32\taskhost.exe

Network Information:
Network Address: -
Port:            -

This event is generated when a process attempts to log on an account by
explicitly specifying that account's credentials. This most commonly
occurs in batch-type configurations such as scheduled tasks, or when
using the RUNAS command.
```

6.4 PERINTAH SISTEM NAMA DOMAIN

Selanjutnya, kami memeriksa perintah yang terkait dengan DNS. DNS dibuat agar kami dapat berkomunikasi dengan perangkat di Internet dengan nama, bukan nomor. Semua perangkat memiliki alamat IPv4 32-bit, alamat IPv6 128-bit, atau keduanya. Mengingat rangkaian bit, bilangan bulat, atau bilangan heksadesimal yang panjang tidaklah mudah. Sebaliknya, kami lebih memilih nama yang mirip bahasa Inggris seperti www.google.com. Nama-nama ini dikenal sebagai alias IP. DNS memungkinkan kami untuk mereferensikan perangkat ini berdasarkan nama, bukan nomor.

Proses pemanfaatan DNS untuk mengkonversi dari alias IP ke alamat IP dikenal sebagai penyelesaian alamat, atau pencarian IP. Kami jarang harus melakukan resolusi alamat sendiri, karena aplikasi seperti browser web dan ping menangani ini untuk kami. Namun, ada empat program yang tersedia untuk kami gunakan jika kami perlu mendapatkan informasi ini. Program-program ini adalah nslookup, whois, dig, dan host.

Program nslookup sangat sederhana tetapi penggunaannya juga terbatas. Anda akan menggunakannya untuk mendapatkan alamat IP untuk alias IP seolah-olah Anda adalah perangkat lunak yang ingin menyelesaikan alias. Perintah memiliki bentuk nslookup alias [server] untuk mendapatkan alamat IP dari alias yang ditentukan, di mana server adalah opsional dan Anda hanya akan menentukannya jika Anda ingin menggunakan server nama DNS selain default untuk jaringan Anda. Perintah nslookup www.nku.edu akan merespons dengan alamat IP untuk www.nku.edu, sebagaimana ditentukan oleh server DNS Anda. Perintah nslookup www. nku.edu 8.8.8.8 akan menggunakan server nama di 8.8.8.8 (yang merupakan salah satu server nama publik Google). Keduanya harus merespons dengan alamat IP yang sama.

Di bawah ini, kita dapat melihat hasil dari kueri pertama kita. Respons dimulai dengan alamat IP server DNS yang digunakan (172.28.102.11 adalah salah satu server nama DNS NKU). Mengikuti entri ini adalah informasi tentang perangkat yang diminta (www.nku.edu). Jika alias IP yang Anda berikan bukan nama sebenarnya untuk perangkat tersebut, maka Anda juga akan diberikan nama sebenarnya (kanonis) perangkat tersebut. Ini diikuti dengan nama dan alamat perangkat.

```
Server: 172.28.102.11
Address: 172.28.102.11#53

www.nku.edu canonical name = hhilwb6005.hh.nku.edu.
Name: hhilwb6005.hh.nku.edu
Address: 172.28.119.82
```

Jika kami mengeluarkan perintah nslookup kedua di atas, menggunakan 8.8.8.8 untuk server, kami juga akan diberi tahu bahwa responsnya tidak otoritatif. Artinya, kami memperoleh respons bukan dari server nama domain tertentu (nku.edu) tetapi dari sumber lain. Kami akan membahas perbedaannya di Bab 5 saat kami memeriksa DNS secara menyeluruh.

Program nslookup memiliki mode interaktif, yang dapat Anda masuki jika Anda tidak memasukkan alias atau Anda menentukan tanda hubung (–) sebelum alias, seperti di nslookup – www.nku.edu. Mode interaktif memberi Anda > sebagai prompt. Setelah dalam mode interaktif, serangkaian perintah tersedia. Ini dijelaskan dalam Tabel 6.5. Whois bukanlah alat

seperti protokol. Pada dasarnya, ini melakukan hal yang sama seperti nslookup, kecuali sering diterapkan langsung di situs web, sehingga Anda dapat melakukan kueri nslookup dari situs web, bukan dari baris perintah. Situs web terhubung ke (atau berjalan langsung di) server whois. Server whois akan menyelesaikan alias IP yang diberikan menjadi alamat IP. Melalui beberapa server whois ini, Anda dapat memperoleh informasi situs web seperti subdomain dan data lalu lintas situs web, riwayat situs web, dan informasi data DNS (misalnya, data A, data MX, dan data NS) serta melakukan operasi ping dan traceroute. Beberapa situs web whois ini juga memungkinkan Anda menemukan nama yang mirip jika Anda ingin memberi nama domain dan, jika tersedia, membeli nama domain.

Program host dan dig memiliki fungsi yang serupa, jadi kami akan menekankan host dan kemudian menjelaskan cara menyelesaikan tugas serupa dengan menggunakan dig. Baik host maupun dig dapat digunakan seperti nslookup, tetapi keduanya juga dapat digunakan untuk mendapatkan informasi yang jauh lebih detail dari server nama. Perintah-perintah ini unik untuk Linux/Unix.

Tabel 6.5 Perintah Interaktif nslookup

Command	Parameter(s)	Penjelasan
Host	[alias] [server]	Perintah ini sama dengan nslookup alias [server], dimana jika tidak ada server yang ditentukan maka default name server yang digunakan.
Server	[domain]	Perintah ini memberikan informasi tentang server nama domain. Tanpa domain, domain default digunakan. Lihat lserver.
lserver	domain	Sama seperti server, kecuali mengubah server default ke server nama domain ini.
Set	keyword[=value]	Setel kata kunci (ke nilai jika ditentukan) untuk pencarian di masa mendatang. Misalnya, set class=CH akan mengubah nslookup untuk menggunakan kelas CH alih-alih kelas default IN. Kata kunci lainnya termasuk domain=domain_name port=port_number type=record_type

Perintah host membutuhkan setidaknya IP alias yang ingin Anda selesaikan. Secara opsional, Anda juga dapat menentukan server nama yang akan digunakan. Namun, kekuatan tuan rumah ada pada berbagai pilihan. Pertama, –c memungkinkan Anda menentukan kelas jaringan (IN untuk Internet, CH untuk kekacauan, atau HD untuk Hesiod). Opsi –t memungkinkan Anda menentukan jenis catatan sumber daya yang ingin Anda kueri dari server nama tentang alias IP yang diberikan. Catatan sumber daya ditentukan berdasarkan jenis, menggunakan singkatan seperti A untuk alamat IPv4, AAAA untuk alamat IPv6, MX untuk server email, NS untuk server nama, dan CNAME untuk nama kanonis. Dengan –W, Anda menentukan jumlah detik yang Anda paksa host untuk menunggu sebelum waktu habis.

Selain opsi di atas yang menggunakan parameter, ada sejumlah opsi yang tidak memiliki parameter. Opsi -4 dan -6 mengirim kueri dengan menggunakan IPv4 dan IPv6, masing-masing (jangan bingung dengan menggunakan –t A vs –t AAAA). Opsi –w memaksa tuan rumah untuk menunggu selamanya (berlawanan dengan menggunakan –W dengan waktu yang ditentukan). Opsi –d dan –v memberikan output verbose, yang berarti bahwa server nama merespons dengan seluruh record. Kami akan memeriksa beberapa contoh segera. Opsi –a mewakili kueri apa pun, yang memberikan detail lebih besar daripada kueri tertentu tetapi tidak sebanyak permintaan verbose. Kombinasi –a dan –l (mis., -al) memberikan kueri apa pun dengan opsi daftar yang menampilkan semua catatan dari server nama.

Opsi –i melakukan pencarian IP terbalik. Pencarian terbalik menggunakan server nama untuk mengembalikan alias IP untuk alamat IP tertentu. Jadi, ini adalah kebalikan dari penggunaan umum dari server nama. Meskipun tampaknya berlawanan dengan intuisi untuk menggunakan pencarian balik, ini berguna untuk tujuan keamanan, karena terlalu mudah untuk memalsukan alamat IP. Reverse lookup memberi server sarana untuk mendeteksi apakah alamat IP dan alias IP perangkat cocok.

Intinya, perintah dig melakukan hal yang sama dengan perintah host. Tidak seperti host, dig dapat digunakan baik dari baris perintah atau dengan melewatkan file yang berisi sejumlah operasi. Pendekatan terakhir ini membutuhkan opsi –f filename. Perintah dig memiliki banyak opsi yang sama dengan host: -4, -6, -t type, dan -c class, dan pencarian balik dilakukan dengan menggunakan –x (bukan –i). Sintaks untuk dig adalah nama dig [@server] [opsi], di mana server menimpa server nama default lokal dengan server yang ditentukan. Satu perbedaan antara dig dan host adalah bahwa dig selalu memberikan keluaran verbose sedangkan host hanya melakukannya berdasarkan permintaan (-d atau –v).

Di sini, kami memeriksa beberapa contoh dari host dan gali. Dalam setiap kasus, kami akan menanyakan www.google. com dengan menggunakan server nama lokal. Kami mulai dengan membandingkan host dan menggali di www.google.com tanpa opsi sama sekali. Perintah host mengembalikan yang berikut ini:

```
www.google.com has address 74.125.196.99
www.google.com has address 74.125.196.104
www.google.com has address 74.125.196.105
…
www.google.com has IPv6 address 2607:f8b0:4002:c07::67
```

Banyak tanggapan karena Google memiliki beberapa server alias www.google.com. Perintah dig mengembalikan respons yang lebih detail, seperti yang ditunjukkan di bawah ini:

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6_4.6 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13855
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.                IN     A

;; ANSWER SECTION:
www.google.com.         197    IN     A      74.125.196.105
www.google.com.         197    IN     A      74.125.196.147
```

```
www.google.com.     197    IN     A      74.125.196.106
www.google.com.     197    IN     A      74.125.196.99
www.google.com.     197    IN     A      74.125.196.103
www.google.com.     197    IN     A      74.125.196.104

;; Query time: 0 msec
;; SERVER: 172.28.102.11#53(172.28.102.11)
;; WHEN: Tue Mar 3 08:40:15 2015
;; MSG SIZE rcvd: 128
```

Ada empat bagian dalam respon dari dig. Bagian HEADER memberi kita informasi tentang kueri. Bagian PERTANYAAN mengulang pertanyaan, yaitu, apa yang ingin kami lihat. Dalam hal ini, kami ingin mengetahui alamat IP untuk www.google.com. Bagian JAWABAN adalah respons dari server nama Google, yang mencantumkan semua data A (perhatikan bahwa ini tidak menyertakan data AAAA, karena kami tidak meminta alamat IPv6). Akhirnya, kami menerima ringkasan komunikasi.

Selanjutnya, kami menggunakan host –d untuk mendapatkan respons yang mirip dengan dig. Sekali lagi, kita melihat bagian yang sama (HEADER, QUESTION, ANSWER, dan ringkasan), tetapi ini diikuti oleh dua keluaran lagi, masing-masing dengan bagian HEADER dan bagian PERTANYAAN dan bagian JAWABAN atau, dalam kasus yang terakhir bagian dari output, bagian AUTHORITY.

```
Trying "www.google.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29282
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.            IN     A

;; ANSWER SECTION:
www.google.com.     1      IN     A      74.125.196.99
www.google.com.     1      IN     A      74.125.196.104
www.google.com.     1      IN     A      74.125.196.105
www.google.com.     1      IN     A      74.125.196.103
www.google.com.     1      IN     A      74.125.196.147
www.google.com.     1      IN     A      74.125.196.106

Received 128 bytes from 172.28.102.11#53 in 0 ms
Trying "www.google.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46770
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.            IN     AAAA

;; ANSWER SECTION:
www.google.com.     267    IN     AAAA   2607:f8b0:4002:c07::69

Received 60 bytes from 172.28.102.11#53 in 0 ms
Trying "www.google.com"
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28338
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.            IN     MX
```

```
;; AUTHORITY SECTION:
google.com.           60    IN    SOA   ns1.google.com.
dns-admin.google.com. 87601031 7200 1800 1209600 300

Received 82 bytes from 172.28.102.11#53 in 35 ms
```

Mari kita lihat lebih hati-hati pada output di atas. Perhatikan di HEADER pertama bahwa bendera yang berbeda sedang digunakan. Kita melihat bahwa benderanya adalah qr, rd, ra. Ini adalah bendera yang sama yang digunakan di HEADER kedua. Namun, HEADER ketiga memiliki bendera qr dan aa. Apa perbedaan antara keempat bendera ini (qr, rd, ra, dan aa)? Bendera qr berarti kueri/respons. Bendera ini akan disetel dalam kueri apa pun, karena kami mengirimkan kueri ke server nama. Kedua flag rd dan ra berdiri untuk rekursi diinginkan dan rekursi diperbolehkan, masing-masing. Kueri rekursif berarti bahwa kueri akan diteruskan ke server nama lain jika server nama yang diberikan tidak dapat menyelesaikannya. Tanpa ini, kueri Anda mungkin dikembalikan tanpa respons yang memadai. Bendera aa adalah respons otoritatif. Karena hanya beberapa server nama yang berwenang untuk sebuah domain, respons yang di-cache di beberapa server nama lain tidak akan bersifat otoritatif. Dengan menyertakan aa di header, kami yakin bahwa kami memperoleh informasi ini dari otoritas. Dalam hal ini, perhatikan bahwa respons menyertakan bagian AUTHORITY, bukan bagian JAWABAN. Satu komentar terakhir: bagian pertama dari respons mencakup catatan A, bagian kedua berisi catatan AAAA (IPv6), dan bagian ketiga berisi informasi otoritas awal (SOA). Kita harus mendapatkan SOA hanya dari otoritas.

6.5 PENGKODEAN BASE64

Base64 bukanlah aplikasi, tetapi merupakan bagian penting dari komunikasi jaringan. Pengkodean Base64 adalah bentuk pengkodean Multipurpose Internet Mail Extensions (MIME). MIME dibuat agar file biner dapat diperlakukan sebagai teks, khususnya sehubungan dengan email, karena protokol email hanya dapat mengirimkan teks. Misalnya, melampirkan file gambar ke email tidak akan mungkin dilakukan tanpa beberapa bentuk pengkodean biner ke teks. Ada banyak bentuk pengkodean MIME. Base64, yang berasal dari tahun 1993 sebagai bagian dari RFC 1421, berurusan dengan peningkatan privasi untuk email, adalah cara populer untuk menyandikan informasi seperti teks terenkripsi atau bahkan kunci enkripsi.

Ide di balik Base64 adalah menerjemahkan urutan 6 bit menjadi karakter yang dapat dicetak (26 = 64). 64 karakter yang dapat dicetak adalah 26 huruf besar (A–Z), 26 huruf kecil (a–z), 10 digit (0–9), dan karakter "+" dan "/". Selain itu, karakter "=" digunakan sebagai kode akhiran khusus. Gambar 6.19 mencantumkan 64 karakter yang dapat dicetak yang digunakan dalam Base64 dan nilai bilangan bulatnya yang sesuai. Urutan 6-bit apa pun akan berkisar antara 000000 dan 111111. Urutan biner ini sesuai dengan bilangan bulat 0 hingga 63. Oleh karena itu, urutan biner 6-bit diubah menjadi bilangan bulat, dan bilangan bulat dipetakan ke karakter yang dapat dicetak.

Mari kita perhatikan sebuah contoh. Kami memiliki urutan 40-bit berikut yang ingin kami enkode.

1100010000101011100101101010101100101110

Kami pertama-tama membagi 40 bit menjadi kelompok masing-masing 6 bit.

110001 000010 101110 010110 101010 110010 1110

Perhatikan bahwa grup terakhir hanya terdiri dari 4 bit. Kami mengisinya dengan dua 0 ke kanan, memberi kami 111000. Sekarang, kami mengonversi setiap urutan 6-bit menjadi karakter yang dapat dicetak. Hasilnya adalah teks yang disandikan xBuWq04. Tabel 6.6 menunjukkan masing-masing grup 6-bit ini dengan nilai biner, bilangan bulat setara, dan karakter yang dapat dicetak.

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Gambar 6.19 Tabel indeks Base64.

Tabel 6.6 Contoh Urutan yang Dikodekan

Bilangan Biner	Bilangan bulat	Karakter yang Disandikan Base64
110001	49	X
000010	2	B
101110	46	u
010110	22	W
101010	42	Q
110010	50	0
111000	56	4

Contoh yang ditunjukkan di atas mungkin tidak terlalu intuitif karena kita tidak tahu apa yang dimaksud dengan 40 bit asli. Urutan data itu mungkin merupakan bagian dari rangkaian karakter, bagian dari kunci enkripsi, data terenkripsi, atau sesuatu yang lain sama sekali. Mari kita lihat contoh lain. Di sini, kita akan mulai dengan teks ASCII dan menyandikannya. Contoh kita akan menjadi string “NKU.” Setiap karakter dalam string kami disimpan dengan menggunakan 8 bit di ASCII. Kita perlu mengubah urutan dari pengelompokan 8-bit menjadi pengelompokan 6-bit. “NKU” dalam ASCII adalah 01001110 01001011 01010101 atau 24-bit 010011100100101101010101. Kami mengelompokkan ulang 24 bit menjadi grup 6-bit, sebagai gantinya memberi kami 010011 100100 101101 010101 (perhatikan di contoh ini kita tidak perlu mem-pad grup terakhir karena 24 bit terbagi rata

menjadi kelompok 6-bit). Dengan demikian, string “NKU” menjadi urutan empat karakter di Base64 dari “TktV”, seperti yang ditunjukkan pada Gambar 6.20.

Text	N								K								U							
ASCII	78								75								85							
Binary	0	1	0	0	1	1	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0	1	0	1
Index	19								36								45							
Base 64 Encoding	T								k								t							

Gambar 6.20 Pengkodean NKU Base64.

BAB 7
SISTEM NAMA DOMAIN

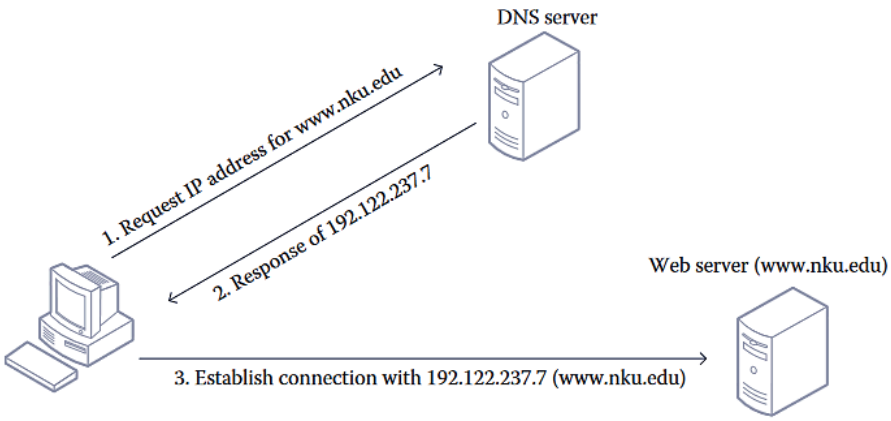
Setiap sumber daya di Internet memerlukan pengidentifikasi unik agar sumber daya lain dapat berkomunikasi dengannya. Pengidentifikasi unik ini disediakan dalam bentuk alamat numerik. Dua bentuk alamat yang umum digunakan adalah alamat Internet Protocol versi 4 (IPv4) dan Internet Protocol versi 6 (IPv6), seperti yang dijelaskan di Bab 3. Karena alamat IPv4 dan IPv6 disimpan dalam biner sebagai nilai 32-bit dan 128-bit , masing-masing, kita cenderung melihatnya sebagai empat bilangan bulat menggunakan notasi desimal bertitik (DDN) atau 32 digit heksadesimal. Orang-orang menyukai nama yang bermakna dan mudah diingat untuk mengidentifikasi sumber daya yang ingin mereka komunikasikan. Oleh karena itu, cara alamat disimpan dan dilihat dan cara kami ingin menggunakannya berbeda. Untuk menyederhanakan penggunaan Internet, Sistem Nama Domain (DNS) diperkenalkan. Dengan DNS, kita dapat mengganti alamat IPv4 atau IPv6 dengan alias yang jauh lebih mudah diingat. Misalnya, server web NKU memiliki alias www.nku.edu, sedangkan server web NKU memiliki alamat IPv4 192.122.237.7. Jelas, nama lebih mudah diingat.

Karena komputer dan peralatan jaringan (router dan switch) tidak diatur untuk menggunakan nama, kita harus memiliki mekanisme untuk menerjemahkan dari nama ke alamat dan sebaliknya. Di sinilah peran DNS. DNS sendiri terdiri dari database yang tersebar di Internet dengan informasi untuk dipetakan dari alias IP ke alamat IP atau sebaliknya. Proses aktual pemetaan dari alias ke alamat dikenal sebagai resolusi nama. Ada beberapa jenis perangkat lunak berbeda yang tersedia untuk menangani resolusi nama, beberapa di antaranya telah kita bahas di Bab 4 (mis., nslookup, dig, dan host).

Mari kita perhatikan sebuah contoh, seperti yang diilustrasikan pada Gambar 7.1. Seorang siswa ingin mengakses server web NKU, www.nku.edu. Siswa menggunakan komputer yang akan kita sebut sebagai klien DNS. Siswa mengetik http://www.nku.edu. ke bilah alamat browser webnya. Untuk mendapatkan halaman web yang diminta, klien menyatukan satu atau lebih paket Hypertext Transfer Protocol (HTTP) untuk dikirim ke server web NKU. Klien ini adalah bagian dari jaringan area lokal (LAN) yang berisi switch atau router. Switch/router memerlukan alamat IP, bukan host bernama, www.nku.edu. Tanpa alamat sebenarnya, switch/router tidak dapat meneruskan/merutekan permintaan dengan tepat. Dengan nama host dimasukkan ke dalam browser web, DNS mengambil alih.

Klien mengeluarkan kueri DNS. Kueri adalah permintaan yang dikirim ke server nama DNS untuk resolusi nama. Server nama DNS menerjemahkan www.nku.edu ke alamat yang sesuai, 192.122.237.7. Server DNS mengembalikan alamat ke klien sebagai respons terhadap permintaan kueri. Respons yang dikembalikan dari server DNS kemudian digunakan oleh klien untuk mengisi alamat di paket HTTP. Dengan alamat tujuan 192.122.237.7 sekarang tersedia, klien dapat mencoba membuat koneksi Transmission Control Protocol (TCP), pertama melalui sakelar/router lokalnya, melalui Internet, dan akhirnya ke server web itu sendiri. Setelah koneksi TCP dibuat, browser mengirimkan permintaan HTTP melalui koneksi TCP. Permintaan

menyertakan alamat IP klien sendiri sebagai alamat sumber. Server web NKU mengirimkan halaman web yang diminta dalam respons HTTP kembali ke klien dengan menggunakan alamat pengembalian (sumber). Terakhir, klien menampilkan halaman web di browser web. Perhatikan bagaimana penggunaan DNS bersifat transparan bagi pengguna akhir. Hal ini memang sudah seharusnya karena DNS disini khusus untuk mempermudah penggunaan internet. Selain DNS yang transparan, seringkali juga sangat cepat. Pengguna akhir tidak dibiarkan menunggu karena tugas tambahan resolusi nama.



Gambar 7.1 Terjemahan nama host dan alamat IP.

Dalam bab ini, kita berkonsentrasi pada DNS: apa itu, bagaimana penerapannya, dan banyak masalah yang diangkat DNS yang harus kita hadapi. Di Bab 6, kita akan mengeksplorasi perangkat lunak yang digunakan untuk mengimplementasikan server DNS yang disebut BIND. Kami juga melihat perangkat lunak terkait DNS lainnya.

7.1 INFRASTRUKTUR SISTEM NAMA DOMAIN

DNS didasarkan pada model jaringan klien-server. Infrastruktur DNS terdiri dari tiga komponen utama: klien DNS, server DNS, dan database DNS. Klien DNS mengirimkan permintaan permintaan DNS ke server DNS. Server DNS menerjemahkan nama host dalam permintaan DNS menjadi alamat IP dengan bantuan database DNS. Alamat IP yang dihasilkan dikembalikan sebagai bagian dari respons DNS ke klien DNS.

Klien Sistem Nama Domain

Klien DNS biasanya disebut penyelesai DNS. Resolver DNS adalah sekumpulan rutinitas pustaka yang menyediakan akses ke satu atau lebih server DNS. Resolusi DNS biasanya merupakan bagian dari tumpukan TCP/IP dari sistem operasi host. Aplikasi, seperti browser web atau klien email, yang berjalan di host memanggil penyelesai DNS saat mereka perlu menyelesaikan nama host. Tanggung jawab dari DNS resolver meliputi hal-hal berikut:

- menerima permintaan resolusi nama dari aplikasi
- nerating permintaan permintaan DNS
- menemukan permintaan kueri DNS ke server DNS yang dikonfigurasi
- menerima respons kueri dari server DNS
- tanggapan pertanyaan ching

- mengubah respons resolusi nama ke aplikasi yang meminta

Resolver DNS dapat melakukan resolusi nama dengan salah satu dari dua cara. Pertama, dapat menggunakan file host untuk menyelesaikan nama host menjadi alamat IP. File host adalah file teks yang berisi pemetaan nama host ke alamat IP. File ini biasanya bernama hosts (catatan: tidak ada ekstensi file pada file hosts). File host terletak di direktori /etc (/etc/hosts) di sistem operasi Linux/Unix/Mac OS X, sedangkan di Windows (7, 8, dan 10), file host terletak di %SystemRoot%\ system32 \drivers\etc\. Kami dapat menganggap file host sebagai database DNS lokal yang dapat melakukan permintaan resolusi nama dari aplikasi yang berjalan di host ini. Contoh file host dari host CentOS Linux ditampilkan sebagai berikut:

```
#
# Table of IP addresses and host names
#
127.0.0.1   localhost
192.168.1.1 router
192.168.1.2 printer
192.168.1.3 pc1
192.168.1.4 pc2
...
```

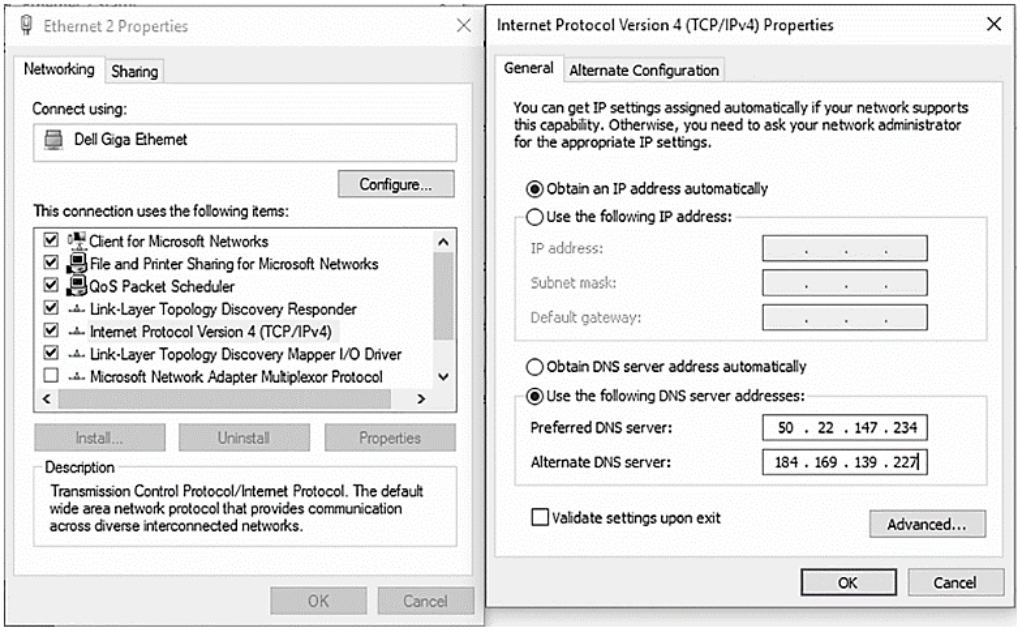
File host dikelola oleh administrator sistem klien. Pengguna ini harus menambahkan, mengubah, dan menghapus entri ke file ini secara manual. Idenya adalah melalui file host, resolusi DNS dapat ditangani sepenuhnya secara lokal, tanpa harus meminta server DNS di tempat lain di Internet. Namun, penggunaan file host bukannya tanpa kekurangan. Dengan menggunakan file host, entri yang salah atau kedaluwarsa akan menghasilkan resolusi nama yang salah dan oleh karena itu host tidak dapat dijangkau. Selanjutnya, seorang administrator sistem yang memelihara banyak komputer harus memodifikasi banyak file host ketika perubahan diperlukan, dan ini bisa memakan waktu dan membosankan.

Mari kita perhatikan contoh bagaimana entri yang salah dalam file host dapat menyebabkan host tidak dapat dijangkau. Anda, sebagai administrator sistem komputer Anda sendiri, telah menambahkan 127.0.0.1 www.google.com ke file host Anda. Sekarang, Anda ingin mengunjungi www.google.com. Resolusi nama menggantikan www.google.com dengan alamat IP seperti yang ditemukan di file host. Ini adalah alamat IP yang salah. Meski begitu, resolusi nama berhasil menyelesaikan permintaan tersebut. Browser web Anda membuat permintaan HTTP ke www.google.com, tetapi alamat IP dalam paket permintaan HTTP adalah 127.0.0.1, localhost Anda. Browser web Anda gagal menghubungi server web karena komputer lokal Anda tidak menjalankan server web. Sekarang, pertimbangkan bahwa Anda tidak memasukkan entri ini ke file host tetapi orang lain melakukannya dan Anda tidak menyadarinya. Meskipun mencoba, Anda tidak dapat menjangkau google, meskipun Anda berhasil menjangkau situs lain.

File host yang ditunjukkan di atas adalah file kecil. Ini tipikal karena kami ingin menghindari penggunaan file host untuk sumber daya apa pun yang alamat IP-nya mungkin berubah. Karena mengubah alamat IP mesin di luar jaringan langsung kami berada di luar kendali kami, biasanya entri dalam file host hanya untuk mesin lokal. Kami jarang akan menggunakan file host, dan ketika kami menggunakannya, itu akan menjadi sumber daya lokal

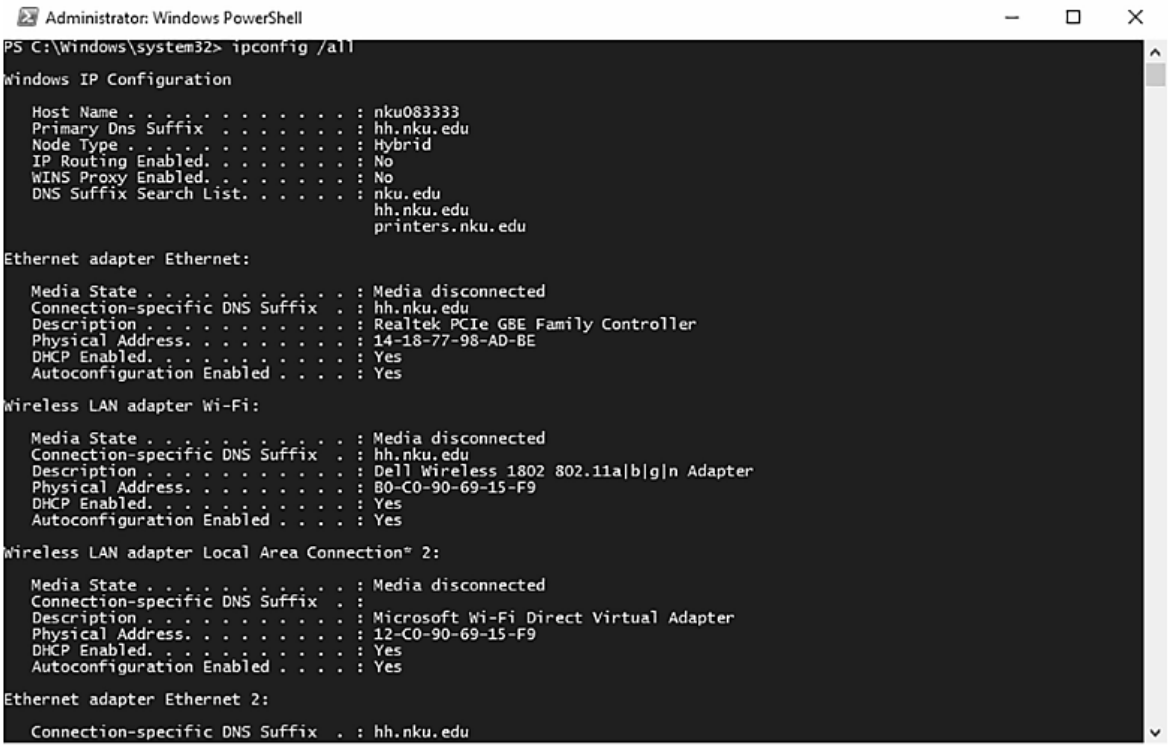
yang alamat IP-nya tidak hanya statis tetapi juga tidak akan berubah di masa mendatang (lebih disukai setidaknya bertahun-tahun).

Karena ada miliaran host di Internet, jelas tidak mungkin bagi administrator sistem untuk mengetahui semua host ini dan menambahkannya ke dalam satu file host. Dengan demikian, DNS resolver memerlukan pendekatan yang lebih terukur untuk menangani resolusi nama. Cara lain di mana DNS resolver dapat digunakan adalah melakukan resolusi nama dengan meminta server DNS yang dikonfigurasi di Internet. Untuk ini, kami membutuhkan server DNS. Sebagian besar LAN akan memiliki server DNS mereka sendiri atau memiliki akses ke server yang akan berkomunikasi dengan perangkat lokal. Oleh karena itu kami mengonfigurasi semua sumber daya lokal kami untuk menggunakan server DNS tertentu (apakah server DNS itu lokal atau di suatu tempat di Internet). Di Windows, pengaturan server DNS untuk DNS resolver dapat dikonfigurasi dan diperiksa secara manual, seperti yang ditunjukkan pada Gambar 7.2. Anda juga dapat mengonfigurasi server DNS Anda atau mendapatkan alamat server DNS dengan mengeluarkan perintah ipconfig /all atau ipconfig /registerdns dari jendela prompt perintah. Hal ini ditunjukkan pada Gambar 7.3.



Gambar 7.2 Pengaturan server DNS di Windows.

Dari Gambar 7.2, kita melihat bahwa ada dua server DNS yang ditentukan: server DNS pilihan dan server DNS alternatif. Dengan mengonfigurasi dua server DNS, kami dapat mencapai ketersediaan yang lebih tinggi. Jika kami hanya mengonfigurasi satu server DNS, sistem kami dapat mengalami satu titik kegagalan saat server DNS itu tidak aktif atau tidak dapat dijangkau (mis., karena kegagalan jalur perutean).



Gambar 7.3 Memeriksa informasi server DNS melalui perintah ipconfig di Windows.

Jika kami mengonfigurasi untuk dua server DNS, kami dapat menentukan server yang berlokasi di wilayah berbeda untuk mengurangi risiko kedua server tidak tersedia secara bersamaan. Resolver DNS akan selalu menanyakan server DNS pilihan terlebih dahulu. Jika server DNS pilihan tidak merespons, maka DNS resolver mengirimkan permintaan permintaan ke server DNS alternatif. Karena server DNS yang disukai ditanyakan terlebih dahulu, server yang lebih cepat (atau lebih dekat) lebih disukai. Misalnya, Anda dapat memilih server DNS dari ISP lokal Anda atau server DNS dalam LAN Anda sebagai server DNS pilihan karena dekat dengan host Anda. Anda mungkin memilih beberapa server DNS publik, seperti server DNS publik Google atau server OpenDNS sebagai server alternatif. Anda dapat memilih server DNS publik sebagai server pilihan Anda tanpa penurunan efisiensi biasanya karena situs tersebut mungkin memiliki cache DNS yang lebih besar daripada server DNS lokal Anda, sehingga lebih banyak kueri yang dapat diakses ke server pilihan Anda. Penyelesai DNS akan melihat latensi respons yang lebih pendek jika server DNS dapat menyelesaikan alamat secara lokal. Kita akan membahas cache DNS dan gagasan respons lokal versus nonlokal nanti di bab ini.

Di Linux, server DNS yang digunakan oleh DNS resolver untuk resolusi nama ditentukan dalam file bernama resolv.conf di bawah /etc. File resolv.conf adalah file konfigurasi DNS resolver, yang dibaca oleh DNS resolver pada saat aplikasi berbasis jaringan dipanggil. Contoh file resolv.conf dari host CentOS Linux adalah sebagai berikut:

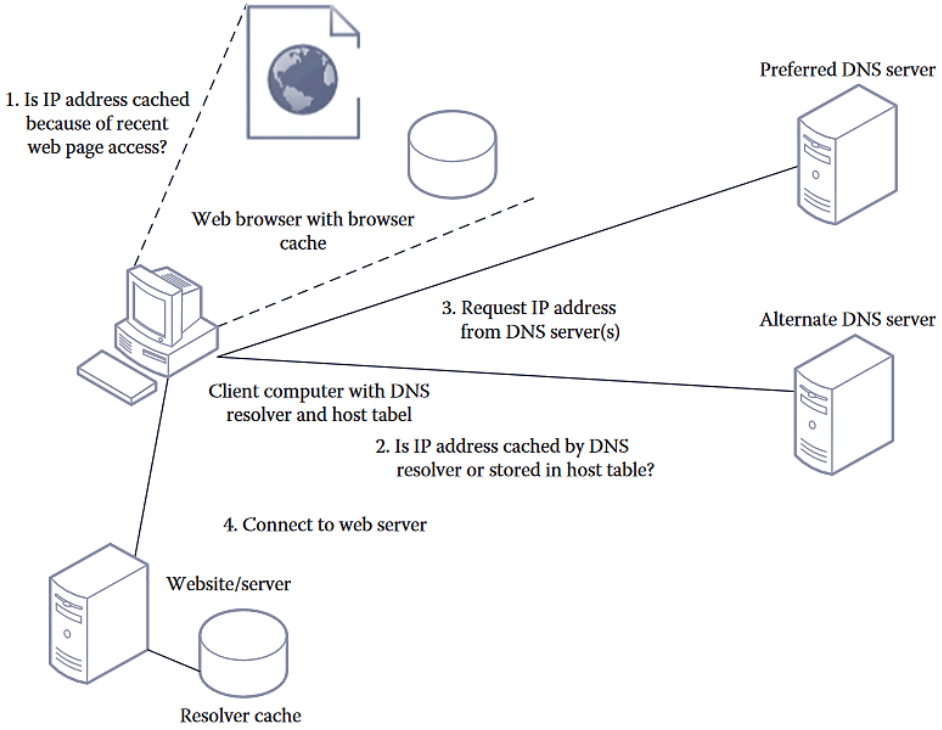
```
nameserver 10.10.8.8
nameserver 8.8.8.8
```

Anda juga dapat menjalankan perintah ifconfig untuk memeriksa pengaturan server DNS untuk DNS resolver. Selain konfigurasi manual server DNS, server DNS dapat dikonfigurasi secara otomatis untuk DNS resolver melalui Dynamic Host Configuration Protocol (DHCP).

Biasanya, DNS resolver menggunakan file host dan server DNS yang dikonfigurasi untuk melakukan resolusi nama. Mari kita perhatikan contoh bagaimana proses bekerja. Pertama, saat sistem operasi diinisialisasi, klien DNS akan dijalankan. Klien dimuat sebelumnya dengan entri apa pun yang ditemukan di tabel host. Itu juga memiliki cache dari alamat yang baru saja diselesaikan. Kami akan berasumsi bahwa pengguna telah memasukkan nama host di kotak alamat browser web. Sekarang, langkah-langkah berikut terjadi:

- Klien DNS pertama-tama memeriksa cache lokalnya (cache browser) untuk melihat apakah ada entri cache untuk nama host.
- Jika ada alamat IP yang di-cache, klien membuat alamat ini tersedia untuk browser untuk membuat koneksi HTTP ke host.
- Jika tidak, browser memanggil penyelesai DNS klien untuk resolusi nama.
- Penyelesai DNS memeriksa cache lokalnya (cache penyelesai), yang menyertakan entri file host yang dimuat sebelumnya.
- Jika ada alamat IP yang di-cache untuk nama host, alamat IP yang di-cache dikembalikan.
- Jika tidak, DNS resolver mengirimkan kueri DNS ke server DNS pilihannya.
- Jika tidak ada respons dalam jangka waktu tertentu, DNS resolver menanyakan server DNS alternatif.
- Saat respons kueri diterima, penyelesai DNS mem-parsing respons dan meng-cache nama host ke pemetaan alamat IP di cache penyelesai untuk permintaan di masa mendatang.
- Resolver DNS mengembalikan alamat IP ke aplikasi. Jika tidak ada respons sama sekali atau jika responsnya adalah nama host yang tidak valid, maka penyelesai mengembalikan kesalahan ke aplikasi (browser web dalam kasus ini). Lihat Gambar 7.4, yang mengilustrasikan proses ini.

Meskipun kami menggunakan browser web sebagai aplikasi dalam contoh ini, proses resolusi nama sisi klien ini berlaku untuk banyak aplikasi lainnya. Misalnya, klien email perlu menyelesaikan bagian alamat IP dari alamat email (yaitu, itu akan menghapus nama pengguna dan menyelesaikan bagian alamat setelah simbol @). Alternatifnya, perintah ssh akan membutuhkan penerjemahan alias IP host tujuan menjadi alamat IP.



Gambar 7.4 Proses resolusi nama sisi klien.

Dengan bantuan server DNS, DNS resolver dapat menyelesaikan permintaan aplikasi apa pun dari nama host ke alamat IP. Secara default, file host dikueri sebelum server DNS dikueri. Dengan menggunakan file host, kita dapat meningkatkan kinerja resolusi nama ketika nama host ditemukan di file host. Namun, kami dapat mengubah urutan kueri dari proses resolusi nama. Pada sistem Linux/Unix, kita dapat memodifikasi file nsswitch.conf, yang berada di bawah direktori /etc. File nsswitch.conf adalah file konfigurasi untuk utilitas Name Service Switch (NSS), yang menyediakan berbagai sumber untuk proses resolusi nama. Urutan layanan yang tercantum di nsswitch. conf menentukan urutan di mana NSS akan menerapkan layanan ini untuk melakukan resolusi nama. Entri konfigurasi file nsswitch.conf pada host Linux CentOS, yang mendefinisikan proses resolusi nama, ditunjukkan di bawah ini.

```
hosts: dns files
```

Entri konfigurasi ini menunjukkan bahwa DNS resolver akan meminta server DNS terlebih dahulu dan kemudian file host, hanya jika server DNS gagal menyelesaikan permintaan. Entri lain yang dapat ditempatkan dalam file ini termasuk nis jika Anda menjalankan layanan informasi jaringan dan db untuk menunjukkan bahwa database akan digunakan daripada file datar seperti file host.

Cache penyelesai dapat meningkatkan kinerja resolusi nama dan mengurangi lalu lintas jaringan dengan mengirimkan lebih sedikit pesan DNS. Sistem operasi Windows memiliki cache penyelesai di dalamnya. Saat sistem Windows melakukan kueri DNS, respons kueri disimpan oleh sistem operasi untuk digunakan di masa mendatang.

```
C: \Users\haow1> ipconfig  /displaydns

Windows IP Configuration

www.google.com
----------------------------------------------------------------
Record Name . . . . . . . . . . . . . : www.google.com
Record Type . . . . . . . . . . . . . : 1
Time To Live  . . . . . . . . . . . . : 189
Data Length . . . . . . . . . . . . . : 4
Section . . . . . . . . . . . . . . . : Answer
A (Host) Record . . . . . . . . . . . : 74.125.225.145


localhost
----------------------------------------------------------------
Record Name . . . . . . . . . . . . . : localhost
Record Type . . . . . . . . . . . . . : 1
Time To Live  . . . . . . . . . . . . : 86400
Data Length . . . . . . . . . . . . . : 4
Section . . . . . . . . . . . . . . . : Answer
A (Host) Record . . . . . . . . . . . : 127.0.0.1
```

Gambar 7.5 Entri DNS yang di-cache di Windows.

Di Windows untuk melihat entri DNS yang di-cache di cache penyelesai, jalankan ipconfig /displaydns dari prompt perintah Windows. Keluaran dari perintah ini menampilkan enam bidang untuk setiap entri DNS. Pertama, kolom Record Name ditampilkan, yang menyimpan nama host yang telah ditanyakan oleh DNS resolver dari server DNS. Berikutnya adalah kolom Record Type, yang menunjukkan bilangan bulat untuk tipe record. Jenisnya adalah jenis sumber daya yang ditanyakan. Misalnya, tipe 1 adalah record A, yang merupakan alamat IPv4, sedangkan tipe 28 adalah record AAAA atau alamat IPv6. Jenis lainnya termasuk CNAME, DNAME, CAA, CERT, IPSECKEY, MX, NS, dan PTR. Kami akan mengeksplorasi jenis sumber daya ini (dikenal sebagai catatan sumber daya) nanti di bab ini. Entri ketiga adalah waktu untuk hidup (TTL) dalam detik untuk berapa lama entri dapat disimpan dalam cache sebelum kedaluwarsa. Entri keempat adalah Panjang Data, yang menunjukkan ukuran respons kueri DNS dalam byte, yang akan menjadi salah satu dari 4, 8, dan 16, bergantung pada jenis nilai yang dikembalikan. Alamat IPv4 adalah 4 byte, alamat IPv6 adalah 16 byte, dan informasi lain seperti yang dikembalikan oleh catatan PTR dan CNAME adalah 8 byte. Entri kelima adalah jawaban atau tambahan, yang menunjukkan apakah respons tersebut mengandung informasi yang memerlukan upaya tambahan untuk mendapatkan respons yang diinginkan. Entri terakhir adalah alamat IP aktual yang diperoleh dari resolusi DNS.

Contoh output dari ipconfig ditunjukkan pada Gambar 7.5. Angka tersebut menunjukkan dua entri DNS yang di-cache. Dalam kedua kasus, entri menyimpan alamat IPv4 (catatan sumber daya, seperti yang ditunjukkan, item keenam dari kedua entri). Entri pertama pada gambar adalah catatan tembolok untuk www.google.com. Entri kedua bukanlah respons kueri DNS melainkan dihasilkan dari file host. Host lokal (alamat IP 127.0.0.1) memiliki nilai TTL yang jauh lebih panjang daripada respons DNS untuk kueri www.google.com karena alamat IP localhost disediakan oleh administrator sistem.

Di Linux, cache penyelesai dapat dikelola dengan daemon cache layanan nama, atau nscd. Anda dapat memasukkan /etc/rc.d/init.d/nscd dari baris perintah untuk memulai

daemon. Di Windows, kita dapat menghapus cache penyelesai dengan menggunakan perintah ipconfig, sebagai berikut:

```
ipconfig /flushdns
```

Di sistem Linux, kita dapat menghapus cache DNS dengan me-restart nscd, seperti perintah berikut:

```
/etc/rc.d/init.d/nscd restart
```

Server sistem nama domain

Server DNS memproses permintaan kueri DNS yang diterima dari klien DNS dan mengembalikan respons kueri DNS ke klien DNS. Oleh karena itu, server DNS perlu memiliki informasi terkini untuk menyelesaikan permintaan resolusi nama apa pun. Karena ada miliaran host di Internet, satu server DNS akan dipanggil untuk menangani semua permintaan DNS. Jika demikian, sistem DNS akan mengalami kinerja yang sangat lambat dan berisiko mengalami satu titik kegagalan. Juga sangat tidak nyaman bagi organisasi untuk harus mengirimkan informasi alamat yang diperbarui ke satu server yang berlokasi jauh.

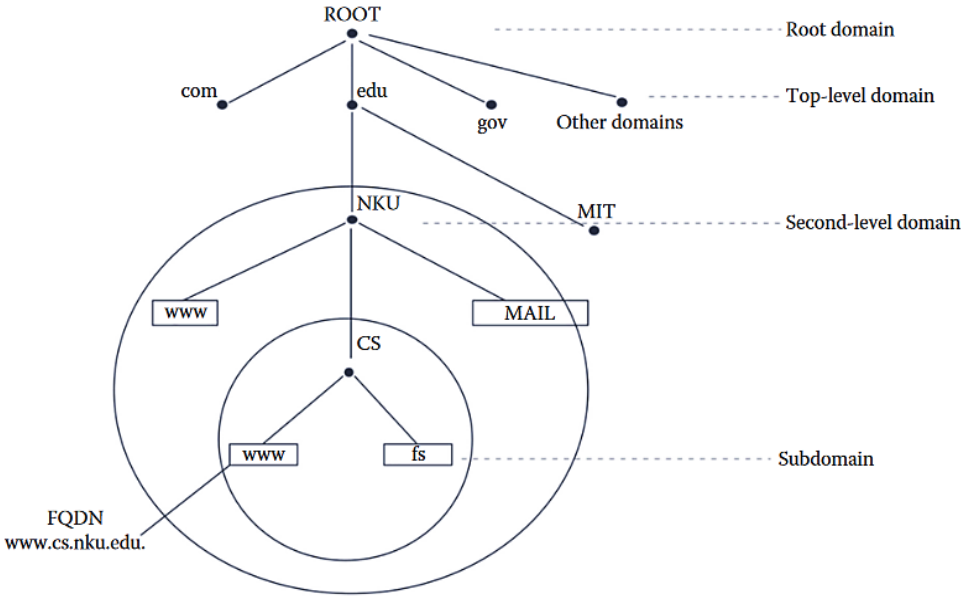
Agar DNS menjadi sangat andal dan dapat diskalakan, DNS menggunakan delegasi untuk menetapkan tanggung jawab atas sebagian namespace DNS ke server DNS yang dimiliki oleh entitas terpisah. Ruang nama DNS adalah subdivisi nama, yang disebut domain, untuk organisasi yang berbeda. Manfaat pendelegasian ruang nama adalah untuk mendistribusikan beban melayani permintaan kueri DNS di antara beberapa server DNS. Ini, pada gilirannya, mendukung skalabilitas, karena, karena lebih banyak ruang nama ditambahkan ke Internet, tidak ada satu pun server yang menerima beban lebih besar. Kami semakin meningkatkan kinerja DNS dengan mengatur domain secara hierarkis. Hal ini menyebabkan pemisahan antara jenis server DNS: yang berhubungan dengan domain tingkat atas (TLD) dan yang berhubungan dengan domain tertentu.

Kita melihat hirarki parsial namespace domain DNS pada Gambar 7.6. Pada level tertinggi adalah domain Root. Root domain mewakili root dari namespace DNS dan berisi semua TLD. Root domain diwakili oleh titik (.) di namespace DNS. Ada lebih dari 1000 TLD; namun, gambar tersebut hanya mengilustrasikan tiga di antaranya. Semua domain yang tersisa diklasifikasikan dalam TLD ini dan disebut domain tingkat kedua. Domain tingkat kedua ini terdiri dari semua domain yang telah diberi ruang nama, seperti nku, mit, google, cnn, facebook, dan sebagainya. Banyak dari domain tingkat kedua ini dibagi menjadi subdomain. Terakhir, di dalam domain atau subdomain mana pun, ada perangkat individual.

Sebuah organisasi yang memiliki domainnya sendiri bertanggung jawab untuk menangani subdomainnya dan memberikan informasi pemetaan yang akurat dari alias IP dari sumber dayanya ke alamat IP yang ditetapkan. Pada Gambar 7.6, NKU memiliki subdomain bernama CS. Domain NKU memiliki dua sumber bernama, www dan MAIL, dan subdomain CS memiliki dua sumber bernama tambahan, www dan fs. Perhatikan bagaimana nama yang sama dapat diulang karena muncul di ruang nama yang berbeda (yaitu, dengan berada di subdomain CS, nama www berbeda dengan nama NKU). Jadi, www.nku.edu dan www. cs.nku.edu dapat berupa entitas yang berbeda dengan alamat IP yang berbeda.

Ada dua jenis TLD: Domain Tingkat Atas generik (gTLD) dan Domain Tingkat Atas kode negara (ccTLD). gTLD tidak memiliki penunjukan geografis atau negara. Nama gTLD memiliki

tiga karakter atau lebih. Beberapa gTLD yang paling sering digunakan termasuk .com (Komersial), .edu (Lembaga Pendidikan AS), .gov (Pemerintah AS), .net (Penyedia Jaringan), dan .org (Organisasi Nirlaba). Daftar gTLD ditunjukkan pada Tabel 7.1 (dari http://en.wikipedia.org/wiki/Generic_top-level_domain). ccTLD dibuat untuk negara atau wilayah. Kode dua huruf digunakan untuk mewakili negara atau wilayah. Contoh ccTLD termasuk .cn (Cina), .de (Jerman), .uk (Inggris Raya), dan .us (Amerika Serikat).



Gambar 7.6 hierarki namespace DNS.

Nama domain tingkat atas dan nama domain tingkat kedua dikelola oleh InterNIC (Internet' Pusat Informasi Jaringan). Subdomain adalah domain yang dibuat oleh organisasi individu, pemilik domain tingkat kedua, di bawah domain tingkat kedua untuk mendukung organisasi yang lebih besar dalam domain tingkat kedua. Subdomain dikelola oleh masing-masing organisasi, bukan InterNIC.

Setiap host di ruang nama domain DNS diidentifikasi secara unik oleh Nama Domain yang Sepenuhnya Memenuhi Syarat (FQDN). FQDN mengidentifikasi posisi host dalam pohon hierarki DNS dengan menentukan daftar nama yang dipisahkan oleh titik di jalur dari host ke root. Pada Gambar 7.6, www. cs.nku.edu. adalah contoh FQDN. www adalah nama host. cs adalah subdomain tempat host www berada. nku adalah domain tingkat kedua yang telah membuat dan memelihara subdomain cs. edu adalah domain tingkat atas untuk institusi pendidikan tempat domain nku didaftarkan. Titik akhir (.) sesuai dengan domain root. Tanpa tanda titik, meskipun kami akan memiliki alias IP yang dapat digunakan browser web Anda, ini bukanlah FQDN. Ingatlah ini saat kami memeriksa DNS secara lebih rinci nanti di bab ini.

Server nama DNS dapat diklasifikasikan ke dalam server DNS root, server DNS domain tingkat atas, dan server DNS tingkat kedua/subdomain. Server DNS root hanya bertanggung jawab untuk domain root. Itu memelihara daftar otoritatif server nama untuk TLD. Ini menjawab permintaan catatan terkait level root dan server nama otoritatif untuk TLD.

13 root name server diberi nama a.root-servers.net melalui m.root-servers. bersih, seperti yang ditunjukkan pada Tabel 7.2. 13 server DNS root logis ini dikelola oleh organisasi yang berbeda. Perhatikan bagaimana kami merujuk ini sebagai 13 server DNS root logis. Ini tidak berarti bahwa ada 13 server fisik. Setiap server DNS root logis diimplementasikan dengan menggunakan sekelompok server fisik. Ada ratusan server fisik di lebih dari 130 lokasi berbeda yang berjalan sebagai server DNS root.

Tabel 7.1 Nama Domain Tingkat Atas Generik (gTLD)

Domain Abbreviation	Intended Use	Domain Abbreviation	Intended Use
aero	Air transport industry	mil	U.S. military
asia	Asia-Pacific region	mobi	Sites catering to mobile devices
biz	Business use	museum	Museums
cat	Catalan language/culture	name	Families and individuals
com	Commercial organizations	net	Network infrastructures (now unrestricted)
coop	Cooperatives	org	Organizations not clearly falling within the other gTLDs (now unrestricted)
edu	Post-secondary educational	post	Postal services
gov	U.S. government entities	pro	Certain professions
info	Informational sites	tel	Telephone network/Internet services
int	International organizations established by treaty	travel	Travel agents/airlines, tourism, etc.
jobs	Employment-related sites	xxx	Sex industry

Tabel 7.2 Server Nama DNS Root

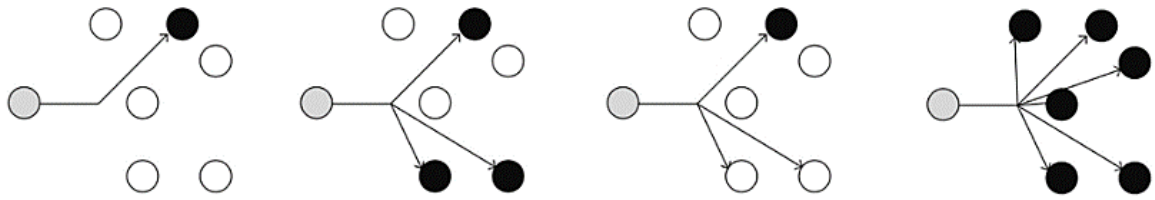
Hostname	IP Address	Manager
a. Root-server.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign. Inc.
b. Root-server.net	192.228.79.210	University of Southern California (ISI)
c. Root-server.net	192.33.4.12	Cogent Communications
d. Root-server.net	199.7.92.13, 2001:500:2d::d	University Of Maryland
e. Root-server.net	192.203.230.10	NASA (Ames Research Center)
f. Root-server.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium. Inc.
g. Root-server.net	192.112.36.4	U.S. Departement Of Defense (NIC)
h. Root-server.net	128.63.2.53, 2001:500:1::803:f:235	U.S. Army (Research Lab)
i. Root-server.net	192.36.148.17, 2001:7fe::53	Netnod
j. Root-server.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc
k. Root-server.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l. Root-server.net	199.7.83.42, 2001:500:3::42	ICANN
m. Root-server.net	202.12.27.33, 2001:dc3::35	WIDE Project

Setiap server nama TLD memiliki tanggung jawab untuk mengetahui domain tingkat kedua tertentu yang ada di dalam domainnya. Ini adalah server nama tingkat kedua dan subdomain yang memiliki tanggung jawab untuk menjawab permintaan sumber daya dalam domain khusus mereka sendiri.

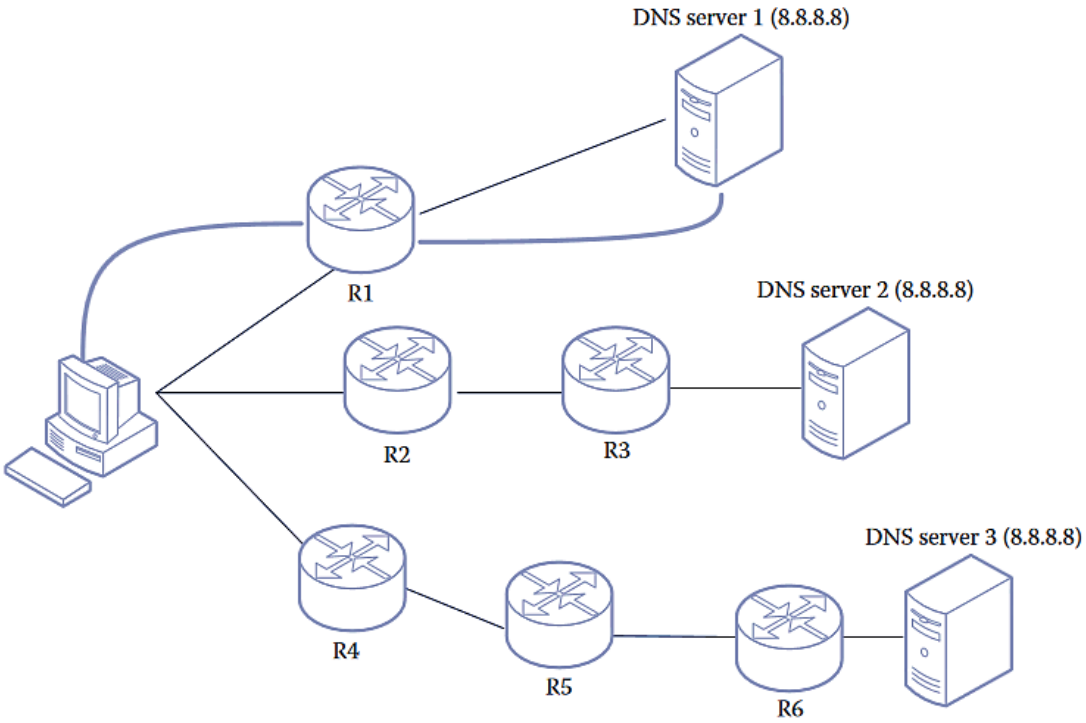
Setiap permintaan resolusi nama dimulai dengan permintaan DNS ke server root untuk mendapatkan alamat IP dari server nama TLD yang sesuai. Misalnya, kueri untuk menyelesaikan www.nku.edu akan dimulai dengan server root untuk menyelesaikan TLD edu. Dengan demikian, root server memainkan peran yang sangat penting dalam proses resolusi nama. Dengan alamat IP TLD yang sesuai, permintaan kueri DNS dapat dikirim ke server nama TLD tersebut untuk meminta resolusi domain tingkat kedua yang ditentukan, nku.edu dalam kasus ini. Server nama TLD merespons dengan alamat IP server nama domain level kedua (alamat IP untuk server nama NKU). Sekarang, permintaan dapat dibuat dari server nama domain tingkat kedua untuk resolusi nama lengkap. Perhatikan bahwa jika alamat IP sudah ada di cache, kueri dapat dihilangkan, dan jika alamat server nama TLD di-cache, permintaan ke tingkat root dapat dilewati.

Untuk mencapai ketersediaan tinggi dan kinerja tinggi di server DNS root, transmisi anycast digunakan untuk menyebarkan informasi. Setidaknya 6 dari 13 root server (C, F, I, J, K, dan M) telah mengadopsi anycast. Anda mungkin mengingat anycast dari Bab 1, di mana router akan merutekan paket dari node sumber ke node tujuan terdekatnya dari sekelompok tujuan (yang terdekat dianggap dalam hal biaya perutean dan kesehatan jaringan).

Mari kita lihat bagaimana anycast digunakan oleh server DNS. Gambar 7.7 memberikan perbandingan antara unicast, anycast, broadcast, dan multicast. Node berwarna abu-abu muda pada gambar mewakili node sumber (pengirim) dan node hitam mewakili node tujuan (penerima). Perhatikan bagaimana di anycast, meskipun pesan mungkin ditujukan ke beberapa lokasi, router mengirimkannya ke satu perangkat yang dianggap terdekat (bukan berdasarkan jarak tetapi dengan jeda waktu komunikasi). Anycast kemudian dikenal sebagai one-to-one-of-many daripada one-to-one (unicast) atau one-to-many (broadcast dan multicast). Biasanya, anycast cocok untuk situasi di mana sekelompok server berbagi alamat yang sama dan hanya satu server yang perlu menjawab permintaan dari klien.



Gambar 7.7 Membandingkan unicast, multicast, anycast, dan broadcast (kiri ke kanan)



Gambar 7.8 DNS Anycast.

Anycast DNS menyediakan layanan DNS dengan menggunakan alamat anycast yang sama tetapi dari beberapa lokasi geografis. Seperangkat satu atau lebih server DNS dikerahkan di setiap lokasi. Klien DNS mengirimkan kueri DNS ke alamat anycast. Kueri DNS diruteka n ke lokasi terdekat menurut lokasi klien DNS dan kebijakan perutean. Mari kita perhatikan contoh DNS anycast pada Gambar 7.8. Tiga server DNS anycast disebarkan di tiga lokasi berbeda dan dikonfigurasi dengan alamat anycast yang sama di 8.8.8.8. Klien DNS mengirimkan kueri ke alamat anycast 8.8.8.8 untuk melakukan resolusi nama. Jarak perutean antara klien DNS dan server DNS 1 adalah dua lompatan. Ada tiga lompatan antara klien DNS dan server DNS 2. Klien DNS berjarak empat lompatan dari server DNS 3. Dalam contoh ini, server DNS 1 adalah server nama terdekat dengan klien DNS. Dengan demikian, kueri dirutekan ke server DNS 1. Jika sembarang server DNS 1 tidak dapat dijangkau karena server 1 mati atau karena jalur perutean gagal, maka kueri klien DNS akan dirutekan ke server nama terdekat kedua, server DNS 2 , melalui router R2 dan R3. Selain itu, kueri klien dapat dimuat seimbang di antara server DNS anycast.

Dari contoh ini, kita dapat melihat bahwa DNS anycast memiliki beberapa keunggulan dibandingkan DNS unicast tradisional. Pertama, DNS anycast memiliki keandalan, ketersediaan, dan skalabilitas yang lebih baik daripada DNS unicast. Sistem DNS unicast mengalami satu titik kegagalan. Akan ada gangguan layanan jika server DNS mati atau jalur perutean ke server DNS gagal. Dalam sistem DNS anycast, sekelompok server DNS dengan alamat IP yang sama disebarkan di beberapa lokasi yang tersebar secara geografis. Jika server DNS terdekat tidak dapat dijangkau, permintaan kueri akan diteruskan ke server DNS terdekat kedua. Jika server DNS anycast kelebihan beban dengan permintaan, server DNS anycast baru dapat ditambahkan ke lokasi yang sama untuk berbagi beban. Klien DNS tidak perlu mengubah

konfigurasi apa pun. Dengan demikian, DNS anycast berisi redundansi untuk membuat layanan DNS anycast sangat tersedia, andal, dan dapat diskalakan.

Kedua, DNS anycast memiliki kinerja yang lebih baik daripada sistem DNS unicast. Dalam sistem DNS unicast, klien DNS akan mengalami latensi respons yang lebih lama karena permintaan mungkin datang dari tempat yang lebih jauh dan/atau karena server DNS menjadi lebih kelebihan beban. Dengan anycast DNS, kueri dirutekan ke server DNS terdekatnya, di mana server DNS terdekat merupakan kombinasi kedekatan dan beban jaringan, sehingga beberapa penyeimbangan beban terjadi. Kedekatan dan beban yang lebih rendah menghasilkan kinerja yang lebih baik.

Ketiga, DNS anycast lebih aman daripada DNS unicast. Dalam anycast DNS, server DNS yang tersebar secara geografis membuat layanan DNS lebih tahan terhadap serangan DOS apa pun karena serangan semacam itu hanya memengaruhi sebagian dari grup server DNS anycast. Keempat, anycast DNS menawarkan konfigurasi klien yang lebih mudah daripada DNS unicast. Dalam sistem DNS unicast, kita perlu mengonfigurasi server DNS yang berbeda untuk klien DNS yang berbeda. Misalnya, kami mengonfigurasi dua server DNS, server DNS pilihan dan server DNS alternatif, untuk klien DNS pada Gambar 7.2. Kita dapat menggunakan alamat IP anycast yang sama untuk semua klien DNS di sistem DNS anycast, sehingga tidak perlu membedakan server pilihan dan server sekunder. Router secara transparan mengalihkan kueri klien ke server DNS yang paling sesuai untuk resolusi nama.

Dari 13 root name server, salah satunya bernama K-root name server. Server ini beroperasi melalui domain RIPE NCC (wilayah yang menggabungkan Eropa dan Timur Tengah). Server nama K-root diimplementasikan dengan menggunakan DNS anycast di lebih dari 17 lokasi (London, Inggris; Amsterdam, Belanda; Frankfurt, Jerman; Athena, Yunani; Doha, Qatar; Milan, Italia; Reykjavik, Islandia; Helsinki, Finlandia; Jenewa, Swiss; Poznan, Polandia; Budapest, Hongaria; Tokyo, Jepang; Abu Dhabi, UEA; Brisbane, Australia; Miami, AS; Delhi, India; dan Novosibirsk, Rusia). Ini ditunjukkan pada Gambar 7.9 (milik http://k.root-servers.org/). Setiap node di suatu lokasi terdiri dari jaringan beberapa server untuk memproses permintaan DNS dalam jumlah besar. Anycast memungkinkan kueri klien dilayani oleh node k-root terdekat mereka. Misalnya, kueri dari klien DNS di Eropa dilayani oleh server nama k-root di Eropa, dan kueri dari klien DNS di Asia dilayani oleh server nama k-root di Asia.

Gambar 7.9 penyebaran K-root.

Seperti yang dinyatakan sebelumnya, server nama TLD bertanggung jawab atas TLD seperti domain teratas generik (mis., .com, .edu, .org, dan .net) dan TLD negara (mis., .us, .ca, dan .cn). Server nama DNS TLD dijalankan oleh pendaftar yang ditunjuk oleh Internet Corporation for Assigned Names and Numbers (ICANN). Misalnya, server DNS domain .com dioperasikan oleh Verisign.

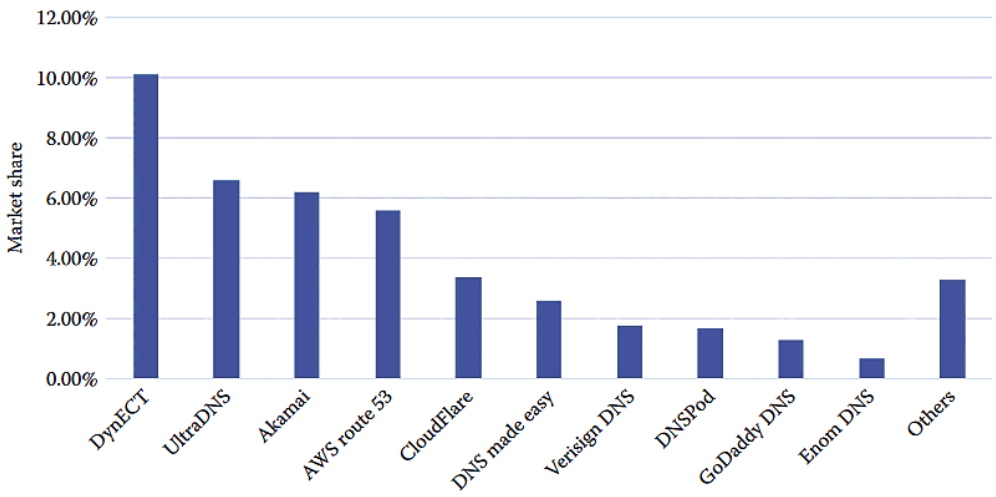
Perhatikan bahwa server nama Domain Root mengetahui server nama TLD, dan setiap server nama TLD mengetahui tentang subdomain dalam domainnya. Namun, hanya di dalam domain itulah subdomain dan sumber daya diketahui. Artinya, setiap server nama berisi database DNS tentang domainnya. Oleh karena itu, setiap host dalam domain memerlukan server DNS untuk menyimpan data DNS-nya. Catatan ini berisi data untuk menerjemahkan nama host ke alamat IP yang sesuai. Dengan demikian, setiap domain membutuhkan server DNS domain.

Server nama DNS domain tingkat kedua bertanggung jawab atas resolusi nama untuk domain tingkat kedua khusus mereka, seperti nku.edu dan amazon.com. Dalam setiap domain, mungkin ada satu atau lebih subdomain, seperti cs.nku.edu. Seringkali, subdomain memiliki server nama DNS sendiri yang bertanggung jawab atas kueri DNS untuk sumber daya apa pun di dalam subdomain tersebut. Saat klien DNS mengirimkan kueri untuk alamat IP host di domain atau subdomain tingkat kedua tertentu, server DNS root dan server DNS TLD tidak dapat memberikan jawaban tersebut. Jadi, sebagai gantinya, mereka merujuk kueri ke server DNS domain tingkat kedua yang sesuai, yang dengan sendirinya dapat merujuknya ke server subdomain jika domain tingkat kedua disiapkan dengan beberapa server hierarkis.

Organisasi atau individu dapat menentukan catatan DNS untuk domain dengan dua cara. Salah satu caranya adalah menyiapkan server nama DNS otoritatif master dan slave untuk domain tersebut. Kami akan membahas cara menerapkan server nama DNS otoritatif master dan slave di Bagian 5.1.3 dan lebih khusus lagi di BIND di Bab 6. Cara lainnya adalah dengan menggunakan layanan DNS terkelola.

Managed DNS adalah layanan yang memungkinkan organisasi mengalihdayakan catatan DNS mereka ke penyedia pihak ketiga. Gambar 7.10 memberikan perbandingan statistik pangsa pasar terbaru untuk layanan DNS terkelola di antara 10.000 situs web teratas Alexa (http://blog.cloudharmony.com/2014/02/dns-marketshare-alexa-fortune-500.html). Misalnya, kami melihat bahwa server DNS terkelola yang paling populer adalah DynECT, UltraDNS, Akamai, Amazon Web Services (AWS) Route 53, dan CloudFlare. Ada beberapa keuntungan menggunakan layanan DNS terkelola untuk menghosting data DNS domain Anda.

Pertama, Anda tidak perlu menyiapkan server DNS master/slave Anda sendiri untuk mengelola DNS. Alasan lainnya adalah kueri DNS untuk domain Anda memiliki latensi respons yang lebih pendek karena penyedia layanan DNS terkelola menggunakan jaringan server DNS global mereka untuk menyimpan catatan DNS Anda. Selain itu, akses ke layanan DNS untuk domain Anda lebih andal dan aman karena merupakan DNS yang dikelola oleh penyedia yang akan melakukan replikasi dan redundansi dalam platformnya. Di Bab 12, kita akan melihat cara menggunakan Amazon Route 53 untuk menghosting catatan DNS untuk sebuah domain.



Gambar 7.10 Penyedia layanan Managed DNS.

Server DNS juga dapat diklasifikasikan menjadi otoritatif dan tidak otoritatif. Server nama DNS otoritatif adalah server yang dikonfigurasi dengan pemetaan nama host-ke-alamat IP untuk sumber daya domain/subdomainnya. Server nama DNS otoritatif memberikan jawaban pasti untuk pertanyaan DNS. Server nama DNS akar, server nama DNS TLD, dan server nama DNS tingkat kedua/subdomain semuanya adalah server DNS otoritatif untuk domain mereka sendiri.

Lalu, apa itu name server nonauthoritative? Server nama non-otoritatif bukanlah otoritas untuk domain apa pun. Itu tidak berisi catatan DNS resmi dari domain apa pun. Lalu, bagaimana pemetaan nama host ke alamat IP? Server nama DNS non-otoritatif memiliki file cache yang dibuat dari kueri dan respons DNS sebelumnya. Ketika server non-otoritatif mengkueri server otoritatif dan menerima jawaban otoritatif, server tersebut menyimpan jawabannya dalam cache-nya. Cache ini digunakan untuk mengurangi latensi kueri DNS. Misalnya, server nama DNS lokal mungkin meng-cache respons kueri yang diterima dari kueri

sebelumnya. Sekarang, klien tidak perlu menunggu server nama DNS otoritatif untuk merespons; itu menunggu server nama lokal.

Setiap jawaban yang diambil dari cache server nama yang tidak otoritatif dianggap tidak otoritatif karena tidak berasal dari server yang otoritatif. Server DNS non-otoritatif terkadang disebut server DNS caching atau server DNS penerusan. Server DNS lokal, yang merupakan server nama yang biasanya dikelola oleh ISP Anda, adalah contoh server DNS non-otoritatif. Server nama DNS lokal biasanya tidak berwenang untuk domain apa pun. Server nama DNS lokal menyimpan informasi DNS yang diambil dari server nama otoritatif di tingkat akar, TLD, domain tingkat kedua, dan subdomain dari respons DNS terbaru. Cache tidak hanya mempercepat kueri DNS di masa mendatang, tetapi juga mengurangi beban pada server dan jaringan nama DNS otoritatif di Internet. Jika kueri DNS tidak dapat dilayani oleh server nama DNS lokal melalui cache-nya, kueri DNS diteruskan ke server nama DNS lain untuk penyelesaian. Sama seperti kami melihat bahwa permintaan DNS kami beralih dari tingkat Root ke TLD ke domain tingkat kedua dan oleh karena itu diteruskan dari server ke server, permintaan kami yang belum terselesaikan secara lokal diteruskan dari server ke server, bekerja hingga ke tingkat akar. Yaitu, kueri, jika tidak dipenuhi secara lokal, mungkin diteruskan ke server nama DNS lain yang memiliki resolusi nama yang di-cache. Jika tidak, maka diteruskan lagi. Akhirnya, diteruskan ke tingkat Root, yang merupakan server DNS otoritatif dan dapat menjawab sebagian dari resolusi, tetapi karena tidak dapat menjawab seluruh kueri, kueri terus diteruskan tetapi, dalam hal ini, menuruni hierarki nama server, hingga mencapai server nama subdomain atau level kedua yang sesuai. Jika kueri tidak dipenuhi dari cache mana pun, maka responsnya akan bersifat otoritatif. Jika kueri dipenuhi oleh cache apa pun, itu tidak akan menjadi tidak otoritatif. Jumlah penerusan mungkin hanya satu atau lebih, tergantung pada pengaturan LAN Anda dan koneksinya ke Internet.

Mari kita periksa respons DNS yang tidak otoritatif. Di sini, kami menggunakan perintah nslookup untuk mendapatkan alamat IP untuk nama host www.google.com. Karena situs web Google sering dikunjungi, kueri DNS terbaru untuk alamat IP telah dibuat dan di-cache. Pada Gambar 7.11, kita melihat bahwa server nama DNS lokal kita, nkuserv1.hh.nku.edu, merespons. Kami tahu ini karena kami diberi tahu bahwa tanggapannya tidak otoritatif.

Kami dapat mengetahui apakah server itu berwibawa atau tidak dengan berkonsultasi dengan domain untuk server namanya. Ini dilakukan dengan menggunakan perintah host Linux/Unix. Opsi –t memungkinkan Anda untuk menentukan jenis sumber daya mana yang Anda minati. Jenis ns adalah server nama. Saat melakukan host –t ns google. com, kami menerima empat tanggapan, server nama google.com ns1.google.com, google. com server nama ns2.google.com, server nama google.com ns3.google.com, dan server nama google.com ns4.google.com, yang merupakan empat server nama Google.

Kembali ke contoh kita dari Gambar 7.11, bagaimana jika kita menginginkan jawaban yang berwibawa? Kami perlu memastikan bahwa permintaan kami dijawab oleh server resmi daripada server lokal kami. Secara default, nslookup menanyakan server lokal kami terlebih dahulu. Sebagai gantinya, kami dapat menentukan server nama yang akan digunakan dengan menambahkan server itu di akhir permintaan kami dan dengan demikian tidak menggunakan

server lokal kami. Pada Gambar 7.12, kita melihat perintah nslookup yang telah direvisi, di mana kita mencari alamat IP untuk www.google.com tetapi kita meminta server nama DNS ns1.google.com untuk memberikan tanggapan. Hasilnya adalah otoritatif, yang dapat kita simpulkan dari gambar tersebut, karena tidak mengatakan jawaban yang tidak otoritatif.

```
C:\Users\harvey>nslookup  www.google.com
Server :    nkuserv1.hh.nku.edu
Address : 172.28.102.11


Non-authoritative answer :
Name :      www.google.com
Addresses :       2607:f8b0:4000:802::1014
            74.125.225.241
            74.125.225.243
            74.125.225.244
            74.125.225.240
            74.125.225.242
```

Gambar 7.11 Respons DNS tidak resmi.

```
C: \Users\harvey>nslookup www.google.com ns1.google.com
Server:     ns1.google.com
Address:    216.239.32.10

Name:       www.google.com
Addresses:  2607:f8b0:4000:802::1013
            173.194.46.20
            173.194.46.17
            173.194.46.19
            173.194.46.18
            173.194.46.16
```

Gambar 7.12 Jawaban otoritatif.

Jika Anda membandingkan alamat IP dari Gambar 7.11 dan 5.12, Anda dapat melihat bahwa kami menerima nilai yang berbeda. Mengapa respons otoritatif berbeda? Ingatlah bahwa server nama DNS lokal kami telah meng-cache responsnya dari permintaan sebelumnya. Apakah data yang di-cache masih valid? Mari kita cari tahu kapan entri DNS yang di-cache akan kedaluwarsa di server DNS lokal. Kita dapat menggunakan opsi debug nslookup untuk mendapatkan nilai TTL entri. Gambar 7.13 menunjukkan bagian terakhir dari output untuk perintah nslookup –debug www.google.com ns1.google.com.

Dari Gambar 7.13, kita dapat menemukan TTL untuk respons yang di-cache. Di sini, kita melihat bahwa respons hanya valid selama 5 menit (300 detik). Server nama DNS lokal diizinkan untuk menggunakan kembali alamat IP yang di-cache ini untuk www.google.com selama 5 menit sebelum kedaluwarsa.

Klien DNS dapat mengirim dua jenis kueri ke server DNS. Salah satunya adalah kueri iteratif dan yang lainnya adalah kueri rekursif. Kueri iteratif memungkinkan server DNS untuk

memberikan sebagian jawaban. Itu dilakukan dengan memberikan informasi lokal terbaik yang dimilikinya, tanpa meminta server DNS lain. Jika server DNS tidak memiliki catatan DNS otoritatif untuk nama host yang diminta, server DNS merujuk klien DNS ke server DNS otoritatif untuk mendapatkan informasi. Ini memaksa klien DNS untuk melanjutkan resolusi namanya di tempat lain.

```
Got answer:
    HEADER:
      opcode = Query, id = 4, rcode = NOERROR
      header flags: response, auth. answer, want recursion
      questions = 1, answers = 5, authority records = 0, additional = 0

    QUESTIONS:
      www.google.com,    type  = A,         class=IN
      ANSWERS:
      ->  www.google.com
            Internet address = 74.125.225.243
            ttl = 300 (5 mins)
      ->  www.google.com
            Internet address = 74.125.225.241
            ttl = 300 (5 mins)
      ->  www.google.com
            Internet address = 74.125.225.242
            ttl = 300 (5 mins)
      ->  www.google.com
            Internet address = 74.125.225.244
            ttl = 300 (5 mins)
      ->  www.google.com
            Internet address = 74.125.225.240
            ttl = 300 (5 mins)
--------------------
--------------------
Got answer:
    HEADER:
      opcode = Query, id = 5, rcode = NOERROR
      header flags: response, auth. answer, want recursion
      questions =1, answers = 1, authority records = 0, additional = 0
    QUESTIONS:
      www.google.com, type = AAA, class = IN
    ANSWERS:
      -> www.google.com
          AAAA IPv6 address = 2607:f8b0:4000:802:1013
          ttl = 300 (5 mins)
-------------------
Name:         www.google.com
Addresses:  2607:f8b0:4000:802:1013
```

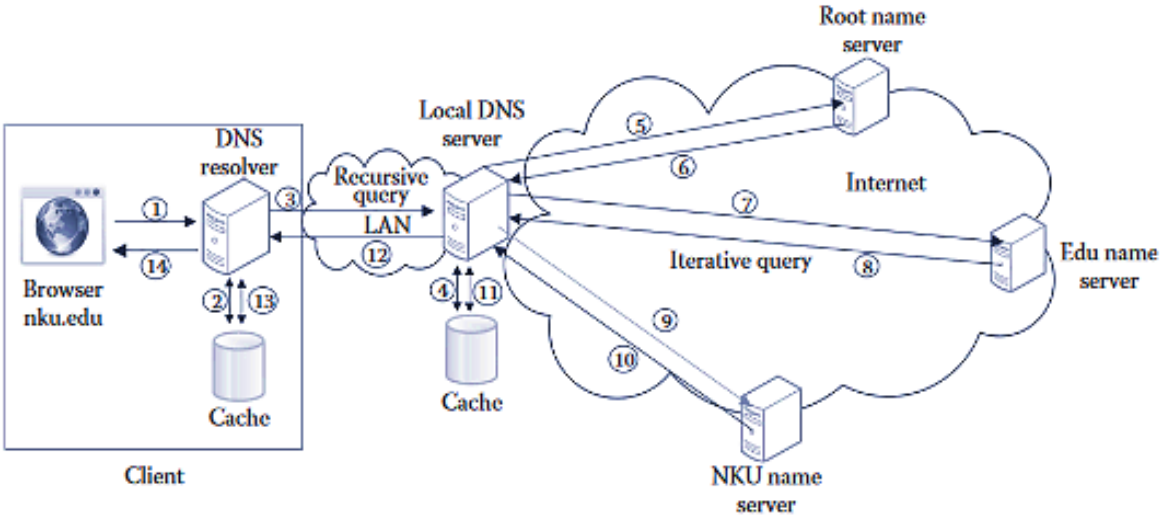
Gambar 7.13 Bagian kedua dari respons nslookup dengan opsi debug.

Kueri rekursif mengharuskan server nama DNS yang diminta harus menjawab kueri sepenuhnya. Jadi, jika informasi tersebut tidak sepenuhnya tersedia secara lokal, itu adalah server nama DNS yang meneruskan permintaan ke server nama DNS lain. Ini dilakukan dengan mengirimkan permintaan iteratif berturut-turut ke server nama lain. Pertama, ini menanyakan server nama DNS root, diikuti oleh server nama DNS TLD dan akhirnya server nama DNS domain tingkat kedua yang otoritatif. Klien DNS biasanya mengirim kueri rekursif ke server DNS lokal mereka, yang kemudian memenuhi permintaan itu sendiri atau meneruskannya secara iteratif. Jika permintaan rekursif dibuat dari server nama DNS lokal, yang dengan sendirinya tidak dapat menyelesaikan permintaan melalui permintaan berulang kepada orang lain, ini akan mengembalikan pesan kesalahan sebagai tanggapan atas permintaan asli.

Sebagai contoh, seorang siswa ingin mengakses situs web NKU dan mengetikkan www.nku.edu di browser webnya. Kami berasumsi bahwa dia mengakses situs web NKU untuk pertama kalinya dan klien pengguna berada di domain tingkat kedua zoomtown.com. Kami selanjutnya berasumsi bahwa server nama DNS lokal pengguna tidak memiliki entri yang di-cache untuk www.nku.edu. Berikut adalah penjelasan langkah demi langkah dari proses yang terjadi:

1. Peramban web perlu menerjemahkan nama host yang ditentukan (www.nku.edu) ke dalam alamat IP-nya. Jadi, browser web meneruskan www.nku.edu ke DNS resolver untuk resolusi nama.
2. Resolver DNS memeriksa file host dan cache untuk melihat apakah sudah memiliki alamat untuk nama ini. Karena pengguna belum mengunjungi situs ini, tidak ada entri DNS yang di cache untuk www.nku.edu.
3. Penyelesai DNS menghasilkan kueri DNS rekursif dan mengirimkannya ke server nama DNS lokalnya.
4. Server nama DNS lokal menerima permintaan dan memeriksa cache-nya.
5. Karena tidak ada entri yang cocok di cache, server nama DNS lokal membuat kueri berulang untuk www.nku.edu dan mengirimkannya ke server nama root.
6. Root name server tidak mengetahui alamat IP dari www.nku.edu. Server nama root mengetahui server DNS otoritatif untuk domain tingkat atas .edu, sehingga mengembalikan alamat server nama DNS .edu ke server DNS lokal.
7. Server DNS lokal menghasilkan kueri iteratif untuk www.nku.edu dan mengirimkannya ke server nama DNS .edu.
8. Server nama DNS .edu juga tidak mengetahui alamat IP www.nku.edu, tetapi mengetahui semua domain tingkat kedua di bawah edu, sehingga server nama DNS untuk domain nku.edu diketahui. Server nama DNS .edu menampilkan alamat server nama DNS nku.edu.
9. Sekarang, server DNS lokal zoomtown.com menghasilkan kueri iteratif untuk www.nku.edu dan mengirimkannya ke server nama DNS nku.edu.
10. Server nama DNS nku.edu memiliki informasi otoritatif tentang domain NKU, termasuk entri untuk server web www.nku.edu. Ia menemukan alamat IP www.nku. edu dan mengembalikannya ke server nama DNS lokal zoomtown.com.
11. Server nama lokal zoomtown.com menerima respons dari server nama DNS nku.edu. Ini menyimpan alamat untuk akses di masa mendatang.
12. Server nama lokal zoomtown.com meneruskan alamat IP ke penyelesai DNS klien.
13. Resolver DNS klien juga melakukan dua hal. Pertama, cache alamat IP www.nku. edu untuk akses di masa mendatang.
14. Resolver DNS klien meneruskan alamat IP ke browser. Browser menggunakan alamat IP untuk memulai koneksi HTTP ke host tertentu.

Perhatikan bahwa keseluruhan interaksi ini biasanya memerlukan waktu ratusan milidetik atau kurang. Gambar 7.14 memperlihatkan langkah-langkah proses resolusi nama DNS.



Gambar 7.14 Proses resolusi nama antara klien DNS dan server DNS.

Dalam contoh ini, server nama root, server nama TLD untuk domain edu, dan server nama DNS untuk subdomain nku.edu adalah server DNS otoritatif. Server DNS lokal adalah server DNS yang tidak resmi. Klien DNS (melalui penyelesainya) mengirimkan kueri rekursif ke server DNS lokal. Server DNS lokal tidak memiliki jawaban otoritatif atas pertanyaan tersebut. Untuk menjawab kueri sepenuhnya, server DNS lokal mengirimkan kueri iteratif secara berurutan ke server DNS otoritatif di tingkat akar, tingkat domain edu, dan domain tingkat kedua nku.edu. Saat menerima kueri berulang, server DNS otoritatif dapat memberikan jawaban otoritatif yang terdiri dari catatan DNS yang diminta atau sebagian jawaban berdasarkan informasi lokal terbaik yang dimilikinya. Server DNS otoritatif tidak meminta server DNS lain secara langsung untuk menjawab kueri berulang. Dalam contoh ini, server nama root dan server nama edu membalas ke server DNS lokal dengan sebagian jawaban. Karena server nama domain nku.edu otoritatif untuk domain NKU, ini berisi semua data DNS untuk domain NKU dan membalas server DNS lokal dengan data DNS yang diminta.

Basis data sistem nama domain

Server nama DNS memerlukan nama host untuk menangani data resolusi. Ini disimpan dalam file yang merupakan cache atau catatan sumber daya otoritatif, atau keduanya. Kami mengacu pada file yang menyimpan catatan sumber daya otoritatif sebagai database DNS. Catatan sumber daya berisi berbagai jenis informasi, di mana jenis catatan menentukan jenis informasi yang disimpan. Informasi ini bukan sekadar pemetaan nama host ke alamat IP, seperti yang kita lihat pada file host sebelumnya. Sebaliknya, setiap catatan sumber daya terdiri dari lima bidang, yang dijelaskan pada Tabel 7.3. Berdasarkan tipe (bidang keempat), tipe data sebenarnya akan bervariasi. Jenis yang paling umum digunakan dijelaskan pada Tabel 7.4.

Sistem DNS diimplementasikan sebagai database terdistribusi. Database DNS dapat dipartisi menjadi beberapa zona. Zona adalah bagian dari database DNS. Zona berisi semua domain dari titik tertentu ke bawah dalam hierarki namespace DNS, kecuali yang otoritatif.

Mari kita lihat contoh zona. Pada Gambar 7.15, domain tingkat kedua yang diberikan memiliki dua subdomain cs (departemen Ilmu Komputer) dan bi (departemen Informatika

Bisnis). Subdomain cs memiliki dua server nama DNS otoritatif (cs-ns1 dan cs-ns2). Server nama DNS ini menjawab semua pertanyaan tentang host dalam subdomain cs (seperti cs-www.cs.nku.edu dan cs-fs.cs.nku.edu). Departemen Informatika Bisnis tidak memiliki server nama DNS otoritatif di subdomain bi-nya. Oleh karena itu, server nama DNS NKU (ns1 dan ns2) menjawab semua pertanyaan tentang host di subdomain bi (bi-www.bi.nku.edu dan bi-fs.bi.nku.edu). Kami kemudian melihat bahwa untuk contoh ini, namespace domain NKU dibagi menjadi dua zona: zona NKU dan zona CS. Zona NKU mencakup domain NKU dan subdomain bi tetapi bukan subdomain cs.

Kami akan memerlukan dua file zona terpisah, satu untuk zona NKU (yang akan menyertakan semua catatan domain NKU, termasuk subdomain bi tetapi bukan subdomain cs) dan satu untuk zona CS (yang berisi informasi DNS hanya untuk subdomain cs). Dari contoh ini, Anda dapat melihat bahwa domain berbeda dengan zona. Sebuah zona dapat berisi satu domain dan nol, satu, banyak, atau semua subdomainnya.

Tabel 7.3 Bidang Catatan Sumber Daya

Field Name	Arti
Name	Domain sumber daya yang diberikan. Seringkali dapat dihilangkan, karena kami akan menentukan domain default untuk semua record.
TTL	Waktu hidup rekaman saat salinan rekaman ini di-cache di suatu tempat oleh server nama DNS non-otoritatif. Setelah catatan kedaluwarsa, itu harus dihapus dari cache. TTL ditentukan dalam hitungan detik; misalnya, 1800 berarti 30 menit. TTL sering dihilangkan, karena ada TTL default untuk seluruh zona yang digunakan saat sumber daya tidak memiliki TTL sendiri.
Class	Kelas jaringan di mana IN, Internet, adalah entri yang paling umum digunakan tetapi yang lain ada seperti CH (Chaos) dan HS (Hesiod).
Type	Jenis rekaman yang menunjukkan cara menginterpretasikan bidang nilai. Kami mengeksplorasi jenis umum pada Tabel 7.4.
Value	Nilai untuk item bernama ini berdasarkan jenis yang ditentukan. Nilainya spesifik untuk tipe tertentu. Lihat Tabel 7.4 untuk detail lebih lanjut.

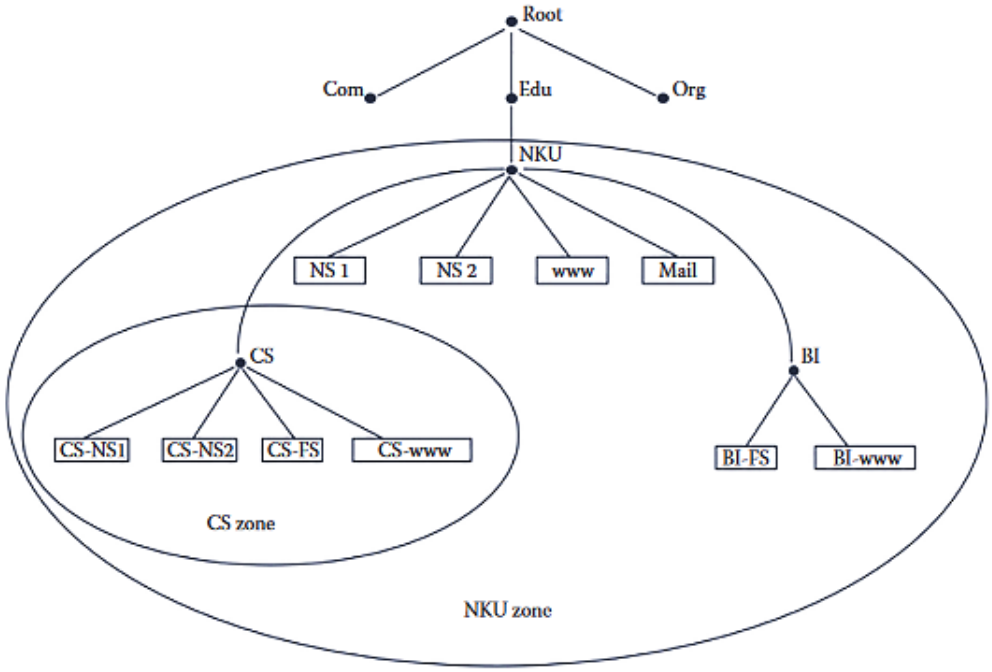
Tabel 7.4 Jenis Catatan Sumber Daya yang Biasa Digunakan

Jenis Rekam	Arti
A (Address) record	Alamat IPv4 untuk host. Ketika sebuah host memiliki beberapa alamat IP, itu harus memiliki satu catatan sumber daya per alamat.
AAAA (IPv6 Address) Record	Alamat IPv6 untuk host. Ukuran nilai record ini adalah 128 bit, sedangkan nilai IPv4 hanya 32 bit.
CNAME (Canonical Name) record	Jenis catatan sumber daya ini menentukan alias untuk sebuah host. Catatan menentukan alias dan kemudian nama sebenarnya dari host. Misalnya, jika Anda memiliki nama host www. nku.edu dan Anda ingin mengizinkan alias nku.edu, catatan sumber dayanya adalah nku.edu. CNAME www.nku.edu.

MX (Mail eXchange) record	Menentukan sumber daya ini sebagai server email untuk domain. Nilai akan menyertakan nama server email dan nilai preferensi. Preferensi menunjukkan urutan perutean untuk router saat beberapa server email tersedia. Jika domain nku.edu memiliki dua server email bernama ms1 dan ms2, dengan nilai prioritas masing-masing 100 dan 200, kita mungkin memiliki dua catatan sumber daya berikut: nku.edu 86400 IN MX 100 ms1.nku.edu nku.edu 86400 IN MX 200 ms2.nku.edu Dalam hal ini, email ke domain nku.edu akan dialihkan ke ms1 terlebih dahulu, karena ini adalah server email yang lebih disukai (semakin rendah angkanya, semakin tinggi preferensi server tersebut). Hanya jika ms1 tidak merespons ms2 akan dicoba.
PTR (Pointer) record	Memetakan alamat IP ke nama host. Ini kebalikan dari apa yang dilakukan catatan A. Kami menggunakan catatan PTR untuk menangani permintaan pencarian DNS terbalik. Catatan sumber daya PTR berikut memetakan alamat IP pc1.nku.edu ke nama domainnya: 11.0.12.10.in-addr.arpa. DI PTR pc1.nku.edu. Berbeda dengan catatan A, alamat IP dibalik, dan kami telah menambahkan in-addr.arpa. Permintaan alamat IP terbalik sering digunakan untuk tujuan keamanan untuk memastikan bahwa alamat IP tidak dipalsukan.
NS (Name Server) record	Menentukan server nama otoritatif untuk domain. Seperti halnya data A dan MX, beberapa server nama akan ditentukan dengan beberapa data. Berbeda dengan jenis server email, tidak ada nilai preferensi yang ditentukan dalam catatan NS.
SOA (Start Of Authority) record	Jenis rekaman ini hanya digunakan sekali per zona (mis., subdomain) untuk menentukan karakteristik seluruh zona. SOA terdiri dari dua bagian informasi yang terpisah. Yang pertama memiliki format yang sama dengan catatan sumber daya lainnya (mis., nama, TTL, kelas, jenis, dan nilai), dengan kolom keenam untuk alamat email administrator domain zona. Mengikuti catatan dan di dalam tanda kurung adalah lima nilai pada baris terpisah. Nilai-nilai ini menunjukkan nomor seri file zona, kecepatan refresh, kecepatan coba lagi, waktu kedaluwarsa, dan TTL domain. Sebagian besar nilai ini digunakan untuk mengontrol server nama budak.

Delegasi adalah mekanisme yang digunakan sistem DNS untuk mengelola namespace DNS. Zona dianggap sebagai titik delegasi dalam hierarki namespace DNS. Titik delegasi ditandai dengan satu atau lebih catatan NS di zona induk. Pada contoh di atas, zona NKU berisi referensi ke cs-ns1 dan cs-ns2 sehingga zona NKU dapat mendelegasikan otoritas zona CS ke

server nama DNS di subdomain cs. Saat klien mengkueri server nama DNS domain nku.edu tentang host di subdomain cs.nku.edu miliknya, server nama DNS domain nku.edu meneruskan kueri ke server nama DNS untuk subdomain cs. Zona dan domain adalah sama jika tidak ada subdomain di bawah domain, dalam hal ini zona berisi semua informasi DNS untuk domain tersebut.



Gambar 7.15 Contoh zona.

File zona menjelaskan zona DNS. File ini adalah file teks yang terdiri dari komentar, arahan, dan catatan sumber daya. Berikut ini adalah file zona yang ditentukan untuk zona CS pada Gambar 7.15.

```
; zone file for cs.nku.edu
$TTL 3600 ; 1 hour default TTL for zone
$ORIGIN cs.nku.edu.
@       IN      SOA    ns1.cs.nku.edu. root.cs.nku.edu. (
        2014071201     ; serial number
        10800          ; Refresh after 3 hours
        3600           ; Retry after 1 hour
        604800         ; Expire after 1 week
        300            ; Negative Response TTL
)
; DNS Servers
                IN        NS       ns1.cs.nku.edu.
                IN        NS       ns2.cs.nku.edu.

; Mail Servers
               IN        MX 10     mx.cs.nku.edu.
               IN        MX 20     mail.cs.nku.edu.

; Host Names
                IN        A        192.168.1.1
localhost       IN        A        127.0.0.1
ns1             IN        A        192.168.1.2
```

```
ns2             IN      A       192.168.1.3
mx              IN      A       192.168.1.4
mail            IN      A       192.168.1.5
fs              IN      A       192.168.1.6
; Aliases
wwwt            IN      CNAME   cs.nku.edu.
```

Komentar dimulai dengan “;” dan diasumsikan berlanjut sampai akhir baris saja. Mulai baris berikutnya dengan ";" jika baris tersebut merupakan kelanjutan dari komentar yang sama (yaitu, jika komentar berlanjut melebihi satu baris). Komentar dapat menempati seluruh baris atau sebagian baris, seperti yang ditunjukkan pada file zona di atas. Direktif dimulai dengan “$” sehingga, misalnya, direktif untuk menentukan TTL default untuk record di zona adalah $TTL, seperti yang ditunjukkan pada contoh di atas. Jika tidak ada TTL yang ditentukan untuk catatan sumber daya tertentu, nilai TTL default ini akan digunakan. $ORIGIN mendefinisikan FQDN untuk zona ini. FQDN akan selalu diakhiri dengan titik. Misalnya, domain cs.nku.com bukan entri FQDN, sedangkan cs.nku.edu. adalah entri FQDN. Titik terakhir dalam entri FQDN mewakili server nama root. Saat file zona sedang diproses, simbol @ diganti dengan nilai $ORIGIN dan nilai $ORIGIN ditambahkan ke entri non-FQDN di file zona. $ORIGIN harus ada sebelum resource record (RR) apa pun.

Periksa catatan zona di atas. Kita melihat TTL default zona, diikuti dengan arahan $ORIGIN. Selanjutnya, kami menemukan catatan sumber daya. Catatan sumber daya pertama adalah catatan SOA (SOA adalah singkatan dari Start of Authority). Catatan SOA diperlukan untuk file zona dan menentukan properti utama dan karakteristik zona. File zona hanya akan memiliki satu catatan SOA. Format generik dari catatan SOA adalah sebagai berikut:

```
Name      TTL   Class    Type    NameServer     EmailAddress (
            SerialNumber
            Refresh
            Retry
            Expire
            Minimum
)
```

Mari kita lihat baris pertama catatan SOA dalam contoh ini. Bidang Nama menentukan nama zona. Penggunaan “@” di sini menunjukkan bahwa nama tersebut sama dengan nilai $ORIGIN. Jadi, nama record SOA ini adalah cs.nku.edu. Entri berikutnya adalah TTL. Nilai legal TTL adalah bilangan bulat antara 0 dan 2147483647. TTL 0 berarti catatan tidak boleh di-cache. Dalam contoh ini, tidak ada TTL yang dicantumkan. Ingatlah bahwa jika RR tidak memiliki TTL, maka TTL default zona akan digunakan. TTL default zona (seperti yang tercantum dua baris sebelumnya) adalah 3600 (1 jam). Bidang Kelas mendefinisikan kelas RR, yang hampir selalu IN (kelas lain adalah untuk jaringan eksperimental). Lihat RFC 2929 jika Anda tertarik mempelajari tentang kelas lain yang tersedia. Bidang Jenis adalah SOA untuk menunjukkan jenis RR ini. Selanjutnya, kita melihat nama server nama untuk zona ini. Ini disediakan sebagai FQDN dan, dalam hal ini, disebut ns1.cs.nku.edu. (sekali lagi dengan periode akhir). Terakhir, alamat email untuk administrator zona disediakan. Alamat email dalam contoh ini adalah root.cs.nku.edu., yang tidak seperti alamat email biasa, tidak memiliki tanda @ karena tanda @

memiliki arti yang berbeda. @ tersirat ada sebelum nama domain di alamat email ini, jadi sebenarnya root.cs.nku.edu ditafsirkan sebagai root@cs.nku.edu.

Sisa dari SOA memberikan lima nilai integer dalam tanda kurung. Bilangan bulat pertama adalah nomor seri rekaman. Ini menentukan nilai yang digunakan untuk melacak apakah file zona telah dimodifikasi. Kami menggunakan ini untuk memberi tahu server nama DNS budak apakah mereka perlu memodifikasi file zona mereka sendiri atau dapat terus menggunakan yang sekarang. Biasanya, saat administrator memodifikasi file zona, dia memperbarui nomor seri. Format umumnya adalah menentukan tanggal dan waktu modifikasi sebagai nomor urut, jadi formatnya adalah yyyymmddss, di mana yyyy adalah tahun, mm adalah bulan, dd adalah hari, dan ss adalah nomor urut yang dimulai dari 01 dan bertambah setelah setiap modifikasi. Dalam contoh ini, 2014071201 menunjukkan bahwa file zona terakhir diubah pada 12 Juli 2014, dan ini adalah pembaruan pertama untuk hari itu. Nomor seri penting untuk transfer zona, yang akan dibahas nanti.

Nilai lainnya berkaitan dengan komunikasi antara server nama DNS master dan budak. Semua nilai ini adalah satuan waktu dalam detik, kecuali ditentukan lain. Kolom Refresh menentukan waktu ketika setiap server DNS budak untuk zona tersebut perlu mengambil catatan SOA dari server DNS master untuk zona tersebut. Kolom Retry menentukan interval yang harus dilewati antara upaya oleh seorang budak untuk menghubungi server nama DNS master ketika waktu penyegaran telah berakhir. Bidang Kedaluwarsa menentukan jumlah waktu yang valid untuk file zona saat ini. Saat kedaluwarsa, jika slave tidak dapat me-refresh file zonanya dari server nama DNS master, respons apa pun yang dikirim oleh slave akan ditandai sebagai tidak otoritatif. Nilai akhir, minimum, dikenal sebagai waktu caching negatif. Caching server nama DNS, seperti server DNS lokal, diperlukan untuk mengingat jawaban negatif. Jika server nama DNS otoritatif mengembalikan respons negatif ke server nama DNS lokal, respons akan di-cache oleh server nama DNS lokal selama waktu minimum ini.

Apa itu tanggapan negatif? Ini muncul ketika kueri mencari informasi tentang host atau domain atau subdomain dari zona yang tidak ada. Tanggapan untuk permintaan tersebut adalah NXDOMAIN (nama domain tidak ada). Misalnya, mengeluarkan perintah nslookup abc.nku.edu menghasilkan respons negatif berikut:

```
Server:         Unknown
Address:    192.168.1.1

*** Unknown can't find abc.nku.edu: Non-existent domain
```

Perintah nslookup meminta resolusi nama host yang tidak ada. Tidak ada host abc di database DNS nku.edu. Server nama NKU merespons dengan nilai NXDOMAIN. Entri ini, seperti resolusi nama yang berhasil, di-cache. Alasan untuk meng-cache entri negatif adalah untuk mengurangi waktu respons saat permintaan lain ke host yang sama dicoba. Sekarang, server DNS lokal mengetahui bahwa host tidak ada dan tidak akan mengganggu server nama NKU lagi (hingga batas waktu berlalu). Untuk catatan SOA contoh kami, waktu minimum diatur ke 300 detik. Ini mungkin terdengar seperti waktu yang singkat, tetapi pada kenyataannya, menyimpan respons negatif dalam cache untuk waktu yang lama dapat menyebabkan masalah yang signifikan karena entri yang kami harapkan akan tetap tidak tersedia bagi kami

meskipun server nama otoritatif telah dimodifikasi untuk sekarang menyertakan alamat host tersebut.

Dari empat kali yang tercantum dalam catatan SOA (segarkan, coba lagi, kedaluwarsa, dan minimum), standarnya adalah menafsirkan setiap detik. Namun, pertimbangkan kedaluwarsa, yang mungkin ingin kami atur selama seminggu. Seminggu dalam hitungan detik adalah angka yang besar (604800). Jadi, kita juga diperbolehkan mengganti satuan waktu default dengan menambahkan salah satu dari m (atau M) untuk menit, h (atau H) untuk jam, d (atau D) untuk hari, dan w (atau W) untuk minggu ke akhir dari setiap waktu ini. Anda juga dapat menggabungkan singkatan ini. Sebagai contoh, di bawah ini adalah entri dengan refresh rate setiap satu setengah hari (1 hari 12 jam), retry rate setiap 1 jam 10 menit, waktu kedaluwarsa 1 minggu 2 hari, dan waktu minimum 6 jam 30 menit. Ingat bahwa nomor pertama adalah nomor seri.

```
… SOA … (
      2016012701
      1d12h
      1h10m
      1w2d
      6h30m
)
```

Mari kita periksa sisa file zona yang terdaftar sebelumnya dengan melihat catatan sumber daya yang ditentukan. Berikutnya dalam file adalah entri untuk server nama (NS) kami. Catatan NS menentukan server DNS mana yang berwenang untuk zona ini. Dalam contoh ini, ada dua server nama DNS otoritatif: ns1.cs.nku.edu. dan ns2.cs.nku.edu. (sekali lagi perhatikan bahwa nama diakhiri dengan titik). Anda mungkin memperhatikan bahwa tidak seperti catatan sumber daya selanjutnya, hanya ada tiga bidang yang digunakan untuk catatan NS. Dua bidang pertama (nama dan TTL) telah dihilangkan. Nama adalah nama domain, yang telah ditetapkan sebagai $ORIGIN. TTL, karena dihilangkan, akan default ke nilai $TTL.

Mengikuti catatan NS kami adalah dua catatan MX (Mail eXchange) untuk menentukan server email untuk domain tersebut. Karena ada dua server email, masing-masing bernama mx dan mail, kami juga menentukan preferensi. Yang pertama, mx, memiliki nilai yang lebih rendah dan demikian juga server email utama kami. Server bernama mail akan berfungsi sebagai server sekunder atau cadangan, jika mx tidak merespons. Perhatikan bahwa jika Anda hanya memiliki satu server email, Anda tetap harus menentukan nilai preferensi.

Kumpulan entri terakhir menentukan host lain dari domain kami. Ini adalah mesin lain seperti workstation individu, mainframe, dan komputer desktop. Entri ini terutama terdiri dari catatan A, yang menentukan untuk setiap item bernama alamat IPv4-nya. Perhatikan bahwa entri pertama tidak mencantumkan nama. Dalam hal ini, defaultnya adalah nama host, seperti yang dinyatakan dalam direktif $ORIGIN. Anda mungkin juga memperhatikan bahwa selain fs (server file), kami juga menentukan alamat IP dari semua host lain yang memiliki catatan sumber daya sebelumnya (server nama dan server email kami). Kami memiliki entri lain yang disebut localhost, yang menentukan alamat IP localhost. Data A diikuti oleh satu data CNAME. Kami ulangi entri di sini, karena secara sintaksis, sulit untuk dipahami.

```
www         IN      CNAME     @
```

Catatan ini mengatakan bahwa www adalah alias untuk host yang bernama @. Ingatlah bahwa @ digunakan untuk menunjukkan nama domain kita, yang telah ditentukan sebelumnya, menggunakan direktif $ORIGIN. Oleh karena itu, nama host www.cs.nku.edu adalah alias untuk nama cs.nku.edu, yang didefinisikan memiliki alamat IP 192.168.1.1.

File zona yang baru saja kita bahas disebut file zona DNS maju. File zona DNS maju digunakan untuk server DNS maju, yang menerjemahkan nama host menjadi alamat IP. Sebagian besar permintaan DNS melakukan pencarian DNS ke depan. Namun, kami sebelumnya telah menyebutkan pencarian balik DNS untuk mendukung langkah-langkah keamanan. Contoh pencarian DNS terbalik ditunjukkan pada Gambar 7.16.

Untuk pencarian DNS terbalik, kami memerlukan catatan pointer, ditentukan menggunakan PTR. Kami menempatkan entri ini ke dalam file zona terpisah yang dikenal sebagai file zona DNS terbalik. Dalam file seperti itu, alamat kami dibalik. Artinya, kami menentukan alamat IPv4 dalam urutan yang berlawanan. Kami juga menambahkan in-addr.arpa. ke nama-nama. Berikut ini adalah file zona DNS terbalik untuk file zona DNS maju kami sebelumnya. Sebagian besar entri adalah sama atau serupa.

```
C:\Users\harvey\nslookup  www.nku.edu
Server:  Unknown
Address: 192.168.1.1

Non-authoritative answer:
Name:  www.nku.edu
Address: 192.122.237.7

C:\Users\harvey\nslookup  192.122.237.7
Server:  Unknown
Address: 192.168.1.1

Name:  www.nku.edu
Address: 192.122.237.7
```

Gambar 7.16 Membalikkan pencarian DNS.

```
$TTL 3600
1.168.192.in-addr.arpa. IN SOA ns1.cs.nku.edu.
     root.cs.nku.edu. (
           2006051501      ; Serial
           10800           ; Refresh
           3600            ; Retry
           604800          ; Expire
           300 )           ; Negative Response TTL

         IN     NS     ns1.cs.nku.edu.
         IN     NS     ns2.cs.nku.edu.

1        IN     PTR    cs.nku.edu.
2        IN     PTR    ns1.cs.nku.edu.
3        IN     PTR    ns2.cs.nku.edu.
4        IN     PTR    mx.cs.nku.edu.
5        IN     PTR    mail.cs.nku.edu.
6        IN     PTR    fs.cs.nku.edu.
```

Catatan PTR, seperti yang tercantum sebelumnya, dimulai dengan alamat IP host. Dalam hal ini, alamat jaringan terdiri dari tiga oktet pertama dari alamat IP, sehingga hanya oktet keempat yang merupakan alamat perangkat host. Oleh karena itu, masing-masing host terdaftar di PTR dengan oktet tunggal ini. Setiap catatan berisi kelas (IN) dan jenis catatan (PTR), diikuti dengan nilainya. Sedangkan pada record A, nilainya adalah alamat IP dari hostname, pada record PTR, nilainya adalah hostname dari alamat IP. Dengan demikian, pemetaan IP terbalik mengubah alamat IP menjadi nama host.

Kami telah menyiapkan file zona maju dan file zona mundur untuk mendukung pencarian DNS maju dan mundur untuk zona cs kami. Namun, kami belum selesai menyiapkan file kami. Ketika server DNS untuk suatu zona menerima kueri untuk host di luar zonanya, ia tidak akan mengetahui alamat IP dari host yang diminta. Server nama DNS perlu meneruskan permintaan ke server nama DNS lain untuk menjawab kueri. Server nama DNS untuk zona tersebut perlu menemukan alamat IP dari server nama DNS lain yang dapat digunakan untuk meneruskan kueri. Bagaimana cara menghubungi server nama lain? Itu, seperti server nama DNS lokal, harus naik ke hierarki DNS untuk menghubungi server nama DNS root.

Mari kita perhatikan sebuah contoh. Host ns1.cs.nku.edu adalah server nama otoritatif untuk zona cs.nku.edu. Itu menerima kueri untuk host di zona nku.edu. Ingatlah bahwa zona ini berada di luar subdomain cs, dan oleh karena itu, alamat IP host tidak akan disimpan di database DNS ns1. Server nama ns1 harus menemukan server DNS otoritatif untuk zona nku.edu untuk menjawab kueri. Ia melakukannya dengan terlebih dahulu menanyakan server root. Server root merujuk ns1 ke server otoritatif untuk zona .edu. Kemudian, ns1 secara berurutan menanyakan server otoritatif untuk zona .edu, yang mencari server nama domain nku.edu dan merujuk ns1 ke server nama tersebut. Ini memungkinkan ns1 mengkueri server otoritatif nku.edu untuk host yang ditentukan dalam kueri yang ada di zona nku.edu (tetapi bukan zona cs.nku.edu). Akhirnya, server nama nku.edu mengembalikan alamat IP dari host yang diminta.

Dari contoh ini, Anda dapat melihat bahwa server nama DNS otoritatif untuk suatu zona perlu mengetahui tidak hanya informasi zona lokalnya (untuk pencarian maju dan mungkin mundur) tetapi juga informasi zona server root. File petunjuk akar yang dikelola oleh Internet Assigned Numbers Authority (IANA) menyediakan informasi zona akar. File tersebut berisi daftar nama host dan alamat IP dari server DNS root. Kutipan dari file petunjuk root ditampilkan sebagai berikut:

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache . <file>"
```

```
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.cache
;           on server           FTP.INTERNIC.NET
;           -OR-                RS.INTERNIC.NET
;
;           last update: June 2, 2014
;           related version of root zone:     2014060201
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:BA3E::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     192.228.79.201
B.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:84::B
;
; FORMERLY C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
C.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:2::C
;
; FORMERLY TERP.UMD.EDU
;
...
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
M.ROOT-SERVERS.NET.      3600000      AAAA  2001:DC3::35
; End of File
```

Server nama DNS root dapat berubah seiring waktu. Anda dapat mengunduh file petunjuk root versi terbaru di ftp://rs.internic.net/domain/named.root dan menggunakannya di file zona Anda. Kami akan membahas cara menggunakan file petunjuk root di file zona di Bab 6 saat kami menjelajahi server nama BIND DNS.

Setiap zona biasanya dilayani oleh setidaknya dua server nama DNS: server DNS master (primer) dan satu atau lebih server nama DNS slave (sekunder). Budak tersedia untuk menyediakan redundansi untuk menghindari satu titik kegagalan dan untuk mendukung penyeimbangan beban untuk mengurangi kemungkinan latensi jika server nama resmi kelebihan beban dengan permintaan. Perbedaan antara server nama DNS master dan server nama DNS budak adalah bahwa server master berisi versi asli dari file data zona. Server budak menyimpan salinan file data zona.

Ketika perubahan dilakukan oleh administrasi (sub) domain, dia akan membuat perubahan tersebut ke file zona yang disimpan di server nama DNS master. Dalam proses memperbarui file zona, catatan SOA harus diperbarui dengan mengubah nomor seri.

Perubahan lain mungkin menambahkan host baru dan catatan A, AAAA, PTR, dan/atau CNAME mereka sendiri, menambahkan server nama tambahan (NS) atau server email (MX), mengubah catatan yang ada dengan mengubah alamat IP dan/atau nama host mereka , dan hapus entri jika host tersebut tidak lagi tersedia.

Proses replikasi file zona dari server DNS master ke server DNS budak disebut transfer zona. Server DNS slave dapat melakukan sinkronisasi dengan server DNS master melalui transfer zona dengan dua cara. Salah satu caranya adalah server nama DNS master mengirimkan pemberitahuan ke server nama DNS budak tentang perubahan pada file zona. Server nama DNS budak memulai proses transfer zona saat pemberitahuan diterima.

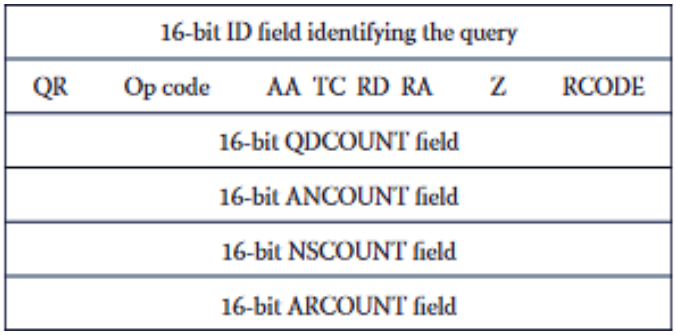
Cara lain adalah untuk server nama DNS budak secara berkala meminta server nama DNS master untuk menentukan apakah nomor seri untuk zona tersebut telah meningkat. Jika ya, server nama DNS budak memulai proses transfer zona. Polling periodik oleh server nama DNS budak dikendalikan oleh tiga nilai dalam SOA mengikuti nomor seri. Angka-angka ini adalah kecepatan penyegaran, kecepatan coba lagi, dan nilai kedaluwarsa. Server nama DNS budak menunggu interval penyegaran sebelum memeriksa dengan server nama DNS master. Jika ditemukan bahwa file zona master memiliki nomor seri yang sama dengan file zona saat ini, tidak ada tindakan lebih lanjut yang diambil. Jika nomor seri bertambah, maka transfer zona dimulai. Jika karena beberapa alasan master tidak merespon atau beberapa masalah lain muncul sehingga slave tidak dapat menyelesaikan pemeriksaan, slave akan menunggu jumlah waktu yang ditunjukkan oleh tingkat coba ulang. Jika budak terus mengalami masalah dalam menghubungi server nama master, file zonanya akan tetap seperti apa adanya. Namun, setelah file zona mencapai interval kedaluwarsa, server budak harus menganggap file zonanya telah kedaluwarsa dan oleh karena itu akan menanggapi pertanyaan apa pun untuk informasi zona sebagai jawaban yang tidak otoritatif. Pada transfer zona yang berhasil, budak mengganti file zonanya saat ini dengan file yang ditransfer dari master. Perhatikan bahwa setelah berhasil menyelesaikan transfer cek atau zona, budak me-reset penghitung penyegarannya.

Server nama DNS biasanya mendukung dua jenis replikasi file zona: transfer zona penuh dan transfer zona inkremental. Transfer zona penuh menyalin seluruh file zona dari master ke budak. Transfer zona inkremental hanya mereplikasi catatan yang telah dimodifikasi sejak transfer zona terakhir.

7.2 PROTOKOL SISTEM NAMA DOMAIN

Pesan DNS dapat menggunakan User Datagram Protocol (UDP) atau datagram TCP. Dengan demikian, kueri DNS dan responsnya biasanya dikirim menggunakan koneksi UDP karena paket UDP berukuran lebih kecil dan memiliki waktu transmisi yang lebih cepat, sehingga memungkinkan penyelesaian DNS memiliki respons yang lebih cepat. Resolusi nama dapat dilengkapi dengan dua paket UDP: satu paket kueri dan satu paket respons. Namun, ukuran paket UDP tidak boleh melebihi 512 byte. Selain kueri resolusi nama, server DNS perlu bertukar pesan lain yang lebih besar. Ini termasuk, misalnya, file zona dalam transfer zona. Jika pesan DNS lebih besar dari 512 byte, maka diperlukan koneksi TCP. File zona biasanya ditransfer melalui koneksi TCP.

Kueri dan respons DNS menggunakan format pesan yang sama. Pertama, ada bagian header. Bagian ini wajib diisi, dan berisi 13 field, seperti yang ditunjukkan pada Gambar 7.17. Bidang ID secara unik mengidentifikasi pesan ini. Nilai ditetapkan oleh klien DNS. Tanggapan apa pun harus berisi ID yang sama. Bidang kedua dari header terutama terdiri dari flag 1-bit. Bendera QR menentukan apakah ini adalah pesan kueri (0) atau respons (1). Bendera AA menunjukkan apakah pesan ini berisi respons otoritatif (diatur oleh server nama DNS) atau respons tidak otoritatif. Bendera TC menunjukkan jika pesan ini harus dipotong (nilai 1) karena kurangnya panjang paket UDP. Bendera RD digunakan untuk menunjukkan apakah ini adalah kueri rekursif (1) atau bukan (0). Terakhir, flag RA, yang ditangani oleh server nama DNS, menentukan apakah server ini mendukung kueri rekursif (1) atau tidak (0).



Gambar 7.17 Format untuk header kueri DNS.

Selain flag, kolom kedua berisi kode op 4-bit. Nilai ini menentukan jenis kueri. Saat ini, ini dapat diatur ke salah satu dari tiga nilai: 0000 untuk kueri standar, 0001 untuk kueri terbalik, dan 0010 untuk permintaan status server. Nilai lainnya dicadangkan untuk penggunaan di masa mendatang. Dua nilai tambahan di bidang kedua header ini adalah Z, bidang 3-bit yang nilainya harus semua 0 karena bidang ini dimaksudkan untuk penggunaan di masa mendatang, dan RCODE, bidang 4-bit yang digunakan untuk menunjukkan kesalahan apa pun yang nilainya tercantum pada Tabel 7.5. Kombinasi 4-bit yang tidak tercantum dicadangkan untuk penggunaan di masa mendatang.

Empat bidang header yang tersisa memberikan nilai 16-bit yang mewakili angka yang terkait dengan kueri atau respons. QDCOUNT adalah jumlah entri kueri di bagian pertanyaan. ANCOUNT adalah jumlah catatan sumber daya yang dikembalikan di bagian jawaban. NSCOUNT adalah jumlah catatan sumber daya server nama di bagian otoritas. Terakhir, ARCOUNT adalah jumlah catatan sumber daya di bagian tambahan.

Sisa dari kueri atau tanggapan terdiri dari empat bagian: bagian pertanyaan, bagian jawaban, bagian otoritas, dan bagian tambahan. Bagian pertanyaan berisi tiga bagian: QNAME, QTYPE, dan QCLASS. QNAME adalah nama domain, dikirim sebagai daftar label. Setiap label dimulai dengan panjangnya dalam byte, diikuti dengan label itu sendiri. Panjang dan label diberikan dalam biner, mewakilinya dalam nilai American Standard Code for Information Interchange (ASCII). Misalnya, www akan ditentukan sebagai 00000011 01110111 01110111 01110111. Entri pertama adalah byte yang menyimpan nomor 3 untuk panjang label, yang terdiri dari 3 byte. Ini diikuti oleh setiap nilai ASCII karakter dalam biner (w adalah

ASCII 119). Di bawah ini, kami mendemonstrasikan QNAME dari www.nku.edu yang ditampilkan dalam notasi heksadesimal daripada biner untuk menghemat ruang.

0 × 03 0 × 777777
0 × 03 0 × 6E6B75
0 × 03 0 × 656475

Tabel 7.5 Kemungkinan Nilai RCODE

Nilai RCODE	Nama Pesan	Arti
0000	NOERROR	Kueri berhasil ditangani
0001	FORMERR	Kesalahan format
0010	SERVFAIL	Server DNS tidak membalas atau menyelesaikan permintaan
0011	NXDOMAIN	Nama domain tidak ada
0100	NOTIMP	Fungsi yang diminta tidak diterapkan di server
0101	REFUSED	Server menolak untuk menjawab pertanyaan
0110	YXDOMAIN	Nama ada tetapi tidak seharusnya
0111	XRRSET	Kumpulan catatan sumber daya ada tetapi seharusnya tidak
1000	NOTAUTH	Server tidak berwenang untuk zona yang diminta
1001	NOTZONE	Nama tidak ditemukan di zona

Perhatikan bahwa setiap bagian dari nama muncul pada baris terpisah. Periode yang harus terjadi di antara mereka tersirat. QTYPE adalah bidang 2-byte yang menunjukkan catatan sumber daya yang diminta. Bidang QCLASS juga merupakan bidang 2-byte, dalam hal ini menunjukkan jenis jaringan (IN, CH, dan seterusnya).

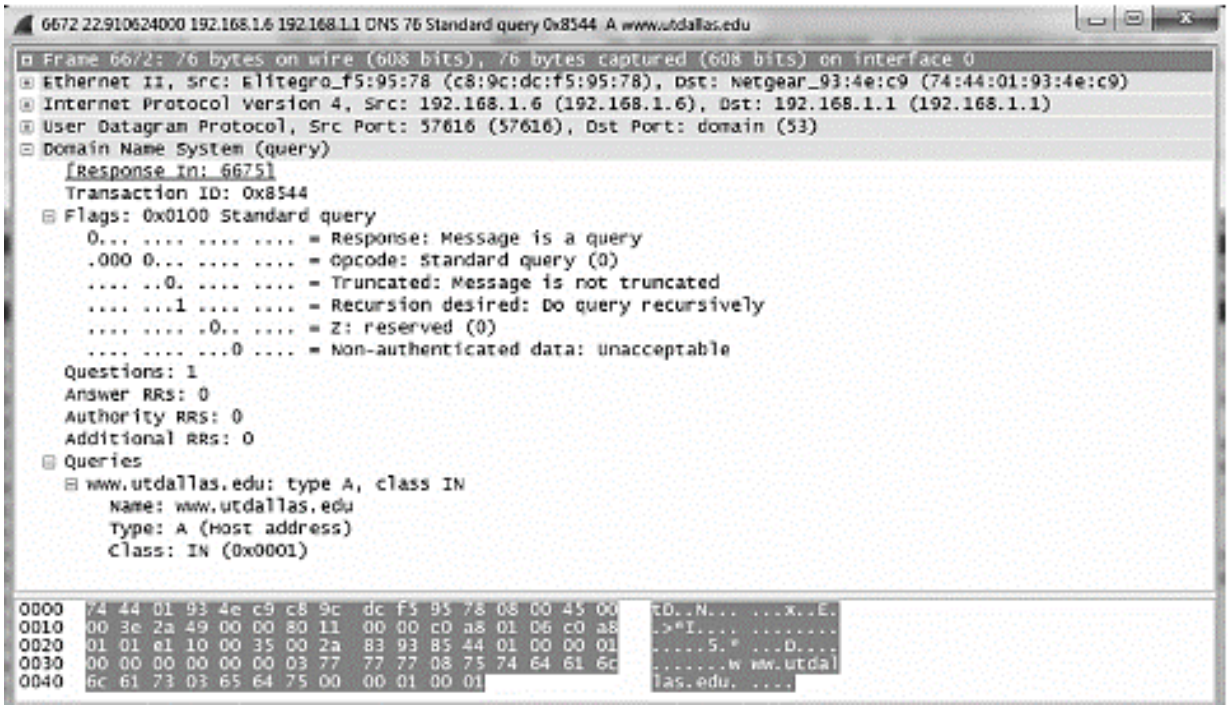
Jawaban, otoritas, dan bagian tambahan menggunakan format yang sama. Format ini ditunjukkan pada Gambar 7.18. Di sini, kita melihat beberapa bidang, masing-masing untuk nama, jenis catatan sumber daya, kelas jaringan, TTL, bidang panjang untuk catatan sumber daya yang dikembalikan, dan terakhir data catatan sumber daya. Semua bidang ini panjangnya 16 bit, kecuali nama, yang merupakan QNAME, seperti yang terlihat di bagian pertanyaan; TTL, yaitu 32 bit; dan data catatan sumber daya, yang memiliki panjang variabel. Panjang QNAME dapat bervariasi, dan jika ruang menjadi masalah (mis., karena pesan dikirim menggunakan UDP), pointer digunakan sebagai pengganti QNAME, di mana pointer merujuk pada kejadian sebelumnya dengan nama yang sama.

Untuk lebih memahami format pesan DNS, mari kita lihat contoh yang ditangkap oleh Wireshark. Kami melihat permintaan DNS, ditunjukkan pada Gambar 7.19, dan respons DNS, ditunjukkan pada Gambar 7.20. Kueri adalah permintaan resolusi alamat untuk nama host www.utdallas.edu, seperti yang ditentukan di bilah alamat browser web. Kueri DNS dikirim ke server nama DNS lokal di NKU (192.168.1.1) dan diteruskan, sebagaimana diperlukan, untuk diselesaikan oleh server nama utdallas.edu. Perhatikan pada Gambar 7.19 bahwa kueri menggunakan protokol UDP. Nomor port tujuan paket kueri UDP adalah 53 (yang merupakan port DNS default). ID kueri adalah 0×8544, dihasilkan oleh klien DNS. Menurut nilai di bidang bendera, ini adalah kueri rekursif standar. Pesan kueri tidak terpotong karena ukuran pesan lebih kecil dari 512 byte.

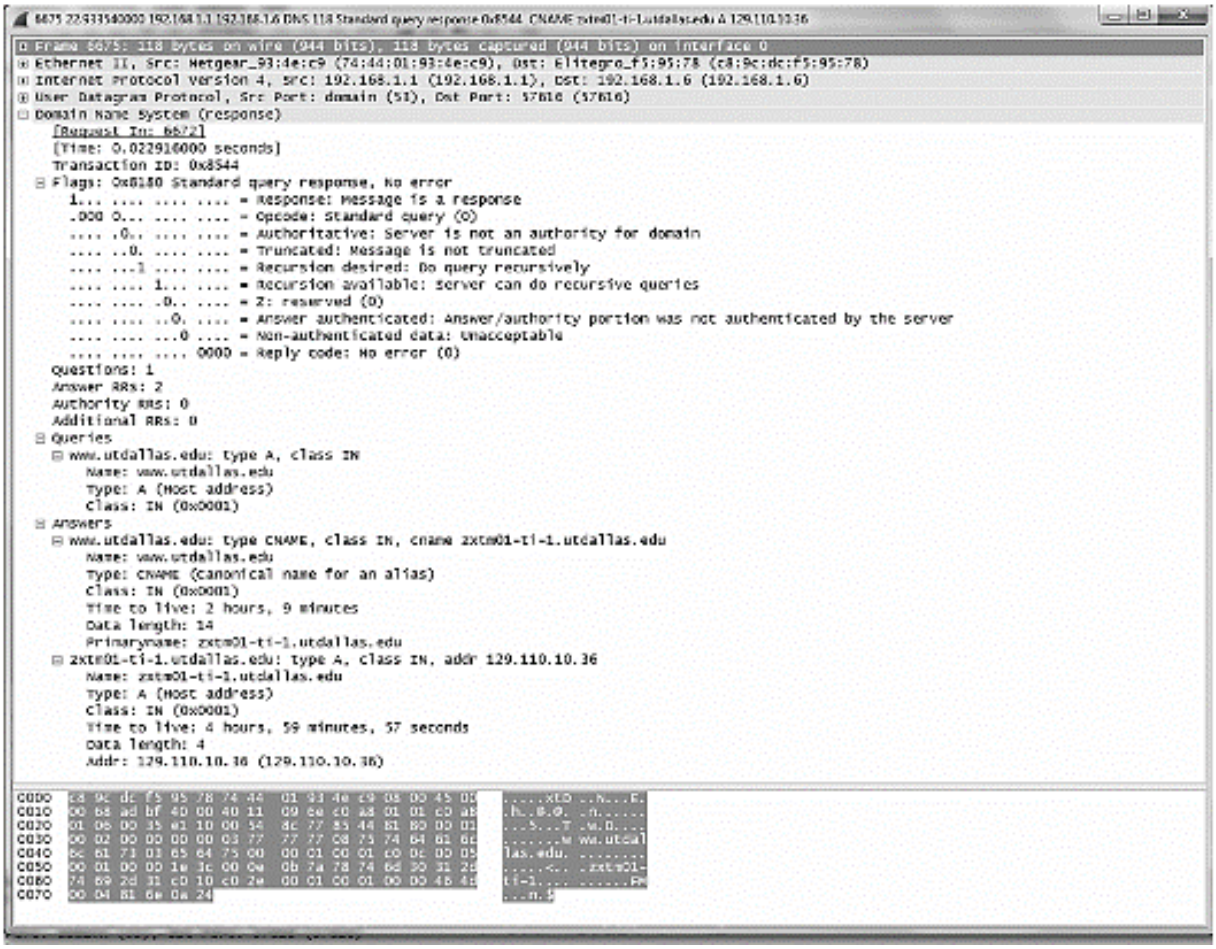
Responnya, seperti yang ditunjukkan pada Gambar 7.20, juga merupakan datagram UDP. Nomor port sumber adalah 53. ID yang sama (0×8544) digunakan dalam pesan tanggapan. Menurut nilai di bidang bendera, ini adalah pesan respons untuk kueri dengan ID 0×8544. Server DNS mendukung kueri rekursif. Jawaban atas pertanyaan menunjukkan bahwa www.utdallas.edu adalah nama alias untuk host zxtm01-ti-1.utdallas.edu, yang memiliki alamat IPv4 129.110.10.36. Catatan sumber daya ini dapat di-cache selama 4 jam, 59 menit, dan 57 detik. Bidang panjang data di bagian jawaban adalah 4 byte (atau 32 bit) karena merupakan alamat IPv4.

Name (QNAME or pointer)
Record type
Network class
Record TTL
Data field length
Resource record data

Gambar 7.18 Format jawaban, wewenang, dan bagian tambahan.



Gambar 7.19 Pesan kueri DNS ditangkap oleh Wireshark.



Gambar 7.20 Respons DNS ditangkap oleh Wireshark.

7.3 KINERJA SISTEM NAMA DOMAIN

Pengguna menginginkan respons Internet yang cepat. DNS menambahkan lapisan komunikasi ke setiap aplikasi Internet, kecuali pengguna sudah mengetahui alamat IP. Jika resolusi alamat diperlukan dan dapat ditangani secara lokal melalui respons yang di-cache sebelumnya, dampaknya minimal. Tanpa respons yang di-cache, total waktu pengguna harus menunggu resolusi alamat terdiri dari beberapa bagian. Ini disebutkan sebagai berikut:

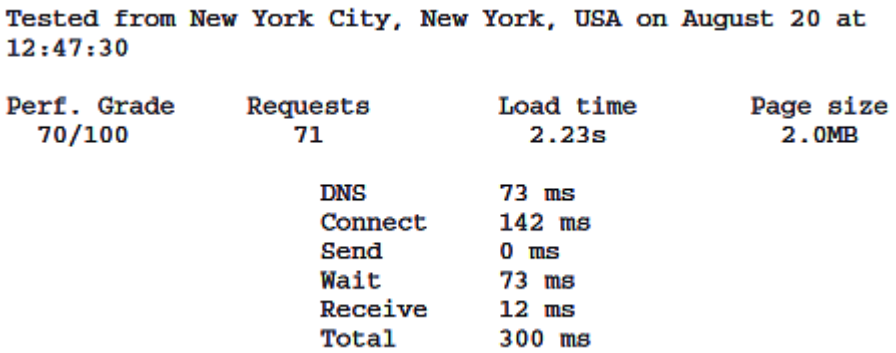
1. Waktu kueri antara penyelesai DNS dan server nama DNS lokal
2. Waktu kueri dari server DNS lokal ke server nama DNS yang sesuai dalam hierarki namespace DNS, yang mencakup hal berikut:
   a. Kueri dari server nama DNS lokal ke domain root dan responsnya
   b. Kueri dari server nama DNS lokal ke server nama DNS TLD yang sesuai dan responsnya
   c. Kueri dari server nama DNS lokal ke server nama DNS otoritatif dari tingkat kedua atau subdomain yang diberikan dan tanggapannya
3. Respons dari server nama DNS lokal ke klien Anda (termasuk, mungkin, waktu untuk meng-cache respons)

Mari kita lihat contoh untuk melihat berapa lama waktu yang dihabiskan permintaan web untuk menunggu pencarian DNS. Kami menggunakan alat uji kecepatan situs web online, Pingdom (http://tools.pingdom.com/fpt/), untuk mengukur waktu respons situs web www.nku.edu. Gambar 7.21 menunjukkan total waktu respon dan perinciannya. DNS

menunjukkan bahwa browser web sedang mencari informasi DNS. Hubungkan berarti bahwa browser web terhubung ke server web. Kirim menunjukkan bahwa browser web mengirim data ke server web. Wait artinya web browser sedang menunggu data dari web server. Terima mewakili browser web yang menerima data dari server web. Total adalah total waktu yang dihabiskan browser dalam memuat halaman web. Dalam contoh ini, waktu pencarian DNS dan total waktu respons www.nku.edu masing-masing adalah 73 dan 300 md, saat kami menguji situs web NKU dari New York City, Amerika Serikat. Waktu pencarian DNS menyumbang sekitar 24,3% (73 ms/300 ms) dari total waktu. Kita dapat melihat bahwa pencarian DNS menambahkan overhead yang cukup besar ke proses penjelajahan web.

Jika tidak ada cache yang terlibat, waktu pencarian DNS bergantung pada jumlah server DNS yang terlibat dalam proses resolusi nama, jarak jaringan antara penyelesai DNS dan server DNS, serta latensi jaringan dan server apa pun. Sebagai perbandingan dengan hasil yang ditunjukkan pada Gambar 7.21, kami menguji situs web NKU dari Amsterdam, Belanda. Waktu pencarian DNS dan total waktu respons www.nku.edu masing-masing adalah 116 dan 502 ms. Server nama otoritatif untuk domain nku.edu terletak di Highland Heights, Kentucky. Waktu pencarian DNS (116 mdtk) dari Amsterdam ke Kentucky lebih lama daripada waktu pencarian DNS (73 mdtk) dari Kota New York ke Kentucky karena jaraknya.

Perlu diingat bahwa halaman web tipikal akan terdiri dari banyak sumber daya yang berpotensi diakses dari banyak server web dan karenanya memerlukan banyak pencarian DNS. Overhead pencarian DNS dalam proses penjelajahan web mungkin jauh lebih besar dari 24% yang kami hitung sebelumnya. Untuk memberikan pengalaman penelusuran pengguna yang lebih baik, kami memerlukan beberapa mekanisme untuk meningkatkan kinerja pencarian DNS. Caching DNS dan prefetching DNS adalah dua teknik yang banyak digunakan untuk meningkatkan kinerja pencarian DNS.


Tested from New York City, New York, USA on August 20 at 12:47:30

Perf. Grade	Requests	Load time	Page size
70/100	71	2.23s	2.0MB

DNS	73 ms
Connect	142 ms
Send	0 ms
Wait	73 ms
Receive	12 ms
Total	300 ms

Gambar 7.21 Menguji kinerja pencarian DNS melalui browser web di seluruh Amerika Serikat.

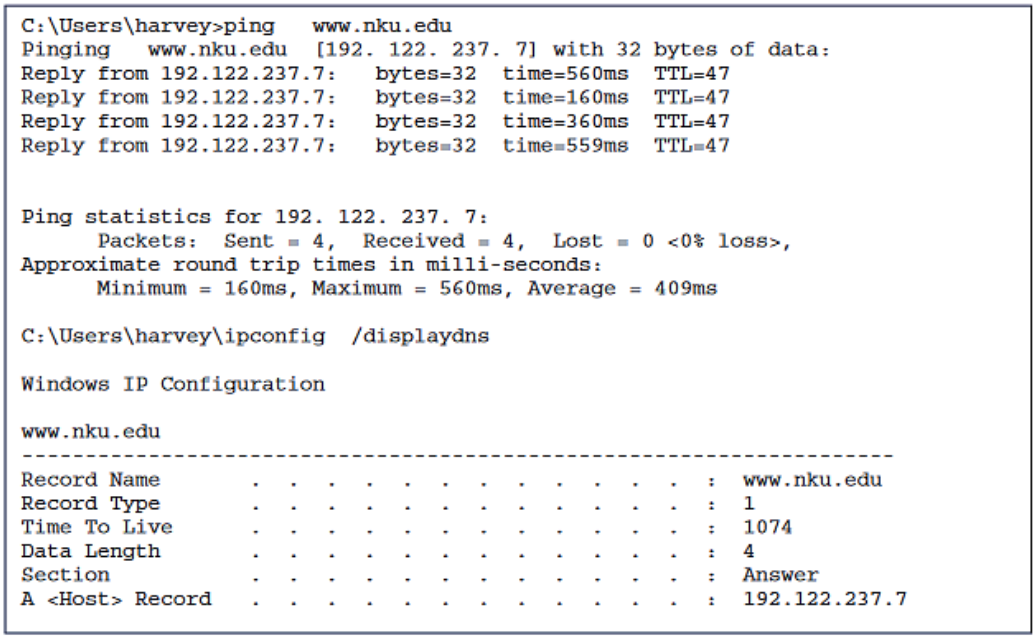
Cachin sistem nama domain sisi klien

Caching DNS adalah mekanisme untuk menyimpan hasil kueri DNS (pemetaan nama domain ke alamat IP) secara lokal untuk jangka waktu tertentu, sehingga klien DNS dapat memiliki respons yang lebih cepat terhadap kueri DNS. Berdasarkan lokasi cache DNS, caching

DNS dapat diklasifikasikan menjadi dua kategori: caching sisi klien dan caching sisi server. Pertama, mari kita pertimbangkan caching sisi klien.

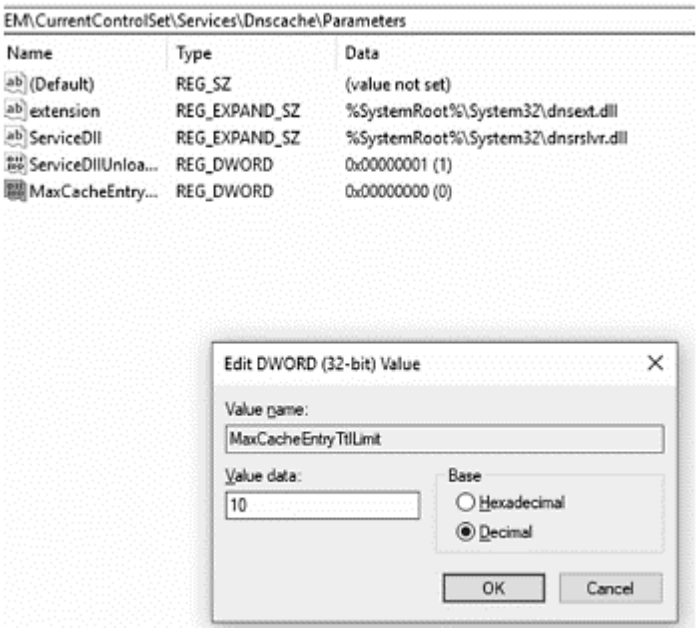
Caching DNS sisi klien dapat ditangani oleh sistem operasi atau aplikasi caching DNS. Caching tingkat OS adalah mekanisme yang dibangun ke dalam sistem operasi untuk mempercepat pencarian DNS yang dilakukan dengan menjalankan aplikasi. Sistem operasi Windows menyediakan cache lokal untuk menyimpan hasil pencarian DNS terbaru untuk akses di masa mendatang. Anda dapat mengeluarkan perintah ipconfig /displaydns dan ipconfig /flushdns untuk menampilkan dan menghapus entri DNS yang di-cache di Windows. Gambar 7.22 menunjukkan contoh entri cache DNS di Windows 7. Pertama, kami menjalankan ipconfig/flushdns untuk menghapus cache DNS. Kami menjalankan ping www.nku.edu, yang mengharuskan penyelesai DNS Windows 7 menyelesaikan alamat www.nku.edu. Setelah menerima respons DNS, informasi tersebut disimpan dalam cache DNS. Dengan menjalankan ipconfig /displaydns, kita melihat bahwa entri yang dihasilkan telah ditambahkan ke cache.

Dari Gambar 7.22, perhatikan bahwa entri TTL dalam cache adalah 1074 (detik). Secara default, nilai TTL dalam respons DNS menentukan kapan entri yang di-cache akan dihapus dari cache DNS. Ingatlah bahwa catatan sumber daya atau SOA file zona menyatakan TTL untuk sebuah entri. Windows menyediakan mekanisme untuk memungkinkan Anda mengurangi nilai TTL dari entri cache DNS. Nilai MaxCacheEntryTtlLimit di registri Windows digunakan untuk menentukan waktu maksimum entri dapat tetap berada di cache DNS Windows. Klien DNS menyetel nilai TTL ke nilai yang lebih kecil antara nilai TTL yang disediakan dalam respons DNS dan nilai MaxCacheEntryTtlLimit yang ditentukan.



```
C:\Users\harvey>ping   www.nku.edu
Pinging   www.nku.edu  [192. 122. 237. 7] with 32 bytes of data:
Reply from 192.122.237.7:   bytes=32  time=560ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=160ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=360ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=559ms  TTL=47


Ping statistics for 192. 122. 237. 7:
      Packets:  Sent = 4,  Received = 4,  Lost = 0 <0% loss>,
Approximate round trip times in milli-seconds:
      Minimum = 160ms, Maximum = 560ms, Average = 409ms

C:\Users\harvey\ipconfig  /displaydns

Windows IP Configuration

www.nku.edu
----------------------------------------------------------------------
Record Name     . . . . . : www.nku.edu
Record Type     . . . . . : 1
Time To Live    . . . . . : 1074
Data Length     . . . . . : 4
Section         . . . . . : Answer
A <Host> Record . . . . . : 192.122.237.7
```

Gambar 7.22 Caching DNS di Windows 7.



Gambar 7.23 Mengubah nilai TTL cache DNS di registri Windows.

Gambar 7.23 menunjukkan cara mengubah nilai MaxCacheEntryTtlLimit di Registri Windows 7. Untuk memodifikasi registri (yang biasanya tidak disarankan), masukkan regedit.exe di kotak pencarian menu tombol mulai untuk menjalankan program Penyunting Registri. Di panel kiri jendela editor registri, terdapat berbagai kategori entri registri yang tersedia untuk diedit, seperti HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, dan HKEY_USERS. Kategori ini tercantum di bawah entri tingkat atas, Komputer. Dari Komputer, luaskan masing-masing HKEY_LOCAL_MACHINE, SYSTEM, CurrentControlSet, Services, Dnscache, dan Parameters. Sekarang, cari entri MaxCacheEntryTtlLimit dan pilih. Dari menu Edit, pilih Modifikasi, dan dari jendela pop-up, masukkan batas waktu maksimum yang baru. Sebagai percobaan, kami mengubah nilainya dari 10 menjadi 4 detik dengan memasukkan 4 ke dalam kotak dan memilih OK di jendela pop-up. Kami kemudian menutup editor registri setelah kami selesai menggunakannya. Catatan: untuk mengubah registri, Anda harus memiliki hak administrator. Anda tidak perlu melakukan perubahan yang kami lakukan. Namun, setelah TTL maksimum, kami memutar ulang perintah ping sebelumnya, diikuti dengan mengeluarkan kembali perintah ipconfig /displaydns. Hasilnya ditunjukkan pada Gambar 7.24. Di sini, kita melihat bahwa nilai TTL untuk entri cache www.nku.edu sekarang adalah 4 detik, karena ini adalah nilai yang lebih kecil daripada nilai yang diberikan oleh server nama DNS otoritatif. Kami menunggu 4 detik dan menjalankan ipconfig /displaydns lagi. Cache kosong karena satu entri www.nku.edu telah kedaluwarsa dan telah dihapus.

Meskipun cache DNS dapat meningkatkan performa, entri DNS yang di-cache juga dapat menyebabkan masalah. Administrator situs web biasanya menggunakan mekanisme penyeimbangan muatan untuk mencapai situs web yang sangat tersedia dan dapat diskalakan. Beberapa server yang secara kolektif memiliki banyak alamat IP digunakan untuk melayani satu situs web. Mari kita jelajahi ide ini dengan sebuah contoh.

Asumsikan bahwa tiga komputer, server1, server2, dan server3, digunakan sebagai server web untuk situs web www.icompany.com. Alamat IP ketiga server masing-masing

adalah 10.10.10.1, 10.10.10.2, dan 10.10.10.3. Klien mengirim permintaan ke situs web dengan menggunakan alias situs web, dan karena ini adalah pertama kalinya klien mengunjungi situs web ini, alias untuk pemetaan alamat tidak ditemukan di cache DNS sisi klien. Klien harus mengirim kueri DNS untuk www.icompany.com ke server nama DNS lokalnya untuk resolusi. Asumsikan lebih lanjut bahwa entri ini tidak di-cache di sana, sehingga kueri dikirim ke server nama DNS otoritatif untuk domain icompany.com. Server nama itu memiliki tiga alamat yang tersedia, tetapi memilih satu alamat untuk dikembalikan, berdasarkan kebijakan penyeimbangan muatan yang telah ditentukan sebelumnya. Asumsikan bahwa alamat server1, 10.10.10.1, dipilih dan dikembalikan ke klien DNS. Saat menerima respons DNS, klien melakukan cache 10.10.10.1 untuk www.icompany.com dalam cache-nya dan mengirimkan permintaan HTTP ke server1. Server1 melayani permintaan, tetapi kadang-kadang, tidak lama kemudian, server1 dimatikan untuk pemeliharaan. Klien mengunjungi www.icompany.com lagi. Entri DNS yang di-cache dari 10.10.10.1 belum kedaluwarsa dan dikembalikan oleh cache. Klien menggunakan alamat IP ini untuk mengirim permintaan HTTP. Permintaan HTTP ini gagal karena server tersebut tidak merespons. Seandainya permintaan resolusi alamat dikirim ke server nama DNS resmi, itu akan mengirimkan alamat IP dari salah satu dari dua server yang tersedia. Ini akan mencegah respon yang salah diterima oleh klien. Dalam contoh ini, caching DNS membuat alamat IP basi, karena alamat IP tidak lagi valid.

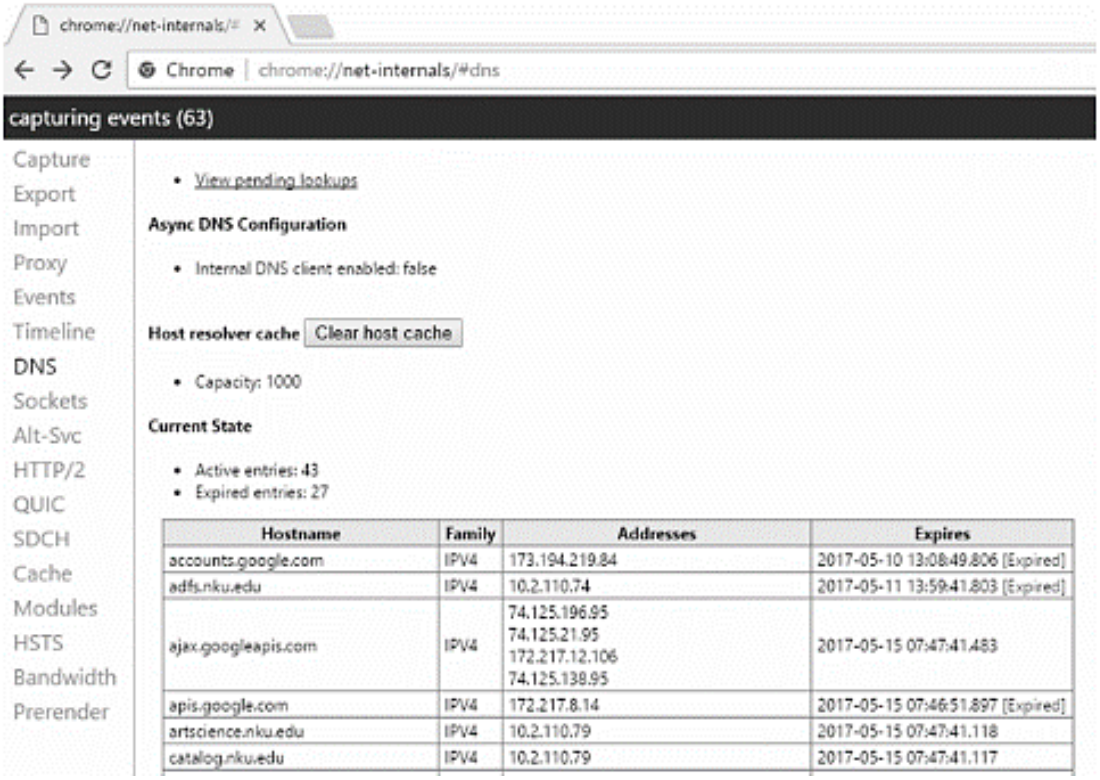
```
C:\windows\system32> ipconfig  /displaydns
Windows  IP  Configuration
C:\windows\system32>ping   www.nku.edu
Pinging   www.nku.edu  [192. 122. 237. 7] with 32 bytes of data:
Reply from 192.122.237.7:   bytes=32  time=595ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=194ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=393ms  TTL=47
Reply from 192.122.237.7:   bytes=32  time=228ms  TTL=47

Ping statistics for 192.122.237.7:
      Packets:  Sent = 4,  Received = 4,  Lost = 0 <0% loss>,
Approximate round trip times in milli-seconds:
      Minimum = 194ms, Maximum = 595ms, Average = 352ms
C:\windows\system32\ipconfig  /displaydns
Windows IP Configuration

www.nku.edu
----------------------------------------------------------------------
Record Name  . . . . . . . . . . . . . . :  www.nku.edu
Record Type  . . . . . . . . . . . . . . :  1
Time To Live . . . . . . . . . . . . . . :  1
Data Length  . . . . . . . . . . . . . . :  4
Section      . . . . . . . . . . . . . . :  Answer
A <Host> Record  . . . . . . . . . . . . : 192.122.237.7

C:\windows\system32\ipconfig  /displaydns
Windows IP Configuration
```

Gambar 7.24 Menjalankan kembali perintah ping dan ipconfig.

Bagaimana kita bisa memecahkan masalah ini? Dari apa yang baru saja kita pelajari, kita dapat menurunkan nilai TTL dari entri DNS yang di-cache ke nilai yang kecil. Solusi lain untuk masalah ini adalah menonaktifkan cache DNS sisi klien sama sekali. Di Linux, nscd dan

dnsmasq adalah layanan (disebut daemon di Linux) yang dapat menangani caching tingkat sistem operasi. Kita dapat memulai dan menghentikannya dengan menggunakan perintah service seperti pada service dnsmasq stop (di Red Hat 6 atau sebelumnya). Kami dapat menonaktifkan layanan cache DNS lokal Windows dengan mengeluarkan perintah net stop dnscache (kami dapat memulai ulang dengan mengeluarkan perintah yang sama tetapi mengubah stop untuk memulai). Seperti halnya mengubah registri, Anda harus menjalankan perintah aktifkan/nonaktifkan dengan hak akses administratif. Perhatikan bahwa jendela mengatakan Administrator Command Prompt bukan hanya Command Prompt. Pesan yang dihasilkan dari perintah ini adalah sebagai berikut:

```
The DNS Client service is stopping.
The DNS Client service was stopped successfully.
The DNS Client service is starting.
The DNS Client service was started successfully.
```



Gambar 7.25 Memeriksa cache browser Google Chrome.

Menurunkan nilai TTL hanya mengurangi kemungkinan penggunaan alamat IP lama sementara juga berpotensi meningkatkan jumlah permintaan resolusi DNS yang tidak dapat dipenuhi oleh cache lokal. Menonaktifkan cache DNS sisi klien menyelesaikan masalah pertama (alamat basi) tetapi menghilangkan semua kemungkinan bahwa klien sendiri dapat menyelesaikan permintaan apa pun. Tidak ada solusi yang sangat menarik.

Pengembang perangkat lunak sering merasa bahwa caching DNS bawaan dalam aplikasi memberikan kinerja yang lebih baik, karena cache dapat dioptimalkan agar sesuai dengan aplikasi itu sendiri. Pendekatan ini lebih menarik karena bentuk khusus aplikasi dari cache DNS disetel untuk mengurangi dampak masalah seperti entri basi, sambil tetap

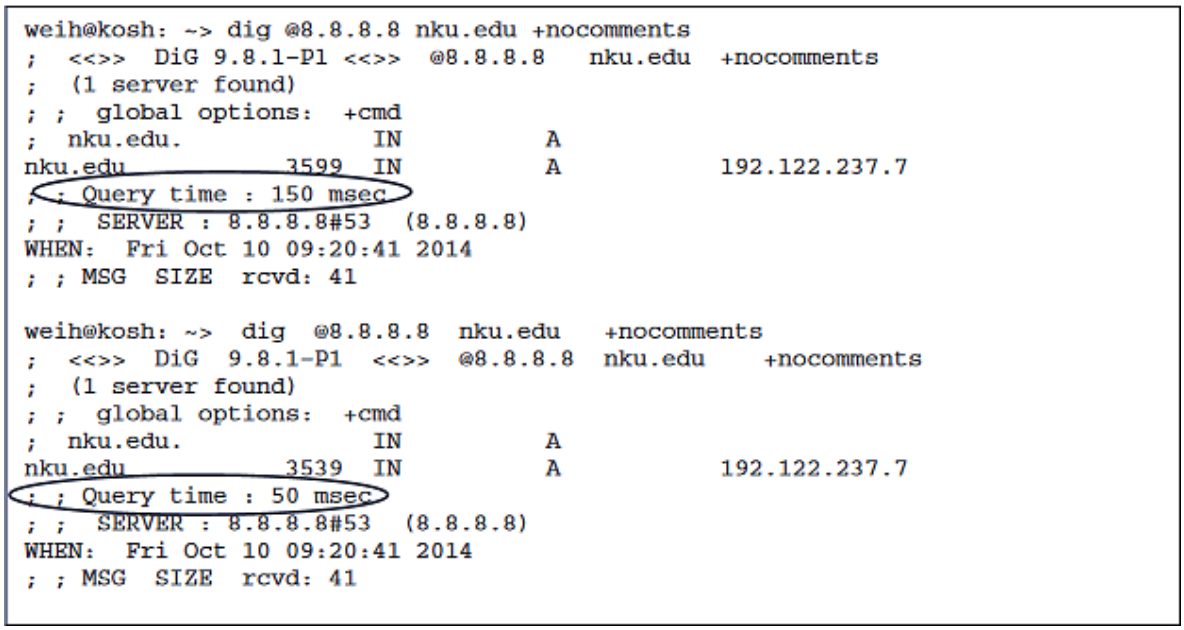
menyediakan cache DNS untuk mengurangi jumlah permintaan DNS yang dilakukan melalui Internet. Cache DNS disediakan oleh banyak aplikasi, termasuk beberapa browser web seperti Google Chrome, Mozilla Firefox, dan Microsoft Internet Explorer. Mari kita jelajahi cache DNS untuk Google Chrome.

Gambar 7.25 menunjukkan contoh caching DNS tingkat aplikasi. Dalam contoh ini, kami menggunakan browser Google Chrome untuk mengunjungi situs web NKU. Kemudian, kami mengunjungi URL chrome://net- internals/#dns, yang menyebabkan entri DNS yang di-cache oleh browser kami ditampilkan. Anda dapat melihat pada gambar daftar nama domain yang disimpan browser dan alamat IP terkait dalam sebuah tabel. Tabel dapat menyimpan hingga 1000 entri. Entri www.nku.edu tidak muncul karena kami hanya melihat beberapa entri pertama. Untuk menghapus entri yang di-cache, pilih tombol "Hapus cache host". Menariknya, sebagian besar cache DNS browser tidak menghargai nilai TTL dari respons DNS dan menyetel sendiri waktu kedaluwarsa.

Caching sistem nama domain server-side

Biasanya, server nama DNS menggunakan caching untuk meningkatkan kinerja. Namun, ada satu jenis server nama DNS khusus yang dirancang hanya untuk caching. Secara alami, ini disebut server nama DNS khusus caching. Jenis server ini tidak menyimpan file zona apa pun sehingga tidak otoritatif untuk domain apa pun. Sebagai gantinya, server nama khusus caching adalah server rekursif yang menerima permintaan rekursif dari klien dan merespons dengan entri cache yang sesuai atau meneruskan kueri dengan mengeluarkan kueri berulang. Kueri ini akan dikirim, satu per satu, ke server DNS akar, server nama TLD yang sesuai, dan server nama tingkat kedua (dan subdomain) yang sesuai, sebagaimana diperlukan. Dengan demikian, cache miss menyebabkan server caching-only beroperasi dengan cara yang sama seperti server nama DNS lokal, menyelesaikan permintaan DNS secara iteratif. Setelah menerima respons kueri dari server nama lain, server khusus caching akan menyimpan hasil kueri untuk akses di masa mendatang. Akses di masa mendatang seperti itu akan menghasilkan hit cache dan dengan demikian secara signifikan mengurangi lalu lintas kueri/respons DNS, mempersingkat waktu kueri DNS yang dirasakan klien. Server khusus caching membalas klien dengan jawaban positif (alamat IP atau nama domain) atau jawaban negatif (pesan kesalahan).

Di Windows dan Linux, server nama DNS lokal sebenarnya adalah server DNS yang hanya menyimpan cache. Gambar 7.26 menunjukkan perbandingan waktu kueri antara cache miss untuk sebuah permintaan dan cache hit untuk sebuah permintaan. Kami menggunakan server nama DNS publik Google dengan alamat IP 8.8.8.8 untuk menanyakan alamat IP nku.edu melalui perintah penggalian Linux. Sebelum mengirim perintah penggalian pertama, kami menggunakan alat tembolok siram Google untuk memastikan bahwa entri DNS nku.edu telah dihapus dari temboloknya. Karena cache dikosongkan, perintah penggalian pertama mengakibatkan cache hilang, dan server nama Google harus meminta server nama lain untuk menyelesaikan permintaan tersebut. Waktu respons kueri adalah 150 md, seperti yang ditunjukkan pada gambar. Perintah dig dijalankan kembali, tetapi sekarang, responsnya di-cache, menghasilkan waktu kueri hanya 50 ms karena cache yang dihasilkan berhasil. Kita dapat melihat bahwa caching dapat secara signifikan mengurangi waktu query.



```
weih@kosh: ~> dig @8.8.8.8 nku.edu +nocomments
;  <<>>  DiG 9.8.1-P1 <<>>  @8.8.8.8   nku.edu  +nocomments
;  (1 server found)
; ;  global options:  +cmd
;  nku.edu.              IN          A
nku.edu           3599  IN          A           192.122.237.7
; ; Query time : 150 msec
; ;  SERVER : 8.8.8.8#53  (8.8.8.8)
WHEN:  Fri Oct 10 09:20:41 2014
; ; MSG  SIZE  rcvd: 41

weih@kosh: ~>  dig  @8.8.8.8  nku.edu   +nocomments
;  <<>>  DiG  9.8.1-P1  <<>>  @8.8.8.8  nku.edu    +nocomments
;  (1 server found)
; ;  global options:  +cmd
;  nku.edu.              IN          A
nku.edu           3539  IN          A           192.122.237.7
; ; Query time : 50 msec
; ;  SERVER : 8.8.8.8#53  (8.8.8.8)
WHEN:  Fri Oct 10 09:20:41 2014
; ; MSG  SIZE  rcvd: 41
```

Gambar 7.26 Perbandingan waktu query antara DNS cache miss dan cache hit.

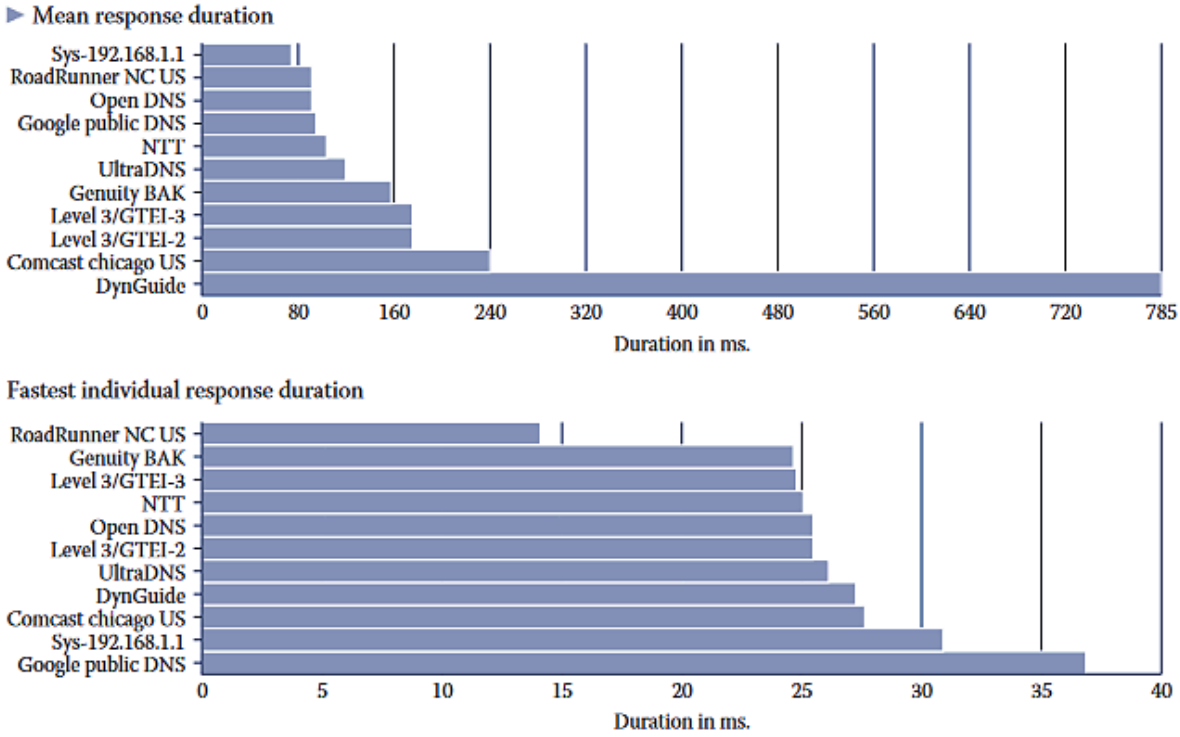
Sebagian besar pengguna Internet menggunakan server nama DNS penyedia layanan Internet (ISP) mereka untuk berfungsi sebagai server nama DNS lokal mereka karena dekat dengan mereka. Kedekatan, bagaimanapun, bukan satu-satunya pertimbangan ketika mencoba untuk meningkatkan kinerja DNS. Server nama DNS publik, seperti DNS publik Google, OpenDNS, dan UltraDNS, mungkin tidak sedekat server nama DNS lokal, server nama tersebut menggunakan cache skala besar untuk meningkatkan tingkat hit cache, dan ditambah dengan infrastruktur jaringan yang kuat yang memanfaatkan secara efektif mekanisme load-balancing, mereka dapat meningkatkan kinerja melalui server DNS lokal.

DNS publik Google memiliki dua tingkat caching. Satu kumpulan mesin berisi alamat nama domain paling populer. Jika kueri tidak dapat diselesaikan dari cache ini, kueri tersebut dikirim ke kumpulan mesin lain yang mempartisi cache berdasarkan nama. Untuk cache tingkat kedua ini, semua kueri untuk nama yang sama dikirim ke mesin yang sama, di mana nama tersebut di-cache atau tidak. Oleh karena itu, klien yang menggunakan server nama publik ini mungkin mendapatkan kinerja pencarian DNS yang lebih baik daripada klien yang menggunakan server nama terdekat, bergantung pada alias IP yang diselesaikan.

Kami menggunakan alat bernama namebench untuk membandingkan kinerja beberapa server nama DNS publik. Gambar 7.27 memberikan hasil pengujian namebench. Eksperimen ini dijalankan pada PC Windows 7 dari kampus Northern Kentucky University. Angka tersebut memberikan dua hasil, waktu respons rata-rata saat menguji sejumlah kueri DNS dan waktu respons tunggal tercepat dari semua kueri DNS yang dikirimkan. Server nama DNS lokal, berlabel SYS-192.168.1.1, memberikan waktu respons rata-rata terbaik (seperti yang ditunjukkan pada baris pertama bagian atas gambar), dengan waktu respons rata-rata 72 ms. RoadRunner North Carolina, Amerika Serikat, memberikan rata-rata terbaik kedua. Di sisi lain, server nama DNS lokal memiliki kinerja termiskin kedua sehubungan dengan tanggapan tunggal tercepat, menunjukkan bahwa hampir semua server publik mengungguli

server lokal untuk permintaan situs web populer, yang sebelumnya akan di-cache di publik ini. Server DNS tetapi tidak di server lokal. Hanya Google Public DNS yang mengungguli server nama lokal sehubungan dengan resolusi alamat populer.

Selain caching sisi klien dan sisi server, router juga dapat menyimpan informasi DNS. Router dapat bertindak sebagai server nama caching untuk klien DNS. Misalnya, router Cisco IOS dapat dikonfigurasi sebagai server nama caching atau server nama otoritatif. Beberapa perute nirkabel juga dapat melakukan caching DNS. Jika Anda terus mendapatkan informasi DNS yang basi setelah cache sisi klien/server dibersihkan, masalahnya mungkin disebabkan oleh cache DNS pada router.



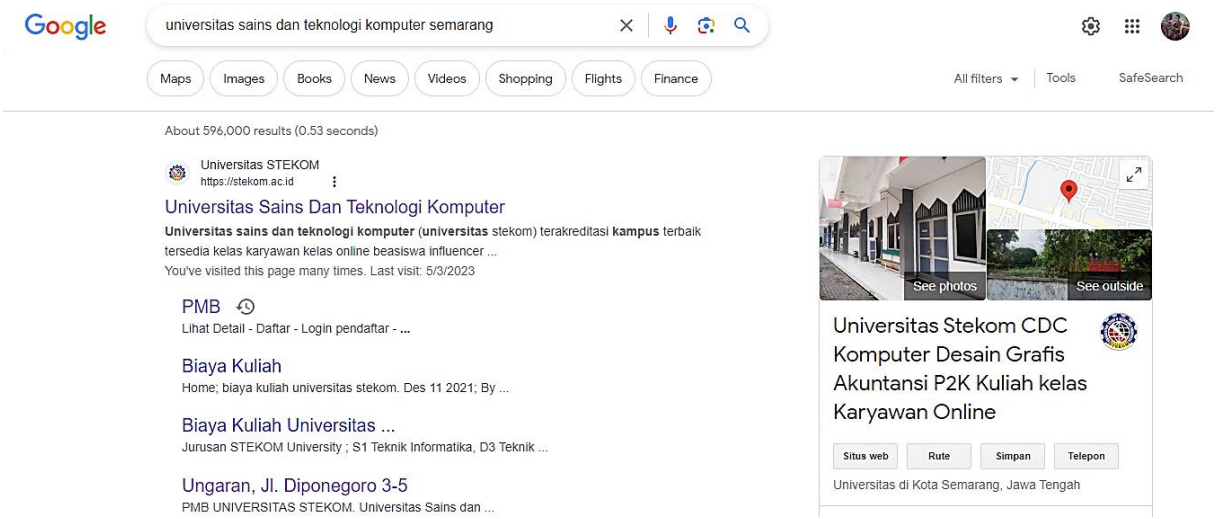
Gambar 7.27 Hasil pengujian Namebench DNS name server publik dan lokal.

Prefetching DNS

Prefetching DNS adalah teknik lain untuk meningkatkan kinerja pencarian DNS. Ini menggunakan waktu siaga prediksi dan unit pemrosesan pusat (CPU) untuk menyelesaikan nama domain menjadi alamat IP sebelum diminta untuk menyelesaikan alamat tersebut. Setelah menerima informasi resolusi yang diambil sebelumnya, ini disimpan dalam cache DNS untuk akses di masa mendatang. Pendekatan ini juga disebut DNS preresolve. Sebagian besar browser web mendukung pengambilan awal DNS untuk meningkatkan pengalaman pengguna web.

Di sini, kami memeriksa bagaimana Google Chrome melakukan prefetching. Gambar 7.28 menunjukkan hasil pencarian ketika kita meng-Google “nku” dengan browser Chrome. Jika Anda perhatikan, di antara tautan yang disediakan tidak hanya milik domain NKU tetapi juga milik Wikipedia, Northerner (surat kabar yang dikelola mahasiswa Universitas, yang bukan bagian dari domain NKU), dan halaman Twitter Universitas (juga bukan bagian dari

domain nku). Jika fitur prefetching DNS diaktifkan, browser akan memindai tag HTML halaman saat ini untuk semua link. Setelah memuat halaman saat ini, browser melakukan pre-resolves nama domain dari masing-masing link lainnya, menyimpan hasil DNS yang telah diselesaikan di cache-nya. Karena hal ini kemungkinan besar terjadi saat pengguna membaca halaman saat ini, operasi prefetching lengkap dapat dilakukan sebelum pengguna siap untuk pindah ke halaman lain.



Gambar 7.28 pencarian Google dengan browser Chrome.

Satu pencarian DNS memakan waktu rata-rata sekitar 100 md. Browser biasanya melakukan beberapa pencarian DNS (prefetches) secara paralel. Dalam kebanyakan kasus, semua tautan pada halaman tertentu akan diselesaikan saat pengguna melihat halaman itu. Prefetching DNS terjadi di latar belakang dan transparan bagi pengguna. Saat pengguna mengklik link yang telah diselesaikan sebelumnya, waktu muat untuk mendapatkan halaman tersebut berkurang karena alamat IP link tersebut sudah ada di cache.

Salah satu kelemahan dari prefetching DNS adalah pemborosan sumber daya (CPU, bandwidth jaringan, dan ruang memori) ketika alamat tautan diselesaikan dan pengguna tidak pernah mengakses tautan itu. Ini mungkin bukan masalah yang signifikan untuk komputer desktop dan server. Namun, ini memiliki potensi untuk secara dramatis memengaruhi kinerja perangkat seluler karena perangkat tersebut memiliki daya CPU, bandwidth jaringan, dan ruang memori yang jauh lebih terbatas. Ini mungkin juga menguras baterai perangkat seluler yang tidak perlu. Selanjutnya, ini dapat menurunkan kinerja server nama DNS karena kami sekarang meminta server nama untuk menanggapi lebih banyak pertanyaan, beberapa di antaranya mungkin tidak pernah diminta.

Apa yang ingin kami lakukan adalah membuat prediksi yang akurat tentang tautan mana yang harus diselesaikan dan dengan demikian di-prefetch daripada memiliki banyak prefetch yang berpotensi tidak perlu. Salah satu solusinya adalah mengintegrasikan kebijaksanaan pengembang halaman web. Untuk halaman web tertentu, pengembang halaman mengetahui konten halaman lebih baik daripada browser (namun, browser semakin pintar). Pengembang dapat membuat prediksi pada tautan yang kemungkinan besar akan

diklik oleh pengguna yang berkunjung dalam waktu dekat. Untuk mendukung ini, pengembang memerlukan beberapa tag HTML khusus untuk memberi tahu browser tentang tautan yang alamatnya harus diambil sebelumnya. HTML 5 mendefinisikan tag dns-prefetch untuk tujuan ini. Tag HTML berikut menginstruksikan browser web untuk mengambil terlebih dahulu alamat IP www.stekom.ac.id (jika tidak tersedia di cache lokal).

```
<link rel="prefetch" href="http://www.stekom.ac.id">
```

Kode HTML ini akan ditempatkan di halaman web dengan teks dan kode HTML lainnya. Tag ini berfungsi sebagai petunjuk ke browser. Browser dapat mengabaikannya.

Tag HTML 5 menarik lainnya yang layak disebutkan adalah prefetch, digunakan untuk menginstruksikan browser untuk melakukan prefetch konten untuk sebuah link, karena pengguna dapat mengklik link tersebut dalam waktu dekat. Dalam hal ini, halaman web itu sendiri yang di-prefetch, bukan hanya alamat IP dari nama host. Tagnya sama seperti di atas, kecuali bahwa prefetch menggantikan dns-prefetch, seperti pada <link rel="prefetch" href="http://www.stekom.ac.id">.

Mari kita pertimbangkan berapa biaya penggunaan dns-prefetch versus prefetch. Dalam kasus dns-prefetch, browser akan mengirimkan kueri DNS dan meng-cache responsnya. Permintaan DNS tipikal adalah sekitar 100 byte. Prefetch halaman membuat permintaan dan respons HTTP. Meskipun permintaan HTTP biasanya cukup kecil (beberapa ratus byte), responsnya adalah halaman web yang ukurannya dapat berkisar dari 1 KB hingga beberapa megabyte, bergantung pada konten di halaman tersebut. Pengguna seluler membayar jumlah bandwidth jaringan yang digunakan komunikasi mereka. Prediksi buruk oleh tag prefetch dapat menghabiskan lebih banyak uang bagi pengguna daripada tag dns-prefetch.

Browser Chrome mengambil data DNS untuk 10 domain yang terakhir diakses pengguna. Ini diambil sebelumnya untuk startup di masa mendatang. Hal ini ditunjukkan pada Gambar 7.29. Anda dapat melihat aktivitas prefetching browser Chrome dengan mengunjungi halaman about:histograms/DNS dan about:dns.

Kelemahan lain dari prefetching DNS muncul jika alamat IP berubah antara waktu prefetching dan permintaan sebenarnya ke server yang bersangkutan. Ini tidak mungkin, karena prefetching biasanya terjadi hanya beberapa detik atau beberapa menit sebelum permintaan pengguna, tetapi mungkin saja. Misalnya, jika pengguna meninggalkan komputer selama beberapa jam dan kembali dan memilih tautan pada halaman web saat ini, hasilnya adalah alamat IP yang diambil sebelumnya sudah kedaluwarsa tetapi masih digunakan dalam permintaan HTTP, menghasilkan pesan kesalahan. bahwa halaman tidak tersedia atau host tidak ditemukan. Untuk mencegah masalah ini, kami dapat menonaktifkan prefetching DNS. Sebagian besar browser mengizinkan pengguna untuk mengaktifkan atau menonaktifkan fitur prefetching DNS dan prefetching halaman web. Kami melihat bagaimana mengontrol ini dengan browser Chrome pada Gambar 7.30, dengan memilih Pengaturan, kemudian memilih Tampilkan Pengaturan Lanjutan, dan kemudian membatalkan pilihan "Prediksi tindakan jaringan untuk meningkatkan kinerja pemuatan halaman" di bagian Privasi.

Sistem Penimbangan Beban Dan Nama Domain

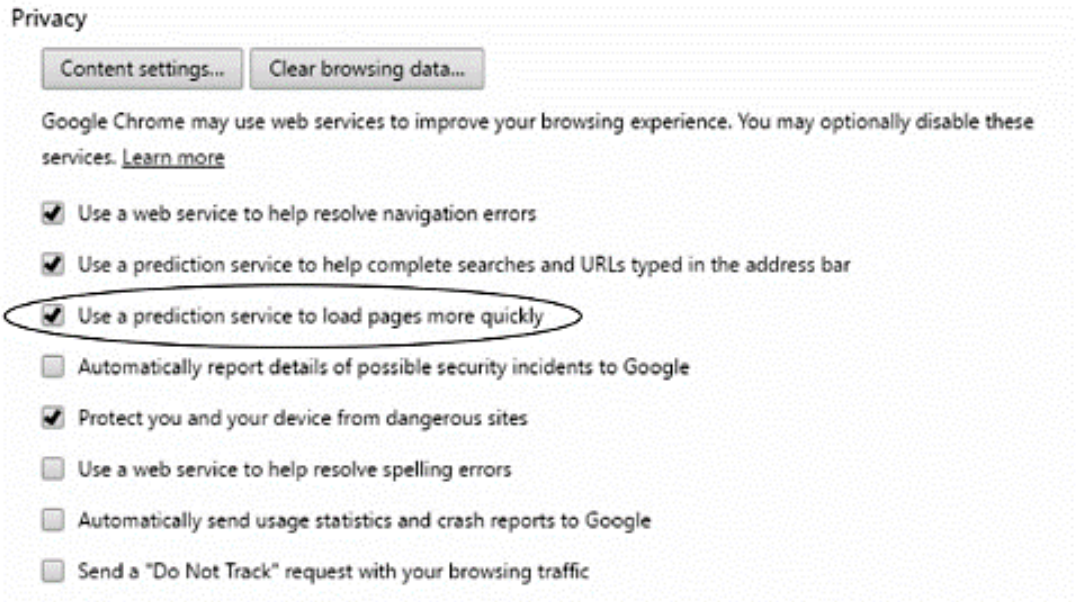
Load balancing adalah teknik distribusi untuk menyebarkan beban kerja ke beberapa sumber daya komputasi (komputer individual, termasuk server dan mesin virtual), sumber daya jaringan (router dan sakelar), dan sumber daya penyimpanan (disk, kaset, penyimpanan terpasang jaringan, dan jaringan area penyimpanan).

Future startups will prefetch DNS records for 10 hostnames

Host name	How long ago (HH:MM:SS)	Motivation
http://cdnjs.cloudflare.com/	02:55	n/a
http://fonts.googleapis.com/	02:55	n/a
http://fonts.gstatic.com/	02:54	n/a
http://www.nku.edu/	02:55	n/a
https://adfs.nku.edu/	89:55:41	n/a
https://ajax.googleapis.com/	89:55:24	n/a
https://mynku.nku.edu/	119:33:57	n/a
https://mynkuerp.nku.edu/	89:55:28	n/a
https://www.google.com/	03:42	n/a
https://www.newhapzing.com/	02:55	n/a

Gambar 7.29 Aktivitas prefetching browser Chrome.

Tujuan dari load balancing adalah untuk mencapai kinerja sistem sebaik mungkin dengan mengoptimalkan sumber daya yang tersedia. Pertimbangkan, misalnya, situs web populer yang menangani ribuan atau bahkan jutaan pengunjung secara bersamaan. Tidak peduli seberapa kuat sebuah server, satu server saja tidak cukup untuk menangani semua permintaan ini. Beberapa server dengan alamat IP berbeda dan skema penyeimbangan beban diperlukan untuk situs web.



Gambar 7.30 Menonaktifkan prefetching DNS di browser Chrome.

Mari kita perhatikan sebuah contoh. Ketika server nama memiliki banyak alamat IP, semua alamat dikembalikan melalui respons DNS. Kemudian, perangkat akan menghubungi server nama yang diberikan dengan menggunakan alamat IP pertama yang diterima sebagai respons.

Kami menggunakan perintah nslookup untuk mendapatkan daftar alamat IP server nama google.com. Kami mengulangi operasi ini beberapa detik kemudian. Seperti yang ditunjukkan pada Gambar 7.31, perintah mengembalikan semua server nama tetapi dalam urutan yang berbeda. Idenya adalah klien yang ingin mendapatkan alamat IP dari server nama google akan menerima alamat IP yang berbeda berdasarkan penyeimbangan muatan. Dalam hal ini, nslookup awal menghasilkan alamat IP pertama menjadi 74.125.225.36. Beberapa detik kemudian, urutan yang berbeda dikembalikan, dimana 74.125.225.41 adalah alamat pertama yang ditawarkan. Perangkat yang menanyakan google.com untuk alamat IP dari server namanya akan menghubungi server fisik yang berbeda karena mereka menerima alamat pertama yang berbeda. Kita dapat melihat bahwa server nama Google merotasi alamat IP dengan harapan permintaan klien menggunakan alamat yang berbeda. Jadi, melalui penyeimbangan muatan, alamat IP yang digunakan untuk berkomunikasi dengan Google akan berbeda permintaan demi permintaan.

Ada dua cara di mana kita dapat melakukan load balancing. Penyeimbangan muatan berbasis perangkat keras dilakukan oleh perangkat perangkat keras, biasanya router. Penyeimbangan beban berbasis perangkat lunak menggunakan aplikasi penyeimbang beban yang berjalan di server. Pertama-tama mari kita fokus pada penyeimbangan muatan berbasis perangkat keras dengan mempertimbangkan router seri Cisco 7200/7500. Router jenis ini dapat melakukan penyeimbangan muatan konten yang bergerak melalui jaringannya. Pertimbangkan situs web yang didukung oleh beberapa server web. Di LAN organisasi ini, kami menempatkan beberapa entri perutean ke dalam perute LAN tersebut. Ini memungkinkan banyak tautan/jalur ke server web tujuan. Router dapat membuat keputusan load-balancing untuk setiap paket HTTP yang masuk berdasarkan tabel routingnya. Keputusan ini kemudian menyebabkan router meneruskan paket ke tujuan yang berbeda. Pendekatan ini disebut penyeimbangan beban lapisan 3, karena terjadi di lapisan ketiga model Open System Interconnection (OSI).

Gambar 7.32 mengilustrasikan ide ini dalam contoh LAN dengan tiga server dan satu router. Ada tiga entri perutean (jalur) untuk situs web di subnet 192.168.1.0/24. Dalam hal ini, router menggunakan skema penjadwalan round-robin; namun, skema distribusi beban lainnya juga dimungkinkan.

```
C:\Users\harvey\nslookup   google.com
Server:  google-public-dns-a.google.com
Address:    8.8.8.8
Non-authoritative answer:
Name: google.com
Addresses:  2607:f8b0:4009:800: :1000
             74.125.225.36
             74.125.225.34
             74.125.225.41
             74.125.225.32
             74.125.225.40
             74.125.225.33
             74.125.225.35
             74.125.225.39
             74.125.225.38
             74.125.225.37
             74.125.225.46

C:\Users\harvey\nslookup   google.com
Server:  google-public-dns-a.google.com
Address: 8.8.8.8

Non-authoritative answer:
Name: google.com
Addresses:  2607:f8b0:4009:800: :1000
             74.125.225.41
             74.125.225.40
             74.125.225.38
             74.125.225.37
             74.125.225.46
             74.125.225.35
             74.125.225.36
             74.125.225.39
             74.125.225.33
             74.125.225.32
             74.125.225.34
```

Gambar 7.31 Kueri DNS untuk google.com menunjukkan algoritma load-balancing.

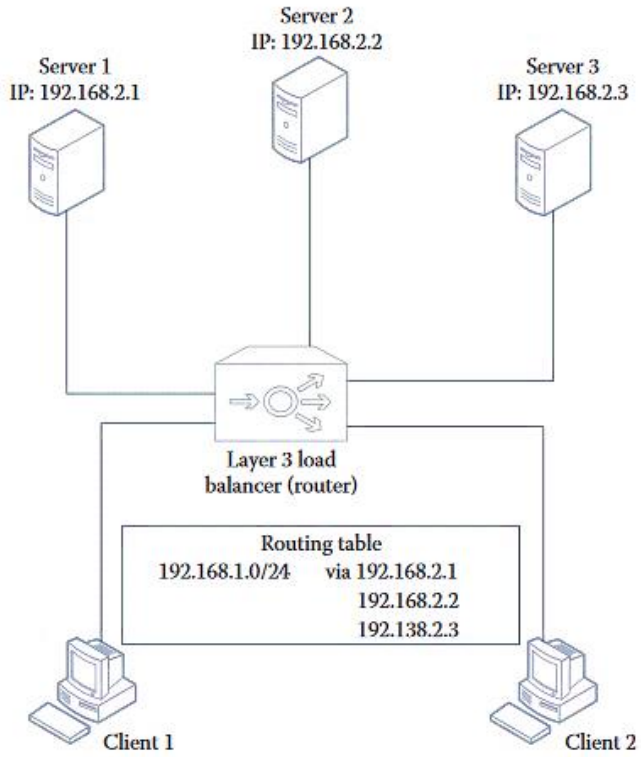
Dengan skema round-robin, paket pertama diteruskan ke server 1, selanjutnya ke server 2, selanjutnya ke server 3 dan selanjutnya kembali ke server 1.

Keuntungan dari load balancing layer 3 adalah sederhana dan mudah diimplementasikan. Sebagian besar router menawarkan penyeimbangan muatan sebagai fungsi bawaan. Masalah utama dengan pendekatan ini adalah router beroperasi pada lapisan jaringan dan tidak memiliki pengetahuan tentang protokol lapisan atas. Dengan demikian, setiap paket diperlakukan sebagai jenis paket yang sama, dan paket diperlakukan secara independen satu sama lain. Bagaimana jika satu paket adalah permintaan yang membutuhkan sumber daya dari server tertentu (mari kita asumsikan bahwa server 1 menangani semua skrip sisi server)? Sekarang, jika paket seperti itu diterima oleh server 2 atau server 3, server yang dipilih inilah yang harus meneruskan paket ke lokasi yang sesuai. Di sisi lain, karena paket diperlakukan secara independen, dua paket milik satu koneksi TCP dapat diteruskan oleh router ke dua server yang berbeda. Dari sudut pandang lapisan jaringan, itu adalah distribusi beban yang masuk akal. Namun, ini adalah distribusi beban yang buruk dari perspektif lapisan transport.

Dengan demikian, load balancing juga dapat dilakukan pada layer transport OSI. Ini membutuhkan jenis switch atau router khusus, yang disebut switch/router layer 4. Di sini, load balancing memperhitungkan konten yang dianalisis pada lapisan transport. Biasanya, switch adalah perangkat layer 2 dan router adalah perangkat layer 3. Switch/router layer 4 lebih kuat

karena dapat mengatasi masalah router yang melihat semua paket sebagai independen dan bebas dari konteksnya (protokol).

Ini membawa kita ke ide lain. Router layer 3 hanya melihat alamat IP tujuan untuk membuat keputusan penyeimbangan beban. Server tujuan mungkin melayani banyak protokol seperti HTTP, Hypertext Transfer Protocol Secure (HTTPS), dan File Transfer Protocol (FTP).



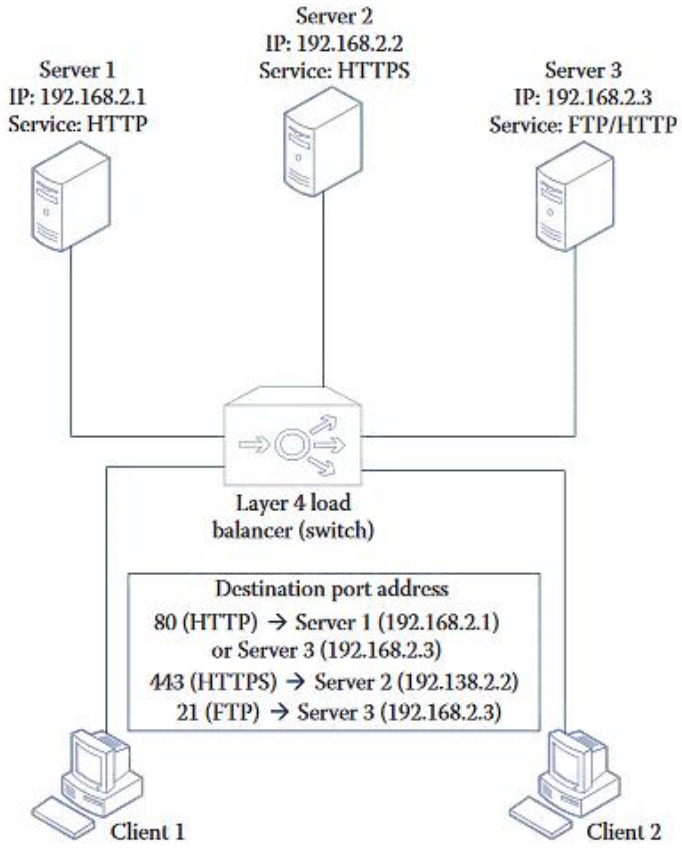
Gambar 7.32 Load balancing layer 3

Keputusan penyeimbangan beban yang dibuat berdasarkan jenis protokol mungkin memberikan kinerja yang lebih baik. Layer 4 switch/router tidak hanya menggunakan alamat IP sumber/tujuan dari sebuah paket tetapi juga nomor port sumber/tujuan dari paket tersebut. Nomor port tujuan mengidentifikasi protokol (aplikasi) yang dapat membantu switch/router mengidentifikasi jenis lalu lintas dan menggunakannya untuk menerapkan kebijakan penyeimbangan muatan yang berbeda. Selain itu, terjemahan alamat jaringan (NAT) terjadi di lapisan 3. Namun, varian yang disebut Port Address Translation (PAT) terjadi di lapisan 4. Ini seperti NAT, kecuali bahwa alamat IP dan nomor port digunakan untuk menentukan alamat IP dan port internal apa yang harus digunakan. Ini diperlukan ketika satu alamat IP eksternal dipetakan ke beberapa alamat jaringan internal (NAT satu-ke-banyak). Banyak organisasi yang menggunakan NAT sebenarnya menggunakan PAT dan memerlukan perangkat lapisan 4 untuk menangani terjemahan.

Contoh load-balancing layer 4 ditunjukkan pada Gambar 7.33. Dalam contoh ini, sakelar lapisan 4 ditempatkan di depan tiga server untuk sebuah situs web. Situs web ini mendukung tiga protokol: HTTP, HTTPS, dan FTP. Server 1 (192.168.2.1) digunakan untuk melayani permintaan HTTP. Server 2 (192.168.2.2) digunakan untuk melayani permintaan

HTTPS. Server 3 (192.168.2.3) dapat melayani permintaan FTP dan HTTP. Nama domain situs web dipetakan ke alamat IP sakelar (192.168.1.1).

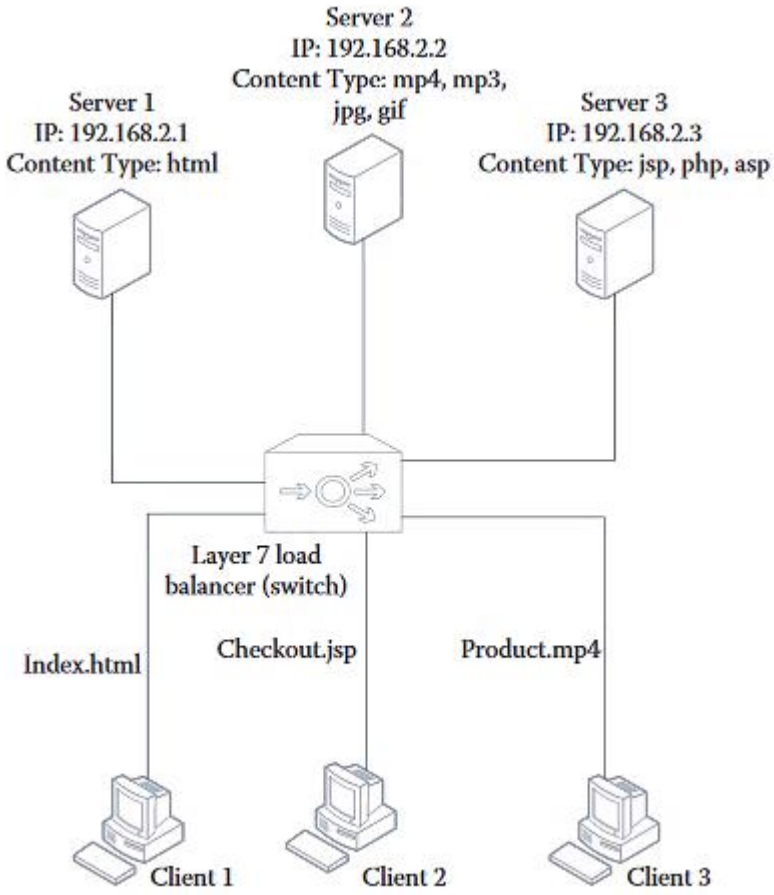
Saat sakelar dalam contoh ini menerima permintaan, ia menggunakan nomor port tujuan dari permintaan tersebut untuk mengidentifikasi jenis permintaan dan kemudian melakukan PAT tujuan atas permintaan tersebut. Dengan demikian, alamat IP tujuan permintaan dipetakan ke alamat IP salah satu dari tiga server, yang menjadi dasar server tersebut dapat menangani aplikasi. Permintaan yang masuk ke port 80 diidentifikasi sebagai permintaan HTTP dan diarahkan ke server 1 atau server 3 (pilihan server mana yang akan digunakan akan didasarkan pada beberapa kriteria lain yang akan dibahas nanti di bagian ini). Permintaan yang dikirim ke port 443 dianggap sebagai permintaan HTTPS dan diteruskan ke server 2. Permintaan yang dikirim ke port 21 diidentifikasi sebagai permintaan FTP dan diarahkan ke server 3. Setelah memproses permintaan, server mengirimkan respons ke sakelar . Sakelar melakukan PAT sumber pada respons, sehingga alamat IP sumber respons dipetakan ke alamat IP sakelar. Dalam konfigurasi ini, klien memiliki ilusi bahwa mereka berkomunikasi dengan satu server, yang sebenarnya adalah saklar. Switch menerima permintaan atas nama sekelompok server dan mendistribusikan permintaan berdasarkan kebijakan penyeimbangan beban yang telah ditentukan sebelumnya. Peralihan itu seperti server virtual untuk situs web. Alamat IP sakelar adalah alamat IP virtual situs web. Situs web menyajikan alamat IP virtual ini kepada kliennya.



Gambar 7.33 Penyeimbangan beban lapisan 4.

Meskipun penyeimbang beban lapisan 4 membuat keputusan yang lebih tepat daripada penyeimbang beban lapisan 3, itu mungkin masih membuat keputusan penyeimbangan beban yang buruk karena tidak memahami protokol di lapisan atasnya, seperti protokol HTTP di lapisan aplikasi. Pada contoh sebelumnya, server 1 dan server 3 melayani semua permintaan HTTP. Mari kita pertimbangkan situasi ini: Klien 1 meminta halaman index.html. Permintaan HTTP ini diteruskan oleh peralihan ke server 1. Server 1 menemukan halaman di disk dan memuat halaman dari disk ke memorinya (cache) untuk menghasilkan respons HTTP. Server 1 mengirimkan respons ke sakelar. Sakelar mengembalikan respons ke klien 1. Permintaan berikutnya, yang berasal dari klien 2, adalah untuk halaman yang sama, index.html. Saklar beroperasi pada lapisan 4 dan memahami bahwa ini adalah permintaan HTTP tetapi tidak mengerti untuk apa permintaan itu (halaman web tertentu). Saklar, menggunakan algoritme penyeimbangan muatannya, meneruskan permintaan ke server 3. Seandainya permintaan dikirim ke server 1, server ini sudah memiliki halaman yang di-cache di memori dan karenanya dapat menghasilkan respons jauh lebih cepat daripada server 3, yang akan memerlukan akses disk, diikuti dengan menghasilkan respons HTTP. Hasilnya adalah ada satu salinan index.html di memori server 1 dan satu salinan index.html di memori server 3. Dengan demikian, penyeimbangan muatan yang ditunjukkan di sini tidak terlalu efektif (dalam kasus ini). Idealnya, permintaan kedua harus diteruskan ke server 1.

Untuk membuat keputusan penyeimbangan muatan yang lebih terinformasi, diperlukan sakelar lapisan 7. Sakelar semacam itu dirancang untuk melakukan penyeimbangan beban pada lapisan aplikasi. Sakelar lapisan 7 membuat keputusan berdasarkan konten protokol lapisan aplikasi dari permintaan yang diberikan. Misalnya, ketika sebuah switch menerima permintaan HTTP, ia mem-parsing header HTTP, mengekstraksi informasi penting seperti Uniform Resource Locator (URL) dari permintaan tersebut. Itu membuat keputusan penerusan berdasarkan URL atau informasi lain dalam permintaan.



Gambar 7.34 load balancing lapisan 7.

Dengan menggunakan switch layer 7, sebuah situs web dapat menggunakan server yang berbeda untuk menyajikan jenis konten yang berbeda. Gambar 7.34 menunjukkan contoh load balancing layer 7. Dalam contoh, server 1 dan server 2 disetel untuk menyajikan konten statis. Server 1 digunakan untuk melayani halaman Hypertext Markup Language (HTML). Server 2 digunakan untuk menyajikan konten multimedia, seperti gambar, audio, dan video. Server 3 dioptimalkan untuk melayani konten dinamis seperti halaman yang memanfaatkan JSP, ASP, dan PHP. Switch membedah header HTTP dari setiap permintaan dan kemudian mengarahkan permintaan ke server berdasarkan jenis objek web yang diminta. Karena permintaan yang sama selalu dilayani oleh server yang sama, itu dapat mencapai tingkat hit cache yang lebih baik. Selain itu, setiap server dapat dioptimalkan untuk melayani jenis permintaan tertentu berdasarkan karakteristik permintaan. Misalnya, permintaan konten statis biasanya menghasilkan aktivitas input/output (I/O) hanya-baca di server. Namun, permintaan konten dinamis dapat menghasilkan aktivitas baca dan tulis I/O di server. Kita dapat menggunakan konfigurasi penyimpanan yang berbeda, seperti konfigurasi RAID, pada server yang berbeda untuk mengoptimalkan kinerja operasi I/O per detik (topik yang dibahas nanti dalam teks). Keputusan penyeimbangan muatan yang lebih cerdas yang dibuat oleh sakelar lapisan 7 dapat mencapai kinerja yang lebih baik untuk klien.

Meskipun switch layer 7 dapat membuat keputusan penyeimbangan muatan yang lebih baik daripada router layer 3 dan switch layer 4, dibutuhkan lebih banyak waktu untuk membuat keputusan penerusan karena kebutuhan untuk melakukan pemeriksaan permintaan yang lebih dalam. Dibandingkan dengan menganalisis informasi header layer aplikasi, router layer 3 hanya perlu mengurai unit data protokol (PDU) layer 1 melalui PDU layer 3 dan switch layer 4 perlu memproses PDU layer 1 melalui PDU layer 4.

Kami dapat memindahkan beberapa atau semua pengambilan keputusan penyeimbangan beban di DNS ke perangkat lunak alih-alih membuat keputusan dibuat oleh router dan sakelar. Ide dasar dari pendekatan load-balancing berbasis perangkat lunak terletak pada kemampuan server DNS untuk menentukan beberapa catatan A untuk nama domain. Misalnya, pertimbangkan situs web yang dilayani oleh beberapa server web. Setiap server memiliki alamat IP yang berbeda. Satu catatan A ditentukan untuk setiap server web. Kutipan dari file zona yang mengilustrasikan hal ini ditunjukkan sebagai berikut:

```
; zone file for cs.nku.edu
$TTL 3600    ;   1 hour default TTL for zone
$ORIGIN cs.nku.edu.
@      IN     SOA    ns1.cs.nku.edu. root.cs.nku.edu. (
…
)
...
; Host Names
www    IN A   10.10.10.10
       IN A   10.10.10.11
       IN A   10.10.10.12
...
```

Dalam contoh ini, tiga server tersedia dengan satu nama www.cs.nku.edu, dengan alamat IP masing-masing 10.10.10.10, 10.10.10.11, dan 10.10.10.12. Ketika server DNS untuk domain cs.nku.edu menerima permintaan resolusi nama untuk www.cs.nku.edu, ia memilih satu server berdasarkan kebijakan yang telah ditentukan dan kemudian mengembalikan alamat IP dari server yang dipilih ke klien. Alternatifnya, server DNS mengembalikan seluruh daftar alamat IP server ke klien, tetapi urutan daftar bervariasi setiap saat.

Mari kita periksa beberapa kebijakan load-balancing yang lebih populer yang mungkin digunakan server DNS atau load balancer layer 3/4/7. Kami sebelumnya menyebutkan kebijakan round-robin, yang hanya merotasi daftar alamat IP (atau merotasi urutan alamat IP). Kami mengeksplorasi bagaimana menetapkan kebijakan round-robin selanjutnya. Lainnya termasuk kebijakan failover, kebijakan berbasis geolokasi, kebijakan berbasis beban, kebijakan berbasis latensi, dan kebijakan berbasis hash. Di bawah ini, kami menjelaskan setiap kebijakan. Kebijakan round-robin memperlakukan semua server secara setara dan merotasi server dalam satu putaran. Misalnya, tiga server dengan kemampuan yang sama (S1, S2, dan S3) dikerahkan untuk melayani situs web. Kebijakan round-robin mengarahkan permintaan ke grup tiga server dalam pola rotasi ini: S1-S2-S3-S1-S2-S3. Di BIND 9.10 (yang akan kita jelajahi di Bab 6), kita dapat menggunakan direktif rrset-order untuk menentukan urutan di mana beberapa record A dikembalikan. Arahan mendukung tiga nilai urutan: tetap, acak, dan siklik. Pilihan tetap mengembalikan catatan dalam urutan yang ditentukan dalam file zona. Pilihan

acak mengembalikan catatan dalam urutan acak. Siklik pilihan mengembalikan rekaman dengan cara round-robin. Misalnya, rrset-order {order cyclic;}; berarti semua catatan untuk semua domain akan dikembalikan dalam urutan round-robin.

Kebijakan round-robin mudah diterapkan. Namun, jika server tidak memiliki kapasitas yang sama, kebijakan round-robin mungkin tidak efisien. Asumsikan bahwa S1 dan S2 memiliki kapasitas yang sama dalam hal CPU, DRAM (memori utama), ruang disk, dan bandwidth jaringan, sedangkan S3 memiliki kapasitas dua kali lipat. S3 akan kurang dimanfaatkan dengan kebijakan round-robin. Untuk mengatasi masalah ini, varian round-robin yang disebut round-robin berbobot ditambahkan ke BIND. Kebijakan round-robin berbobot memberikan bobot pada setiap server. Peluang server, Si, dipilih ditentukan oleh persamaan berikut:

$$\text{Chance}(S_i) = \frac{\text{Weight}(S_i)}{\sum_1^n \text{Weight}(S_j)} * 100\%$$

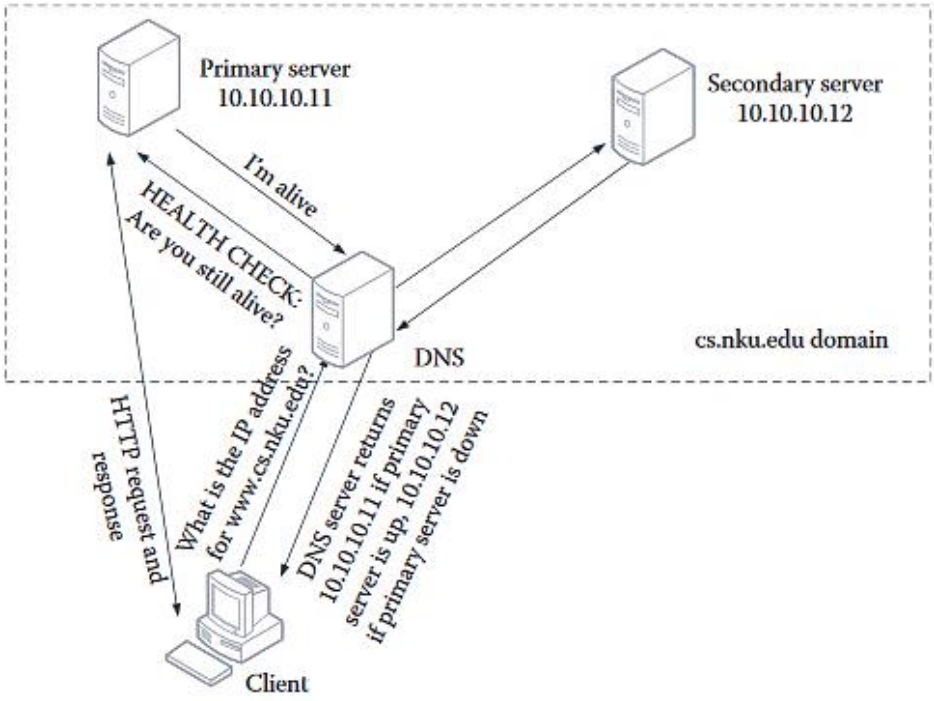
Dengan kebijakan ini, Anda dapat menetapkan bobot yang lebih besar ke server dengan kapasitas lebih besar, sehingga server menerima dan memproses lebih banyak permintaan. Jika S1, S2, dan S3 masing-masing memiliki bobot 0,25, 0,25, dan 0,5, maka S3 harus menerima permintaan dua kali lebih banyak daripada S1 atau S2. Kebijakan round-robin berbobot mengarahkan permintaan ke grup tiga server dalam pola rotasi ini: S1-S3-S2-S3-S1-S3-S2-S3.

Kebijakan failover mengarahkan permintaan berdasarkan ketersediaan server. Failover adalah operasi toleransi kesalahan yang memungkinkan pekerjaan yang biasanya dilakukan oleh server utama ditangani oleh server lain, jika server utama menjadi tidak tersedia (karena kegagalan atau pemeliharaan). Layanan DNS biasanya merupakan titik kontak pertama antara klien dan situs web. Dengan demikian, server DNS adalah tempat yang baik untuk menerapkan kebijakan failover. Dalam pengaturan failover, server DNS secara berkala memeriksa kesehatan server (daya tanggap server) dan merespons pertanyaan klien dengan hanya menggunakan server yang sehat. Jika server tidak responsif, catatan DNS-nya akan dihapus dari kumpulan server sehingga alamat IP server tidak dikembalikan ke klien. Saat server tersedia kembali dan mulai merespons kueri pemeriksaan kondisi, catatan DNS-nya ditambahkan kembali ke kumpulan server sehingga server DNS mulai mengarahkan permintaan ke server.

Server nama DNS mengirimkan setiap server kueri pemeriksaan kesehatan pada interval yang ditentukan. Jika server merespons dalam periode waktu tunggu yang ditentukan, pemeriksaan kesehatan berhasil dan server dianggap sehat. Jika server dalam keadaan sehat tetapi server gagal merespons sejumlah pemeriksaan kueri berturut-turut, server DNS mengubah statusnya menjadi tidak sehat dan berhenti menggunakan alamat IP server ini sebagai tanggapan atas permintaan DNS klien. Jumlah pemeriksaan kueri berturut-turut ditentukan oleh nilai ambang yang tidak sehat. Jika server dalam keadaan tidak sehat tetapi server berhasil merespons sejumlah pemeriksaan permintaan berturut-turut, server DNS mengubah keadaan server web kembali menjadi sehat dan mulai mengarahkan permintaan klien ke server itu. Jumlah pemeriksaan kueri berturut-turut ditentukan oleh nilai ambang

yang sehat. Server nama DNS menentukan nilai untuk interval kueri pemeriksaan kesehatan, batas waktu respons, ambang sehat, dan ambang tidak sehat.

Kita melihat contoh kebijakan failover pada Gambar 7.35. Dalam contoh ini, ada dua server, server primer dan server sekunder, untuk situs web www.cs.nku.edu. Saat server utama sehat, server DNS merespons permintaan DNS dengan alamat IP server utama (10.10.10.11). Server utama menangani permintaan klien, sehingga dalam keadaan aktif. Server sekunder tetap diam, sehingga dalam keadaan pasif. Jika server primer menjadi tidak sehat, server DNS kemudian merespons permintaan DNS klien dengan alamat IP server sekunder (10.10.10.12). Server sekunder kemudian akan mulai menangani permintaan klien. Pemeriksaan kesehatan dilakukan terus menerus. Jika server utama mulai merespons health check nanti, itu akan dicatat sebagai sehat kembali. Sekarang, server nama DNS akan kembali mengarahkan semua permintaan ke server utama, dan server sekunder akan masuk ke keadaan pasif.

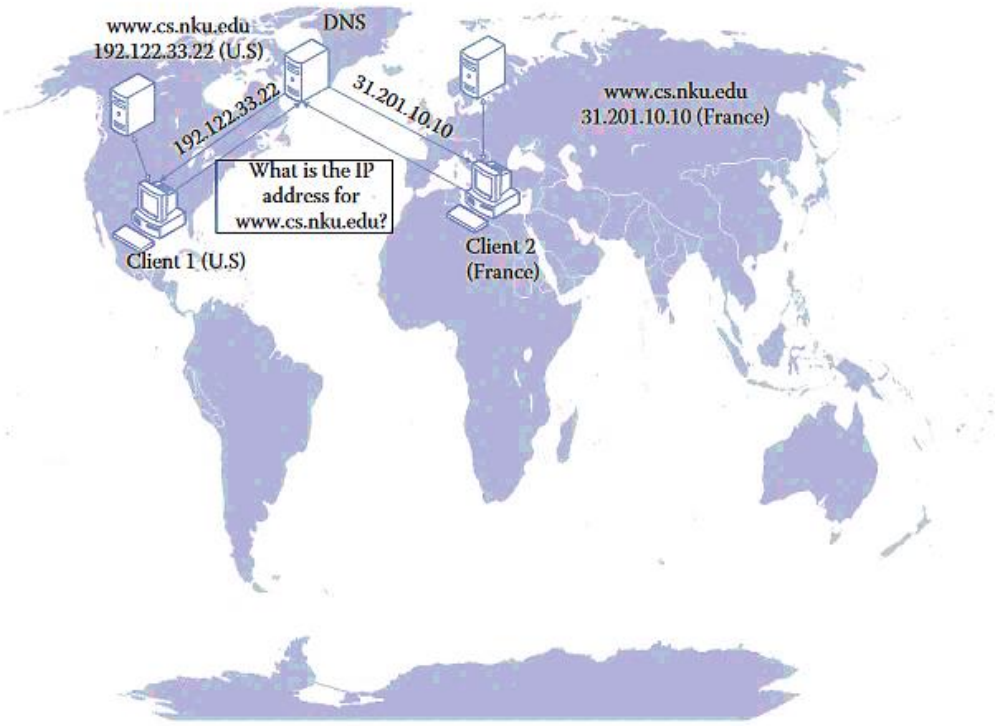


Gambar 7.35 Kebijakan load-balancing failover.

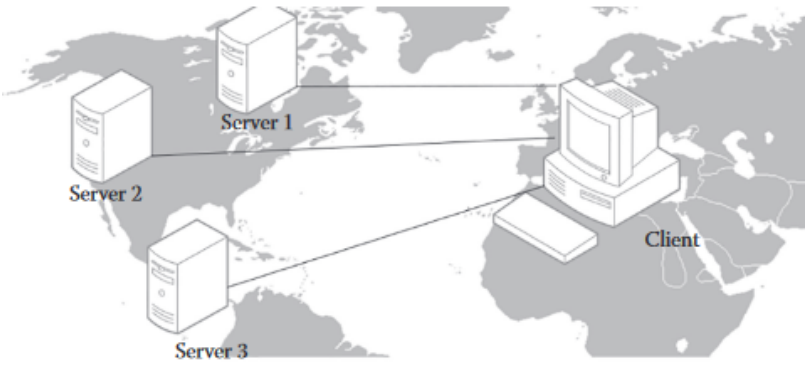
Dalam konfigurasi dua server ini, satu server dalam keadaan aktif untuk menangani permintaan dan server lainnya dalam keadaan pasif. Konfigurasi failover ini disebut failover aktif-pasif. Untuk mencapai ketersediaan sistem yang lebih baik, grup server primer dan grup server sekunder dapat ditentukan. Setiap grup terdiri dari beberapa server, bukan satu server. Ketika server dalam grup ditemukan tidak sehat, itu akan dihapus dari grup. Ketika server menjadi sehat kembali, itu ditambahkan kembali ke grup. Ada beberapa konfigurasi failover lainnya, seperti failover aktif-aktif dan failover campuran.

Kebijakan berbasis geolokasi mengarahkan permintaan klien berdasarkan lokasi geografis klien. Dengan geolokasi, alamat IP klien digunakan untuk menentukan lokasi geografis klien. Server DNS memilih server terdekat dengan klien untuk melayani permintaan.

Gambar 7.36 mengilustrasikan kebijakan berbasis geolokasi dengan mereplikasi situs web www.cs.nku.edu di lokasi geografis kedua (aslinya di Amerika Serikat, dan salinannya ditempatkan di Prancis). Server nama DNS mengarahkan semua permintaan klien ke lokasi yang lebih dekat. Klien 1, berlokasi di Amerika Serikat, menanyakan server nama DNS untuk www.cs.nk.edu. Server nama DNS memeriksa alamat IP klien dan menghitung jarak antara klien dan dua server web, menentukan bahwa server AS lebih dekat. Dengan demikian, server nama DNS merespons klien dengan alamat IP 192.122.33.22, dan Klien 1 mengirimkan permintaan HTTP ke server di Amerika Serikat. Klien 2 berlokasi di Prancis. Server DNS mengarahkan permintaan Klien 2 ke server (31.201.10.10) di Prancis.



Gambar 7.36 Kebijakan berbasis geolokasi.



	IP address	Latitude	Longitude	Geographic distance
Client	10.0.0.1	40.1234	–80.1568	
Server 1	20.10.1.2	39.4321	–78.1547	10.0015
Server 2	31.15.15.5	38.1230	–77.1534	20.1250
Server 3	2.15.15.10	38.1120	–76.1234	15.0125

Gambar 7.37 Jarak geografis antara klien dan server.

Dua metrik kedekatan sering digunakan untuk menentukan kedekatan antara klien dan server. Satu metrik adalah jarak geografis. Alamat IP klien dipetakan ke garis bujur dan garis lintang lokasi untuk subnet tempat alamat IP klien berada. Jarak geografis antara klien dan server kemudian dapat dihitung. Contoh jarak geografis ditunjukkan pada Gambar 7.37. Metrik lainnya adalah jumlah lompatan untuk pesan yang dikirim antara klien dan server. Perintah traceroute, yang telah dibahas di Bab 4, melacak jalur jaringan dari sebuah pesan, menampilkan perangkat yang dijangkau di sepanjang jalur tersebut. Sebagian besar hop perantara adalah router yang meneruskan paket.

Kebijakan berbasis beban meneruskan permintaan ke server yang paling sedikit dimuat. Penyeimbang beban memantau beban setiap server, seperti rata-rata beban CPU. Misalnya, kami mungkin menemukan informasi berikut dengan menggunakan perintah uptime Linux:

```
### 192.168.1.15(stat: 0, dur(s): 0.36):
00:29:57 up 35 min, 0 users, load average: 0.08, 0.02, 0.01
### 192.168.1.12(stat: 0, dur(s): 0.71):
12:47 AM up 52 mins, 0 users, load average: 0.16, 0.03, 0.01
```

Rata-rata tiga beban adalah beban CPU rata-rata selama menit terakhir, 5 menit, dan 15 menit. Kita dapat melihat bahwa server kedua telah berjalan untuk waktu yang lebih lama dan memiliki beban CPU dua kali lipat selama menit terakhir.

Metrik lain yang dapat digunakan untuk load-based balancing adalah jumlah koneksi aktif. Server dengan lebih banyak koneksi aktif biasanya lebih sibuk daripada server dengan koneksi aktif lebih sedikit. Jumlah koneksi sama dengan jumlah permintaan yang dilayani. Gambar 7.38 menunjukkan cara menggunakan perintah netstat (di Windows) untuk mengukur koneksi aktif server. Di bawah kebijakan berbasis beban, penyeimbang muatan akan meminta server dan meneruskan permintaan ke server yang memiliki koneksi aktif paling sedikit dengan klien. Ini terkadang disebut kebijakan koneksi paling sedikit.

```
C:\>netstat
Active  Connections
Proto       Local Address          Foreign Address        State
TCP         192.168.207.14:443     ocs01:64172            ESTABLISHED
TCP         192.168.207.14:443     ocs01:64187            ESTABLISHED
TCP         192.168.207.14:5061    ocs01:64177            ESTABLISHED
TCP         192.168.207.14:5062    ocs01:63999            ESTABLISHED
TCP         192.168.207.14:50112   dc01:58661             ESTABLISHED
TCP         192.168.207.14:50116   dc01:58661             ESTABLISHED
TCP         192.168.207.14:50117   dc01:58661             ESTABLISHED
TCP         192.168.207.14:50830   dc01:58661             ESTABLISHED
TCP         192.168.207.14:50888   dc01:5061              ESTABLISHED
```

Gambar 7.38 Koneksi aktif di server.

Kebijakan berbasis latensi mengarahkan permintaan berdasarkan latensi respons dari permintaan terbaru. Latensi respons adalah waktu yang terjadi antara pengiriman byte pertama permintaan hingga menerima byte terakhir respons. Ini termasuk waktu yang dihabiskan permintaan dan respons dalam melintasi jaringan dan waktu yang dihabiskan server dalam memproses permintaan. Semakin jauh jarak antara klien dan server, biasanya

semakin lama latensi respons. Semakin banyak memuat server, semakin lama latensi respons mungkin. Dengan demikian, kebijakan ini menggabungkan metrik kedekatan dan beban server untuk membuat keputusan penyeimbangan beban. Server nama DNS harus menjaga waktu respons dari server yang menjadi otoritasnya.

Perintah ping dapat digunakan untuk mengukur latensi respons. Salah satu kelemahan menggunakan ping adalah menggunakan Internet Control Message Protocol (ICMP), yang dinonaktifkan oleh banyak administrator melalui firewall untuk melindungi informasi internal jaringan mereka dari serangan pengintaian. Jika demikian, latensi yang akurat dapat diperoleh dengan menggunakan alat berbasis HTTP yang mengukur latensi tersebut. Program ab adalah alat pembandingan server HTTP Apache. Itu dapat mengukur latensi respons HTTP. Selain itu, perhatikan bahwa HTTP adalah protokol lapisan 7, sedangkan ICMP adalah protokol lapisan 3. Jika ping digunakan untuk mengukur latensi, latensi tidak akan menyertakan waktu yang dibutuhkan server tujuan untuk memproses permintaan. Dengan ab, Anda mendapatkan gambaran yang lebih baik tentang bagaimana memuat server, karena ab akan memeriksa paket melalui ketujuh lapisan. Gambar 7.39 menunjukkan contoh perintah ab yang mengukur latensi respons HTTP.

Kebijakan berbasis hash menggunakan fungsi hash untuk mendistribusikan permintaan ke sekelompok server. Sebuah fungsi hash, h, dapat didefinisikan sebagai sisa yang diperoleh dengan membagi jumlah representasi ASCII dari karakter dalam URL permintaan dengan jumlah_server (mendapatkan sisa menggunakan operator modulo, sering ditulis sebagai %). Kita dapat mendefinisikan h sebagai berikut:

$$h = \left(\sum_{i=1}^{\text{length (URL)}} (\text{int})\text{URL}_i \right) \% \text{number\_of\_servers}$$

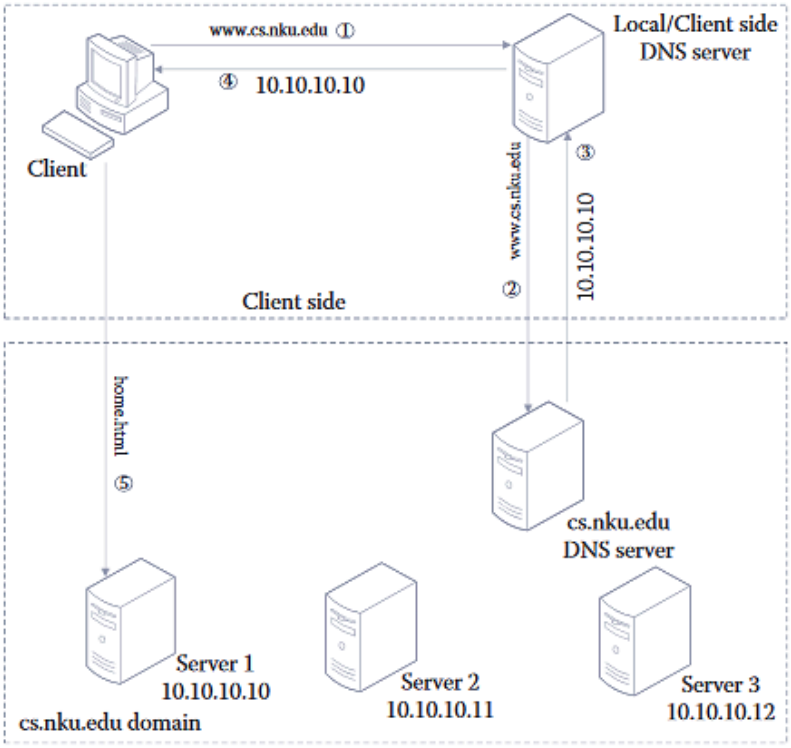
Mari kita perhatikan contoh dengan tiga server untuk sebuah situs web. Fungsi hash akan memetakan URL apa pun ke dalam ruang hash {0, 1, 2}. Kami melakukan hash pada URL permintaan untuk menentukan server mana yang harus melayani permintaan tersebut. Jika h(URLi) = 0, maka request dikirim ke server 1. Jika h(URLi) = 1, maka request dikirim ke server 2. Jika h(URLi) = 2, maka request dikirim ke server 3 .

```
[root@CIT668]# ab -n 10 -c 1 http://yahoo.com/
Benchmarking yahoo.com (be patient) . . . . . done
Server Software:  ATS
Server Hostname:  yahoo.com
Server Port:            80

Document Path:    /
Document Length:  1450 bytes

Concurrency Level:      1
Time taken for tests:   1.033 seconds
Complete requests:      10
Failed requests:        0
Write errors:           0
Non-2xx responses:      10
Total transferred 17458 bytes
HTML transferred  14500 bytes
Requests per second:    9.68 [#/sec] (mean)
Time per requests:      103.332 [ms] (mean, across all concurrent requests)
Transfer rate:          16.50 [Kbytes/sec] received

Connection Times (ms)
                Min   mean [+/-sd]      median           max
Connect:        46    47    0.6   47          48
Processing: 48  57    18.3  48          98
Waiting:        48    57    18.3  48          98
Total:          94    103   18.6  95          145

Percentage of the requests served within a certain time (ms)
50%   95
66%   95
75%   96
80%   131
90%   145
95%   145
98%   145
99%   145
100%  145 (longest request)
```

Gambar 7.39 latensi permintaan HTTP diukur dengan ab.

DNS sisi klien atau DNS sisi server

Ada dua tempat di mana kita dapat melakukan load balancing berbasis DNS: server DNS lokal dan server nama DNS otoritatif. Pada Gambar 7.40, kita melihat contoh load balancing server otoritatif. Di sini, klien meminta server nama DNS lokal untuk alamat IP www.cs.nku.edu. Server nama DNS lokal tidak memiliki entri ini di-cache sehingga menghubungi server nama DNS otoritatif untuk resolusi nama. Tiga server web digunakan untuk www.cs.nku.edu, bernama server 1, server 2, dan server 3, dengan alamat IP masing-masing 10.10.10.10, 10.10.10.11, 10.10.10.12. Oleh karena itu, server nama otoritatif memiliki tiga data A untuk nama www.cs.nku.edu. Server nama DNS otoritatif harus melakukan penyeimbangan muatan untuk memutuskan mana dari tiga alamat IP yang akan dikembalikan. Dalam hal ini, ia mengembalikan 10.10.10.10, yang kemudian diteruskan dari server nama DNS lokal ke klien. Klien sekarang mengirimkan permintaan HTTP untuk halaman home.html ke server 1 (10.10.10.10).



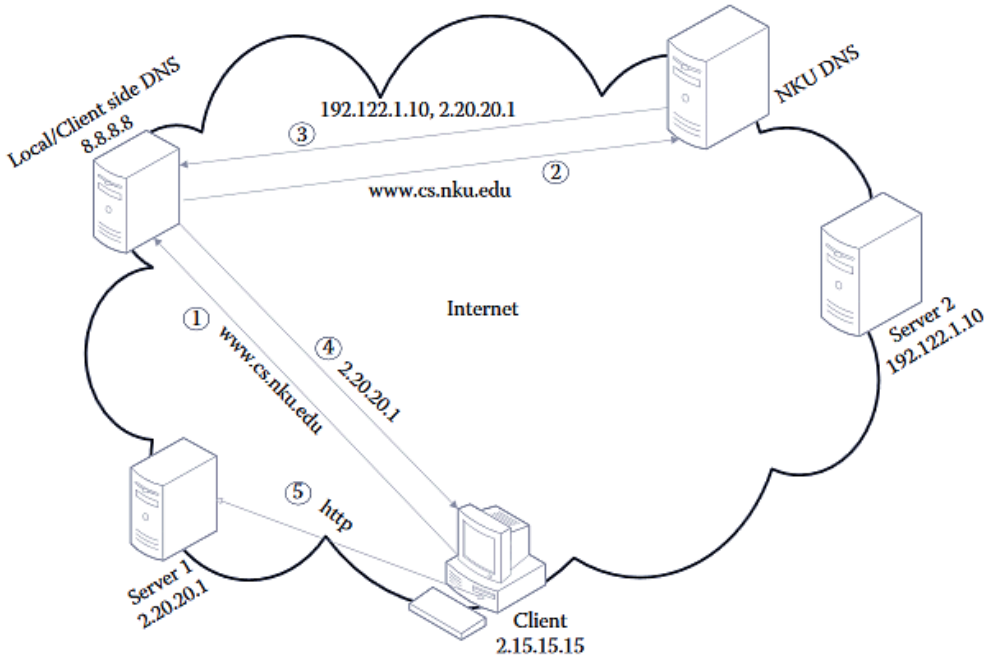
Gambar 7.40 Load balancing pada server DNS otoritatif.

Biasanya, klien menggunakan server nama DNS ISP mereka sebagai server nama DNS lokal mereka. Ini secara geografis dekat dengan klien. Namun, pengguna juga dapat mengonfigurasi DNS resolver mereka untuk menggunakan server nama DNS publik, seperti server OpenDNS Google. Server nama DNS publik biasanya akan lebih jauh dari klien daripada server nama DNS ISP. Hal ini dapat menyebabkan masalah untuk load balancing sisi server DNS otoritatif jika server nama otoritatif tersebut menggunakan kebijakan load-balancing berbasis geolokasi.

Sebagai contoh, kami telah mereplikasi situs web www.cs.nku.edu ke dalam dua server di dua lokasi geografis. Server 1 dengan alamat IP 2.20.20.1 berlokasi di Prancis, dan server 2 dengan alamat IP 192.122.1.10 berada di Amerika Serikat. Klien di Prancis dengan alamat IP 2.15.15.15 menggunakan salah satu server publik Google dengan alamat IP 8.8.8.8 di Amerika Serikat sebagai server nama DNS lokalnya. Klien mengirimkan permintaan ke server nama DNS lokal untuk alamat IP www.cs.nk.edu. Server nama DNS lokal mengirimkan kueri ke server nama DNS resmi NKU untuk resolusi nama. Server nama otoritatif NKU memeriksa alamat IP sumber kueri, yaitu 8.8.8.8, alamat IP di Amerika Serikat. Karena itu, server nama DNS otoritatif NKU mengembalikan alamat IP server 2 (192.122.1.10), karena server 2 lebih dekat ke alamat IP sumber. Sayangnya untuk pengguna ini, server 2 lebih jauh dari klien daripada server 1. Masalahnya di sini adalah server nama DNS otoritatif NKU hanya mengetahui alamat IP server nama DNS lokal, bukan klien. Kebijakan penyeimbangan beban geolokasi masuk akal jika klien dan server nama DNS lokalnya dekat, tetapi dalam situasi ini, kebijakan dikalahkan, karena tidak menyadari bahwa klien menggunakan server nama lokal yang lebih jauh.

Untuk mengatasi masalah ini, kita dapat melakukan penyeimbangan muatan di ujung server nama DNS lokal alih-alih melakukannya di server nama DNS resmi. Gambar 7.41

memberikan contoh di mana load balancing ditangani oleh server nama DNS lokal. Perbedaan utama di sini adalah bahwa server nama DNS otoritatif NKU mengembalikan alamat IP dari dua server ke server nama DNS lokal. Server nama DNS lokal kemudian memilih server yang lebih baik untuk klien, berdasarkan lokasi. Karena server nama DNS lokal mengetahui alamat IP klien, itu dapat membuat keputusan berdasarkan informasi bahwa server 1 memiliki kedekatan yang lebih dekat dan karenanya merupakan pilihan yang lebih baik.

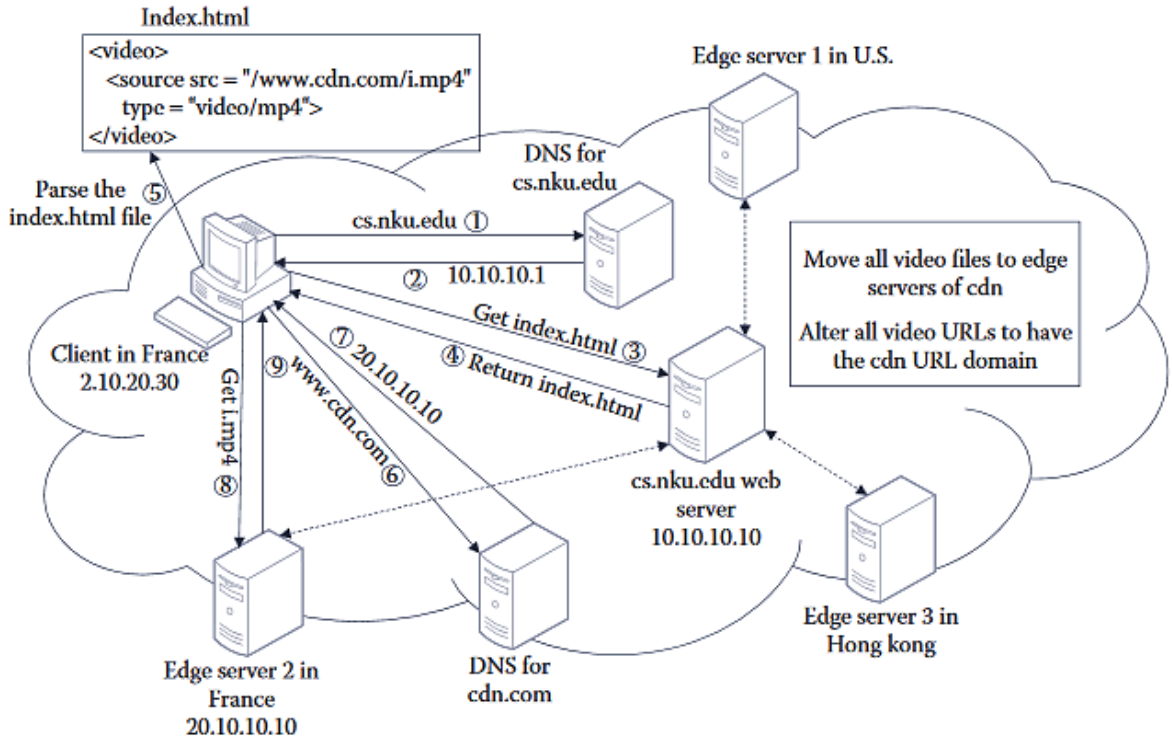


Gambar 7.41 Load balancing di server nama DNS lokal.

7.4 JARINGAN DISTRIBUSI KONTEN BERBASIS SISTEM NAMA DOMAIN

Jaringan distribusi konten (CDN) memainkan peran penting dalam infrastruktur web saat ini. CDN disiapkan oleh penyedia distribusi konten (CDP) untuk menyediakan berbagai bentuk konten di lokasi geografis yang strategis di seluruh dunia. Mereka melakukannya dengan menyediakan server tepi dengan konten yang direplikasi atau di-cache. Permintaan untuk beberapa konten mungkin dapat dikirimkan oleh server edge, yang lebih dekat daripada server web tujuan. CDP mencakup organisasi seperti Akamai, Amazon, Verizon, dan Microsoft dan di antara CDN yang tersedia adalah Amazon's CloudFront, Microsoft's Azure CDN, Verizon's EdgeCast CDN, dan CloudFlare. Penyedia konten internet (ICP) berlangganan layanan CDP dan mereplikasi konten dari server web mereka ke server edge CDP.

Apa itu server tepi? Ini adalah server yang berada di antara jaringan. Mereka mungkin ada di pinggiran antara tulang punggung Internet dan LAN yang mengisi sebagian besar Internet, atau mereka mungkin berada di batas antara jaringan pribadi dan koneksi Internetnya. Secara umum, server tepi dapat melakukan keamanan untuk jaringan pribadi (misalnya, sebagai firewall) atau menangani penyeimbangan muatan. Namun, dalam konteks ini, edge server berfungsi sebagai cache untuk konten yang mungkin diinginkan untuk kumpulan klien atau jaringan klien.



Gambar 7.42 CDN berbasis DNS.

Server tepi bersifat transparan bagi klien yang mengaksesnya. Klien menggunakan URL situs web ICP mereka (server web asli) untuk mengakses konten. Dengan demikian, URL harus dialihkan secara transparan ke server edge terdekat yang berisi duplikasi konten. Bagaimana kita mengatur ini? Salah satu pendekatannya adalah menggunakan DNS untuk mengalihkan permintaan klien ke server edge di CDN. Gambar 7.42 menunjukkan situs web cs.nku.edu, dengan sebagian kontennya disalin ke server edge di jaringan CDN cdn.com. Secara khusus, file video direplikasi dan ditempatkan di server tepi.

Mari kita bayangkan halaman index.html server web memiliki kode HTML berikut.

```
<video id="video" width="420">
     <source src="http://cs.nku.edu/1.mp4" type="video/mp4">
</video>
```

Pengalihan untuk file 1.mp4 dilakukan dengan mengganti cs.nku.edu dengan www.cdn.com, membuat kode HTML baru, seperti yang ditunjukkan di bawah ini:

```
<video id="video" width="420">
     <source src="http://www.cdn.com/1.mp4" type="video/mp4">
</video>
```

Mari kita pertimbangkan secara lebih spesifik bagaimana ini bekerja sehubungan dengan Gambar 7.42.

1. Klien meminta file index.html dari situs web cs.nku.edu. Pertama, mengirimkan permintaan DNS ke server nama DNS yang bertanggung jawab atas domain cs.nku.edu untuk menyelesaikan alamat.

2. Server nama DNS untuk domain cs.nku.edu mengembalikan alamat IP 10.10.10.10.
3. Klien mengirim permintaan HTTP ke server web (10.10.10.10) untuk index.html.
4. Server web mengembalikan halaman index.html.
5. Browser klien mem-parsing file index.html dan menemukan tautan video tersemat untuk www. cdn.com/1.mp4.
6. Klien mengirimkan kueri DNS untuk alamat IP www.cdn.com ke server nama DNS otoritatif yang sesuai untuk domain cdn.com.
7. Saat server nama DNS otoritatif menerima kueri, ia mengekstrak alamat IP klien. Itu menghitung jarak antara klien dan setiap server tepi yang berisi salinan file 1.mp4. Ini mengembalikan alamat IP dari server tepi terdekat ke klien. Pada gambar, edge server terdekat adalah Edge server 2 dengan alamat IP 20.10.10.10.
8. Klien mengirimkan permintaan HTTP ke server Edge 2 untuk video 1.mp4.
9. Edge server 2 mengembalikan video ke klien.

Dalam contoh, klien yang meminta tautan video tersemat dari www.cdn.com/1.mp4 permintaan DNS-nya dialihkan ke domain cdn.com, sehingga permintaan video akan dilayani oleh server edge di CDN jaringan. Pendekatan lain untuk merutekan permintaan klien ke jaringan CDN adalah dengan menggunakan catatan CNAME untuk menentukan nama domain alternatif. Untuk contoh ini, kami dapat menentukan data CNAME di file zona otoritatif untuk domain cs.nku.edu sebagai berikut:

```
cs.nku.edu. 10800 IN CNAME www.cdn.com.
```

Saat klien meminta cs.nku.edu/1.mp4, catatan CNAME memicu pencarian DNS di www. cdn.com. Pencarian diserahkan ke server DNS otoritatif untuk domain www.cdn.com, sehingga permintaan video dilayani oleh server tepi. Perhatikan bahwa meskipun ini membutuhkan lebih banyak waktu untuk menangani kueri DNS, pada akhirnya dibutuhkan lebih sedikit waktu untuk memenuhi permintaan karena file mp4 besar lebih dekat jaraknya (kami akan berasumsi bahwa file mp4 ini akan ditransmisikan dalam jumlah banyak, mungkin ribuan atau lebih, paket).

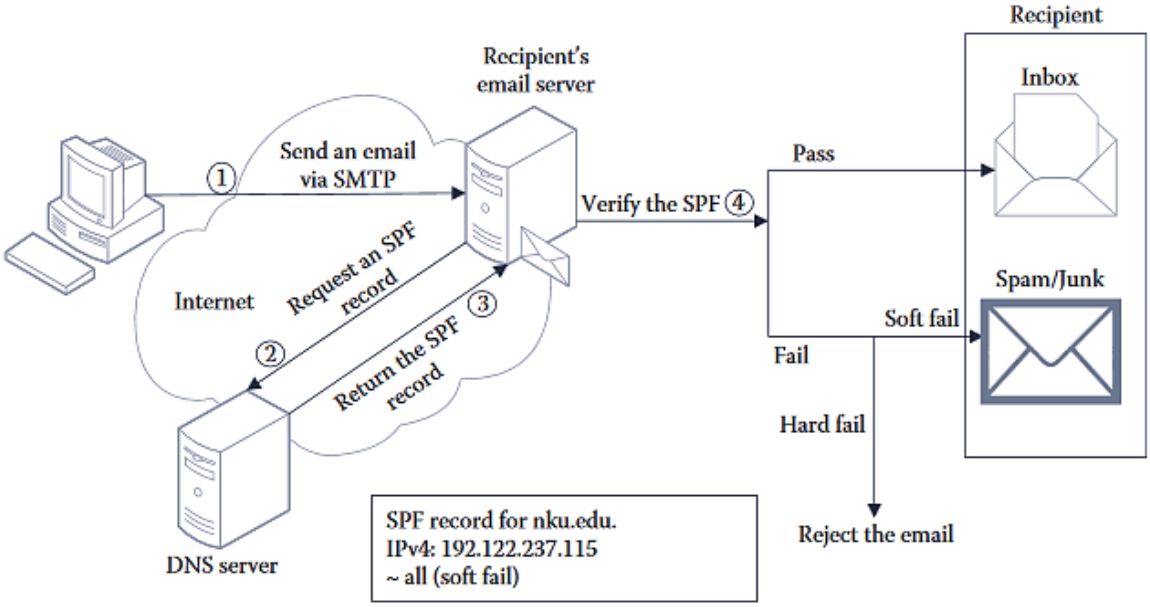
7.5 PENCEGAHAN SPAM BERBASIS SISTEM NAMA DOMAIN

Data Sender Policy Framework (SPF) adalah jenis data DNS yang mengidentifikasi server mana yang berwenang mengirim email untuk domain. Catatan SPF dapat digunakan untuk mencegah spammer mengirim email dengan alamat Dari yang dipalsukan. Di dalam domain tertentu, server nama DNS otoritatif akan menyimpan data SPF, yang mencantumkan server email resmi yang dapat mengirim email dari domain tersebut. Setelah menerima email dari domain tersebut, server email penerima dapat menghubungi server nama DNS otoritatif untuk mendapatkan data SPF. Jika email tersebut berasal dari salah satu server yang tercantum dalam data SPF, email tersebut dianggap sebagai email yang valid. Namun, jika email tersebut berasal dari server yang tidak ada dalam daftar ini, email tersebut akan ditolak sebagai spam. Server email juga dapat menolak email dari domain yang tidak memiliki catatan

SPF yang tersedia, karena mereka tidak dapat memverifikasi apakah email tersebut valid atau tidak. Gambar 7.43 mengilustrasikan alur pemrosesan SPF.

Mari kita lihat Gambar 7.43 untuk melihat cara kerjanya. Kami berasumsi bahwa gambar mewakili domain nku.edu dan menyimpan data SPF untuk server email domain dengan alamat IP 192.122.237.115.

1. Email dikirim melalui SMTP dari abc@nku.edu ke beberapa penerima domain lain. Dalam hal ini, pengirim memiliki nama pengguna abc, dan domain asalnya adalah nku.edu.
2. Server email penerima menerima email dan mengirimkan kueri DNS ke domain nku.edu untuk data SPF.
3. Server nama DNS otoritatif nku.edu menerima permintaan, mengambil catatan SPF untuk domain nku.edu, dan mengirimkan respons DNS ke pemohon. Data SPF menunjukkan bahwa server email resmi untuk domain tersebut memiliki alamat IPv4 192.122.237.115.
4. Server email penerima membandingkan alamat IP pengirim dengan alamat dalam respons DNS (dalam hal ini, satu alamat IP). Jika ada kecocokan, maka email diterima dan ditempatkan di kotak masuk penerima. Jika tidak, verifikasi SPF gagal. Jika aturan gagal lunak ditentukan dalam catatan SPF, maka email tetap diterima tetapi ditandai sebagai gagal oleh server email. Menurut kebijakan server email, email yang ditandai sebagai gagal dapat ditempatkan ke dalam folder spam atau sampah penerima. Jika aturan gagal keras ditentukan dalam data SPF, email akan ditolak.



Gambar 7.43 Alur pemrosesan SPF.

Untuk lebih memahami aturan soft/hard fail dari data SPF, mari kita lihat contoh dari data SPF. Entri ini seperti catatan file zona DNS lainnya. Dalam hal ini, kami mengambil catatan dari server nama otoritatif google.com dengan menggunakan perintah nslookup. Secara khusus, kami menjalankan nslookup –type=txt google.com. Perhatikan bahwa kami menentukan jenis txt, artinya respons akan mencantumkan semua catatan yang dilambangkan sebagai teks. Responsnya ditunjukkan sebagai berikut:

```
Non-authoritative answer:
      google.com    text = "v=spf1 include:_spf.google.com
      ip4:216.73.93.70/31 ip4:216.73.93.72/31 ~all"
```

Bagian pertama dari data SPF, v=spf1, menentukan versi yang digunakan, sehingga server email mengetahui bahwa catatan teks ini digunakan untuk SPF. Empat bagian data SPF lainnya menentukan empat pengujian untuk memverifikasi pengirim email. Tanggapannya meliputi: _ spf.google.com adalah arahan penyertaan. Arahan ini menyertakan server _spf.google.com dalam daftar yang diizinkan untuk mengirim email. Selanjutnya, kami memiliki dua rentang alamat IP yang ditentukan: ip4:216.73.93.70/31 dan 216.73.93.72/31. Ini menentukan alamat IP dari server email yang valid dalam domain ini. Entri terakhir menentukan "~all", yang berarti bahwa kegagalan harus dianggap sebagai kegagalan lunak. Gagasan di balik "semua" adalah bahwa itu adalah default atau selalu cocok jika a da alamat IP yang terdaftar sebelumnya tidak cocok. The ~ menunjukkan kegagalan lunak. Dengan demikian, alamat IP apa pun yang tidak cocok dengan yang terdaftar harus dianggap gagal total. Di tempat ~semua, entri –semua akan menunjukkan bahwa alamat IP lainnya adalah kegagalan keras.

Pengujian (dua rentang alamat IP diikuti oleh ~all) dijalankan sesuai urutan yang ditentukan dalam data SPF. Saat ada kecocokan, pemrosesan berhenti dan mengembalikan operan atau kegagalan. Beberapa administrator mungkin tidak peduli untuk menentukan daftar SPF. Dalam kasus seperti itu, mereka mungkin menyertakan satu catatan SPF yang menyatakan +all. Tidak seperti ~all, yang secara default menunjukkan soft fail, penggunaan + menunjukkan sukses. Jadi, +all adalah default yang mengatakan bahwa setiap alamat IP adalah alamat yang valid.

Selain mekanisme verifikasi yang ditunjukkan pada contoh, mekanisme lain yang dapat digunakan untuk memverifikasi pengirim adalah sebagai berikut. Pertama, ip6 menggunakan alamat IP versi 6 untuk verifikasi. Misalnya, "v=spf1 ip6:2001:cdba::3257:9652 -all" memungkinkan host dengan alamat IP 2001:cdba::3257:9652 mengirim email untuk domain tersebut. Semua host lain tidak diizinkan mengirim email untuk domain tersebut. Entri a menggunakan catatan DNS A untuk verifikasi. Dengan "v=spf1 a –all", semua host yang tercantum dalam data A file zona dapat mengirim email untuk domain tersebut. Semua host lain tidak diizinkan mengirim email untuk domain tersebut. Demikian pula, entri mx akan digunakan untuk menentukan bahwa email legal berasal dari server email (data MX) dari file zona. Terakhir, entry ptr akan menggunakan record PTR untuk verifikasi. Misalnya, "v=spf1 ptr -all" memungkinkan semua host yang dipetakan terbalik dalam domain untuk mengirim email untuk domain tersebut. Semua host lain tidak diizinkan mengirim email untuk domain tersebut.

BAB 8
BIND DAN DHCP

Dalam bab ini, kami menyatukan beberapa konsep dari bab TCP/IP mengenai pengalamatan Protokol Internet (IP), dengan materi dari bab DNS. Secara khusus, kami memeriksa cara menginstal, mengkonfigurasi, dan menjalankan dua jenis server perangkat lunak Internet yang sangat penting: server nama domain dan server Protokol Konfigurasi Host Dinamis (DHCP). Kami menjelajahi program BIND untuk yang pertama dan server DHCP Internet Systems Consortium (ISC) untuk yang terakhir. Kami mengeksplorasi konsep terkait lainnya dan masalah implementasi saat kami menelusuri bab ini.

8.1 PENDAHULUAN

BIND adalah nama resmi untuk server perangkat lunak Domain Name System (DNS) sumber terbuka yang berjalan di sebagian besar sistem Unix/Linux. Program sebenarnya yang Anda jalankan disebut bernama (singkatan dari nama daemon). BIND adalah implementasi sumber terbuka yang paling banyak digunakan untuk server nama DNS. Awalnya dirancang di University of California Berkeley pada awal 1980-an dan telah mengalami sejumlah revisi. Versi terbaru disebut BIND 10, yang pertama kali dirilis pada tahun 2014; namun, BIND 9, yang dikendalikan oleh ISC, mungkin lebih stabil dan populer, karena BIND 10 tidak lagi didukung oleh ISC. BIND 9 mendukung Ekstensi Keamanan DNS (DNSSEC), nsupdate, pengalamatan Internet Protocol versi 6 (IPv6), layanan rndc Unix/Linux, dan tanda tangan transaksi.

Memasang BIND

Kode sumber untuk BIND tersedia di http://www.isc.org/downloads. Versi stabil saat ini pada saat penulisan ini adalah 9.11.0; namun, dalam bab ini, kami membahas 9.10.2 yang sedikit lebih tua (ini sebagian karena 9.11, meskipun dianggap sebagai versi stabil, masih diuji). Pada bagian ini, kita akan melakukan pengunduhan, konfigurasi, dan penginstalan perangkat lunak. Banyak paket berbeda tersedia untuk diunduh dan diinstal. Anda mungkin ingin mengikuti petunjuk di lampiran untuk menginstal versi aman dengan menggunakan checksum tanda tangan dan kriptografi. Di sini, kami hanya akan mengunduh dan menginstal versi tanpa keamanan ini. Jika Anda mendapatkan versi 9.10.2, Anda akan mengeluarkan perintah berikut untuk mengekstrak dan mengekstrak paket. Tentu saja, ubah perintah tar Anda berdasarkan paket tertentu yang Anda unduh.

```
tar -xzf bind-9.10.2-P2.tar.gz
```

Perintah tar membuat subdirektori baru, bind-9.10.2-P2 (namanya akan didasarkan pada paket yang Anda unduh, jadi milik Anda mungkin sedikit berbeda). Dalam memeriksa direktori ini, Anda akan menemukan isinya seperti yang ditunjukkan pada Gambar 8.1. Item bin, contrib, doc, docutil, lib, libtool.m4, make, unit, util, dan win32utils adalah subdirektori. Item lainnya adalah file teks American Standard Code for Information Interchange (ASCII). Ini termasuk configure, install-sh, dan mkin-stalldirs, yang merupakan skrip configure dan

instalasi. File yang namanya dikapitalisasi (mis., PERUBAHAN dan README) adalah file bantuan. Anda harus meninjau file README sebelum mencoba instalasi.

Instalasi perangkat lunak open source bisa sangat sederhana jika Anda ingin menggunakan instalasi default. Langkah-langkahnya adalah pertama-tama buat skrip makefile dengan mengeksekusi skrip configure. Dari direktori ini, ketik ./configure. Menjalankan skrip ini mungkin memakan waktu beberapa menit, tergantung pada kecepatan komputer Anda. Setelah selesai, Anda mengkompilasi kode sumber dengan mengeksekusi makefile. Jangan ketik ./makefile, tetapi gunakan perintah Unix/Linux make. Terakhir, untuk menyelesaikan penginstalan (mis., memindahkan file ke lokasi tujuan yang tepat sambil juga membersihkan file yang dibuat sementara), ketik make install. Baik make dan make install dapat memakan waktu beberapa menit untuk dieksekusi, seperti dengan eksekusi skrip ./configure.

```
acconfig.h          config.sub          HISTORY                 Makefile.in
aclocal.m4          config.threads.in   install-sh              mkinstalldirs
Atffile             configure           isc-config.sh.1         README
bin                 configure.in        isc-config.sh.docbook   srcid
bind.keys           contrib             isc-config.sh.html      unit
bind.keys.h         COPYRIGHT           isc-config.sh.in        util
CHANGES             doc                 lib                     version
config.guess        docutil             libtool.m4              win32utils
config.h.in         FAQ                 ltmain.sh
config.h.win32      FAQ.xml             make
```

Gambar 8.1 Isi direktori bind-9.10.2-P2.

Konfigurasi default untuk BIND akan menempatkan file di berbagai lokasi dari struktur direktori Unix/Linux Anda. Anda dapat mengontrol penginstalan dengan menyediakan skrip konfigurasi dengan berbagai jenis parameter. Untuk melihat semua opsi, ketik ./configure --help.

Kami akan berasumsi bahwa instalasi Anda menempatkan semua file di satu lokasi bernama/usr/local/bind. Selain itu, kami akan menerapkan opsi --tanpa-openssl, sehingga instalasi BIND ini tidak akan memiliki paket openSSL, yang diinstal secara default. Kita dapat melakukan perubahan ini dari default dengan menjalankan perintah berikut, diikuti dengan make dan make install, seperti sebelumnya.

```
./configure --prefix=/usr/local/bind --without-openssl
```

Setelah penginstalan selesai menggunakan langkah-langkah konfigurasi, buat, dan instal di atas, kami akan menemukan sebagian besar BIND berada di bawah /usr/local/bind. Direktori ini akan mencakup enam subdirektori, seperti yang dijelaskan pada Gambar 8.1. Direktori yang paling penting adalah sbin, yang menyertakan program yang dapat dieksekusi, dan var, yang akan menyimpan file konfigurasi zona kami.

Directory	Usage	Sample Files
bin	Executable programs to query a DNS name server	dig, host, nslookup, nsupdate
etc	Configuration files	named.conf, bind.keys
include	C header files (.h files) of shared libraries	Various, located under subdirectories such as bind9, dns, and isc
lib	C static library files (.a files)	libbind9.a, libdns.a, libisc.a
sbin	Executable programs	named, rndc, dnssec-keygen
share	BIND man pages	Various
var	BIND data files	Initially empty

Gambar 8.1 Direktori BIND setelah Instalasi

MENGKONFIGURASI BIND

File konfigurasi utama BIND disebut named.conf (ingat bahwa program sebenarnya dikenal sebagai nama daemon nama). named.conf berisi dua jenis item: arahan dan komentar. Arahan menentukan bagaimana BIND akan berjalan dan lokasi file data zona server Anda. Komentar adalah pernyataan yang menjelaskan peran arahan dan/atau arahan yang harus diisi. Beberapa arahan juga dapat dikomentari, artinya muncul sebagai komentar. Sebagai administrator jaringan atau sistem, Anda mungkin memutuskan untuk mengomentari beberapa arahan default atau Anda mungkin menempatkan arahan dalam komentar yang mungkin ingin Anda sertakan nanti. Komentar dalam file skrip biasanya dimulai dengan karakter # tetapi juga dapat dilambangkan seperti komentar pemrograman C dengan menggunakan // sebelum komentar atau menyematkan komentar di dalam notasi /* dan */. Pendekatan terakhir ini berguna untuk komentar multi-baris. Sebagian besar atau semua komentar yang Anda lihat di name.conf akan dimulai dengan karakter #.

Tata letak umum dari file named.conf adalah sebagai berikut. Item acl, options, logging, zone, dan include adalah arahan file konfigurasi.

```
acl name{…};
options {...};
logging {...};
zone {...};
include...;
```

Singkatan acl adalah singkatan dari daftar kontrol akses. Acl adalah pernyataan yang mendefinisikan daftar alamat yang cocok. Artinya, melalui pernyataan acl, kita mendefinisikan satu atau lebih alamat yang akan dirujuk melalui nama string. Acl yang ditentukan kemudian digunakan dalam berbagai opsi untuk menentukan, misalnya, siapa yang dapat mengakses server BIND. Kami secara singkat melihat penggunaan pernyataan acl di bagian ini tetapi akan menjelajahinya lebih detail saat kami memeriksa server proxy Squid secara detail di Bab 10. Struktur direktif acl di BIND adalah sebagai berikut:

```
acl acl_name {
     address_match_list
};
```

Nilai address_match_list akan menjadi salah satu dari beberapa kemungkinan nilai. Pertama, ini bisa berupa alamat IP atau awalan alamat IP (ingat awalan dari Bab 3). Misalnya, jika kita ingin mendefinisikan localhost, kita dapat menggunakan acl berikut:

```
acl localhost {127.0.0.1;};
```

Kami mungkin mendefinisikan seluruh subnet, seperti berikut ini:

```
acl ournet {10.2.0.0/16;};
```

Kita dapat menentukan beberapa item dalam address_match_list, seperti berikut ini, di mana kita menentukan tiga alamat IP spesifik dan dua subnet:

```
acl ourhosts { 172.16.72.53; 192.168.5.12;
     10.11.14.201; 10.2.3.0/24; 10.2.4.0/24; };
```

Ada juga empat kata yang telah ditentukan sebelumnya yang juga dapat digunakan, ditunjukkan sebagai berikut:

- **any**—cocok dengan semua host di jaringan
- **localhost**—cocok dengan semua antarmuka jaringan pada sistem lokal
- **jaringan lokal**—cocok dengan semua host di jaringan server
- **tidak ada**—tidak ada host yang cocok

Tabel 8.2 Subpernyataan Pilihan Umum

Substatement	Arti
allow-query {address_match_list};	Menentukan alamat IP yang diizinkan untuk mengirim kueri DNS ke server BIND ini, dengan default semua
allow-query-cache {address_match_list};	Menentukan host yang diizinkan untuk mengeluarkan kueri yang mengakses cache server BIND ini (default untuk subpernyataan ini bergantung pada nilai untuk perbolehkan-rekursi)
allow-recursion {address_match_list};	Menentukan host yang diizinkan untuk mengeluarkan kueri rekursif ke server BIND ini, dengan default semua
allow-transfer {address_match_list};	Menentukan host yang diizinkan untuk mentransfer file data zona dari server BIND ini, dengan default semua
allow-update {address_match_list};	Menentukan host yang diizinkan mengirimkan pembaruan dinamis untuk zona master ke DNS Dinamis, dengan default tidak ada
blackhole {address_match_list};	Menentukan host yang tidak diizinkan untuk mengkueri server BIND ini
directory "path_name";	Menentukan direktori kerja server BIND, di mana path_name dapat berupa jalur absolut atau relatif yang default ke direktori tempat server BIND dimulai

dnssec-enable (yes | no);	Menunjukkan apakah DNS aman sedang digunakan, dengan default ya
dnssec-validation (yes | no);	Menunjukkan apakah server BIND caching harus mencoba memvalidasi balasan dari zona yang diaktifkan (ditandatangani) DNSSEC, dengan default ya
dump-file "path_name";	Menentukan jalur absolut tempat server BIND membuang cache-nya sebagai respons terhadap perintah "rndc dumpdb", dengan default file bernama_dump.db di bawah direktori yang ditentukan oleh substatmen direktori
forward (only | first);	Menentukan perilaku penerusan server BIND, di mana hanya berarti bahwa BIND tidak akan menjawab kueri apa pun, meneruskan semua kueri, sedangkan yang pertama akan meminta BIND meneruskan kueri terlebih dahulu ke server nama yang tercantum dalam subpernyataan penerusan, dan jika tidak ada kueri yang dijawab, maka BIND akan menjawab
forwarders {ip_addr [port:ip_port]; [...]};	Menentukan daftar alamat IP untuk server nama tempat permintaan kueri akan diteruskan; perhatikan bahwa alamat port bersifat opsional, seperti forwarder {10.11.12.13; 10.11.12.14 porta 501;};
listen-on [port ip_ port] {address_match_list};	Menentukan port dan alamat IP yang akan didengarkan oleh server BIND untuk kueri DNS, di mana port bersifat opsional, dan jika dihilangkan, defaultnya adalah 53; ada juga listen-on-v6 untuk alamat IPv6
notify yes | no | explicit;	Mengontrol apakah server ini akan memberi tahu server DNS budak saat zona diperbarui, di mana secara eksplisit hanya akan memberi tahu server budak yang ditentukan dalam daftar beri tahu juga dalam pernyataan zona
pid-file "path_name";	Menentukan lokasi file ID proses yang dibuat oleh server BIND saat memulai; defaultnya, file tersebut adalah /var/run/named/named.pid
recursion yes | no;	Menentukan apakah server akan mendukung kueri rekursif
rrset-order order {fixed | random | cyclic;};	Menentukan urutan di mana beberapa rekaman dari jenis yang sama dikembalikan, di mana rekaman pengembalian tetap dalam urutan yang ditentukan dalam file zona, rekaman pengembalian acak dalam urutan acak, dan rekaman pengembalian siklik dengan cara round-robin
sortlist {address_match_list;...};	Menentukan serangkaian pasangan daftar pencocokan alamat, di mana alamat pertama dari setiap pasangan digunakan untuk mencocokkan alamat IP klien, dan jika ada kecocokan, server mengurutkan daftar alamat yang dikembalikan sehingga setiap alamat yang cocok dengan yang kedua address_match_list dikembalikan terlebih dahulu, diikuti oleh semua alamat yang cocok dengan address_match_list ketiga, dan seterusnya; misalnya, daftar urutan yang ditunjukkan di bawah ini akan menyebabkan permintaan klien apa pun dari 10.2.56.0/24 untuk

	mengembalikan alamat dari 10.2.56.0/24 terlebih dahulu, diikuti oleh alamat tersebut dari 10.2.58.0/24 selanjutnya, sedangkan klien bukan dari 10.2.56.0/ 24 akan mengembalikan alamat berdasarkan pernyataan rrset-order
statistics-file "path_name";	Menentukan lokasi alternatif untuk file statistik BIND, defaultnya adalah /var/named/named.stats

Pernyataan opsi mendefinisikan opsi yang ditentukan secara global di seluruh server BIND serta nilai default. Hanya akan ada pernyataan opsi tunggal di named.conf, tetapi Anda dapat menyertakan sejumlah opsi dalam pernyataan ini. Sintaks pernyataan opsi ditunjukkan di bawah ini. Kami menggunakan istilah substatement di sini untuk menunjukkan opsi yang ditentukan dalam pernyataan itu. Gambar 8.2 menjelaskan opsi yang lebih umum digunakan. Ada lebih dari 100 pilihan yang tersedia. Lihat dokumentasi BIND untuk melihat semuanya. Perhatikan bahwa address_match_list dapat berupa acl yang ditentukan sebelumnya atau salah satu entri yang sama yang diizinkan dalam pernyataan acl (satu atau lebih alamat IP, prefiks IP, atau salah satu dari empat kata yang ditentukan: all, none, localhost, dan localnets).

```
options {
     substatement1;
     substatement2;
     …
};
```

Berikut ini adalah contoh dari beberapa pernyataan acl yang diikuti oleh pernyataan pilihan. Kami melihat beberapa daftar kontrol akses berbeda yang dibuat untuk digunakan di beberapa subpernyataan dalam pernyataan opsi.

```
acl me { 127.0.0.1; 10.11.12.14; };
acl us {10.11.0.0/16; };
acl them { 1.2.3.4; 1.2.3.5; 3.15.192.0/20; };
acl all { any; };

options {
     listen-on port 53 { me; };
     directory "/var/named";
     allow-query { all; };
     allow-access-cache { me; };
     allow-recursion { me; us; };
     blackhole { them; };
     forward first;
     forwarders { 10.11.12.15 port 501;
           10.11.12.16 port 501; };
     notify explicit;
     recursion yes;
};
```

Arahan lain menentukan bagaimana server BIND akan mencatat permintaan. Untuk ini, kami menggunakan pernyataan logging. Seperti opsi, Anda akan memiliki satu direktif logging, tetapi direktif tersebut dapat menentukan banyak saluran logging yang berbeda. Saluran

dapat ditentukan untuk menulis jenis acara yang berbeda ke lokasi yang berbeda. Sintaks untuk logging agak rumit dan ditampilkan sebagai berikut:

```
logging {
   [ channel channel_name {
     ( file path_name
         [ versions ( number | unlimited ) ]
         [ size size_spec ]
       | syslog syslog_facility
         | stderr
         | null );
     [ severity (critical | error | warning | notice | info |
               debug [ level ] | dynamic ); ]
     [ print-category yes | no; ]
     [ print-severity yes | no; ]
     [ print-time yes | no; ]
   }; ]
   [ category category_name {
     channel_name ; [ channel_name ; ... ]
   }; ]
   ...
};
```

Kami mengeksplorasi subpernyataan yang berbeda untuk direktif logging pada Gambar 8.3.

Gambar 8.3 Pernyataan Umum untuk Petunjuk Logging

Substatement	Arti
channel channel_name	Menentukan saluran untuk menentukan lokasi dan jenis pesan yang akan dikirim; nama_saluran yang ditentukan dapat digunakan dalam subpernyataan kategori (lihat di bawah)
file "path_name"	Lokasi (nama dan jalur) file log yang akan dikirimi pesan oleh saluran ini
versions (number | unlimited)	Menentukan jumlah versi file log yang dipertahankan untuk tujuan rotasi log; ketika sebuah log menjadi terlalu penuh (lihat ukuran di bawah), itu dipindahkan ke filename.0, dengan filename.0 diputar ke filename.1, dll; jika Anda menggunakan angka (mis., 3), ini menentukan berapa banyak file log yang dipertahankan dalam rotasi ini; tak terbatas mempertahankan semua file log
size size_spec	size_spec menentukan ukuran file log sebelum rotasi dilakukan; nilai ini adalah bilangan bulat dalam byte, tetapi Anda juga dapat menggunakan k (kilobyte), m (megabyte), atau g (gigabyte)
syslog syslog_ facility	Subpernyataan ini digunakan untuk merujuk ke layanan Linux/Unix tertentu yang ingin Anda gunakan untuk masuk, seperti daemon syslogd
stderr	Jika stderr ditentukan, maka semua pesan kesalahan standar diarahkan ke saluran ini
null	Jika null ditentukan, maka semua pesan dari saluran ini diarahkan ke /dev/null; perhatikan bahwa file subpernyataan, syslog, stderr, dan null saling eksklusif (hanya satu yang diizinkan dalam pernyataan saluran apa pun)
severity (critical | error | warning | notice | info | debug [level] | dynamic)	Menetapkan tingkat keseriusan bahwa tindakan harus berada pada atau di atasnya untuk dicatat; misalnya, jika disetel ke error, hanya pesan error dan kritis yang dicatat, sedangkan dengan info, sebagian besar pesan nondebugging dicatat

print-time yes | no	Menunjukkan apakah tanggal/waktu ditulis ke saluran
print-severity yes | no	Menunjukkan apakah tingkat keparahan ditulis ke saluran
print-category yes | no	Menunjukkan apakah nama kategori ditulis ke saluran
category category_name	Menentukan kategori apa yang dicatat ke saluran; dua contoh nama_kategori adalah default (mencatat semua nilai yang tidak didefinisikan secara eksplisit dalam pernyataan kategori) dan kueri (mencatat semua transaksi kueri)

Berikut ini adalah contoh direktif logging. Dalam hal ini, tiga saluran ditentukan. Kami melihat bahwa channel1 berputar hingga empat versi dengan ukuran 100 MB dan menyertakan waktu dan tingkat keparahan untuk setiap pesan yang levelnya peringatan ke atas, sedangkan pesan yang berada pada level info ditulis ke daemon syslog untuk logging, dan semua pesan lainnya akan dibuang.

```
logging {
     channel channel1 {
          file "bind.log" versions 4 size 100m;
          severity warning;
          print-time yes;
          print-severity yes;
     }
     channel channel2 {
          severity info;
          syslog syslogd;
     }
     channel everything_else { null; };
};
```

Selanjutnya, kita memeriksa pernyataan zona. Ini adalah pernyataan zona yang menentukan domain DNS dan informasi tentang domain tersebut seperti lokasi file zona dan apakah server DNS ini adalah master atau budak untuk zona ini. Seperti dua arahan sebelumnya, arahan zona memiliki sintaks di mana subpernyataan terdaftar. Gambar 8.4 berisi subpernyataan yang lebih umum. Sintaks untuk pernyataan zona adalah sebagai berikut. zone_name menentukan nama zona. Kelas adalah jenis jaringan, default ke IN (Internet) jika dihilangkan.

```
zone "zone_name" [class] {
     substatement1;
     substatement2;
     …
};
```

Selain sub-pernyataan yang tercantum dalam Gambar 8.4, pernyataan zona juga mengizinkan sub-pernyataan seperti perbolehkan-permintaan, perbolehkan-transfer, dan perbarui-perbolehkan, seperti yang dibahas sebelumnya sehubungan dengan opsi. Deskripsi lengkap tentang subpernyataan zona diberikan dalam dokumentasi BIND, bersama dengan deskripsi sintaks yang diperlukan untuk subpernyataan seperti master dan peran tipe lain yang tidak dijelaskan dalam Gambar 8.4. Berikut ini adalah contoh pernyataan dua zona. Dalam hal ini, server DNS ini adalah server master untuk satu zona dan budak untuk zona kedua.

```
zone "cit.nku.edu" {
     type master;
     file "cit.nku.edu.db";
     notify yes;
     allow-transfer { 10.11.12.14; };
};

zone "csc.nku.edu" {
     type slave;
     masters { 10.11.12.15; };
     file "csc.nku.edu.bk"; };
};
```

Jenis direktif terakhir adalah pernyataan include. Pernyataan include menyisipkan file konfigurasi lain ke dalam file konfigurasi ini di lokasi pernyataan include di dalam file ini. Penggunaan pernyataan include memungkinkan file konfigurasi berukuran lebih kecil sehingga file ini lebih mudah untuk diedit dan di-debug. Anda juga dapat dengan mudah mengomentari pernyataan penyertaan jika Anda tidak lagi ingin konfigurasi tertentu dimuat. Misalnya, Anda dapat menempatkan beberapa definisi zona dalam file terpisah. Jika zona tersebut tidak akan dilayani untuk saat ini, Anda akan mengomentari pernyataan sertakan yang akan memuat file terpisah itu.

Mari kita pertimbangkan situasi lain untuk memisahkan pernyataan zona menjadi beberapa file dan menggunakan pernyataan sertakan. Untuk organisasi tertentu, zona dikelola oleh beberapa departemen berbeda. Setiap departemen menempatkan pernyataan konfigurasinya dalam file konfigurasinya sendiri. Melalui penggunaan pernyataan include di named.conf, masing-masing dapat ditambahkan saat BIND dikonfigurasi. Hal ini memungkinkan setiap departemen untuk mengedit filenya sendiri, tanpa harus memiliki akses ke name.conf. Selain itu, data konfigurasi sensitif seperti kunci enkripsi dapat ditempatkan di file terpisah dengan izin terbatas.

Pernyataan sertakan memiliki sintaks sederhana, sertakan "nama file" ;, seperti pada tiga perintah berikut. Perhatikan bahwa dalam kasus dua yang terakhir, file zona tidak terletak di dalam direktori BIND di /usr/local/bind. File include ketiga berisi kunci enkripsi rndc.

```
include "named.aux";
include "/home/zappaf/mybind_zone_data.conf";
include "/etc/rndc.key";
```

Sejumlah arahan lain tersedia untuk named.conf. Ini dijelaskan dalam Gambar 8.5. Contoh mengikuti tabel.

Berikut ini adalah beberapa arahan tambahan yang mungkin kita temukan di file named.conf kita (atau file include). Kami menyelingi beberapa diskusi di antara contoh-contoh.

```
controls {
     inet 127.0.0.1 port 953
     allow { 127.0.0.1; } keys { "rndc-key"; };
};
```

Saluran kontrol inet adalah soket TCP yang mendengarkan di port TCP yang ditentukan pada alamat IP yang ditentukan. Di sini, kita melihat bahwa kita mendengarkan host lokal melalui port 953. Pernyataan ini mengizinkan akses hanya oleh host lokal menggunakan kunci rndc. Jika * digunakan untuk pernyataan inet, koneksi akan diterima melalui antarmuka apa pun.

Gambar 8.5 Direktif lain untuk named.conf

Pernyataan	Arti
controls { [inet (ip_addr | *) [port ip_port] allow { address_match_list } keys { key_list };] }	Mengonfigurasi saluran kontrol untuk utilitas rndc
key key_id { algorithm string1; secret string2; };	Menentukan kunci autentikasi/otorisasi menggunakan TSIG (Transaction SIGnature); key_id (nama kunci) adalah nama domain yang mengidentifikasi kunci secara unik; string1 adalah nama simbolis dari algoritma autentikasi, dan string2 adalah rahasia yang disandikan menggunakan base64 yang digunakan oleh algoritma
lwres {…}	Jika ditentukan, server ini juga bertindak sebagai daemon penyelesai ringan; arahan ini memiliki sintaks yang rumit dan tidak tercakup di sini (lihat dokumentasi BIND)
managed-keys { name initial-key flags protocol alg key-data […] };	Menentukan akar keamanan DNSSEC untuk kunci siaga jika kunci tepercaya (lihat di bawah) telah disusupi; pernyataan ini dapat memiliki banyak kunci, di mana setiap kunci ditentukan oleh nama, algoritme, kunci awal untuk digunakan, bilangan bulat yang mewakili bendera, nilai protokol, dan string data kunci
masters	Menentukan server master jika ada beberapa master untuk domain tertentu yang dilayani server ini; sintaks untuk direktif ini sama dengan yang ditunjukkan pada Gambar 8.4, ketika master digunakan dalam pernyataan direktif zona
server ip_addr[/prefixlen] {…} ;	Menentukan opsi yang memengaruhi cara server merespons server nama jarak jauh; jika panjang awalan ditentukan, maka ini menentukan rentang alamat IP untuk banyak server; arahan ini mengizinkan sejumlah opsi yang tidak tercakup di sini (lihat dokumentasi BIND untuk detailnya)
statistics-channels { [inet (ip_addr | *) [port ip_port] [allow { address_match_list }];] […] };	Mendeklarasikan saluran komunikasi untuk informasi statistik permintaan yang datang ke host ini melalui alamat IP dan/atau nomor port yang ditentukan; daftar yang diizinkan, juga opsional, menunjukkan alamat IP mana dari data permintaan yang harus dikompilasi; Anda dapat mengatur beberapa alamat/port dan mengizinkan pernyataan
trusted-keys	Kunci tepercaya menentukan akar keamanan DNSSEC yang digunakan saat kunci publik yang diketahui tidak dapat

{ string1 number1 number2 number3 string2 […] };	diperoleh dengan aman; string1 adalah nama domain kunci, angka1 adalah bilangan bulat yang mewakili nilai untuk bendera status, angka2 adalah protokol, angka 3 adalah algoritme, dan string2 adalah representasi Base64 dari data kunci; opsional, beberapa kunci dapat dicantumkan, dipisahkan dengan spasi
view view_name [class] { match-clients { address_match_list }; match-destinations { address_match_list }; match-recursive-only (yes | no) ; [options { … }]; [zone { … }; […]]; };	Server BIND dapat merespons kueri DNS secara berbeda, berdasarkan host yang membuat permintaan DNS, seperti yang dikontrol melalui direktif tampilan; beberapa pernyataan tampilan dapat disediakan, di mana masing-masing memiliki view_name yang unik; pernyataan tampilan tertentu akan dipilih berdasarkan pernyataan tampilan yang cocok dengan alamat IP klien (daftar klien yang cocok) dan target kueri DNS (daftar tujuan yang cocok); seperti yang ditunjukkan di sini, setiap pernyataan tampilan dapat menentukan opsi dan zona spesifiknya sendiri

```
view "us" {
      match-clients { us; };
      match-destinations { us; them; };
      recursion yes;
};

view "them" {
      match-clients { them; };
      match-destinations { any; };
      recursion no;
}
```

Pernyataan tampilan pertama menetapkan bahwa server BIND kami akan berfungsi sebagai server DNS rekursif untuk setiap klien yang ditentukan dalam acl "kami" di mana alamat IP yang diminta ditentukan dalam acl "kami" atau "mereka". Namun, pernyataan tampilan kedua menetapkan bahwa server ini tidak akan bertindak sebagai server rekursif untuk setiap permintaan yang datang dari klien pada daftar acl "mereka", tidak peduli apa alamat IP permintaan itu. Kami dapat menyertakan opsi dan pernyataan zona sesuai kebutuhan, jika kami ingin menyesuaikan lebih lanjut bagaimana BIND beroperasi untuk salah satu dari kumpulan kueri ini.

```
trusted-keys {
      ourdomain.org. 0 2 6 "...";
      theirdomain.com. 0 2 6 "...";
};

managed-keys {
      "." initial-key 257 2 3 "..."
};
```

Di sini, kita melihat bahwa kita memiliki dua set kunci terpercaya: satu untuk domain kita.org dan satu lagi untuk domain mereka.com. Keduanya menggunakan algoritma 6, protokol 2, dengan flag status 0. Tanda "…" menunjukkan data kunci. Itu tidak ditampilkan di

sini karena masing-masing akan memiliki panjang beberapa baris. Kami juga memiliki satu kunci terkelola untuk digunakan jika ada kunci tepercaya yang tidak dapat dikonfirmasi karena, mungkin, kunci tersebut telah tidak valid. Kunci ini menggunakan algoritma 3 protokol 2 dan memiliki bendera status 257. Sekali lagi, data kuncinya dihilangkan.

Untuk dokumentasi tambahan tentang named.conf, lihat manual BIND dari situs web ISC BIND di www.isc.org/downloads/bind/doc. Anda juga dapat menemukan informasi melalui halaman manual named.conf (man named.conf).

Seperti disebutkan di atas saat membahas direktif zona, kita juga harus menyediakan file konfigurasi untuk setiap zona yang ditentukan. Kami melakukan ini melalui file zona. Akan ada setidaknya dua file zona yang ditentukan untuk server DNS apa pun, atau sangat mungkin lebih. Minimal, kita memerlukan file zona localhost dan file zona sebenarnya. Jika ada beberapa zona, maka ada satu file zona per zona. Selain itu, mungkin ada file zona maju, file zona mundur, dan file zona petunjuk root.

Menjalankan server BIND

Kita perlu memastikan bahwa sintaks dari named.conf dan file zona sudah benar sebelum kita memulai server BIND. Program bernama-checkconf melakukan pemeriksaan sintaks pada file konfigurasi. Itu tidak dapat mendeteksi kesalahan logis (semantik). Untuk menjalankan program ini, berikan nama file dari file konfigurasi, seperti pada named-checkconf /etc/named.conf. Tanggapan dari menjalankan name- checkconf menunjukkan bahwa kesalahan sintaks ditemukan (tidak ada tanggapan berarti tidak ada kesalahan yang terdeteksi).

Setelah menguji kedua set file, kita dapat memulai BIND. Seperti layanan Unix/Linux lainnya, skrip kontrol tersedia untuk memulai daemon BIND (ingat bahwa ini disebut bernama). Kita dapat mengeluarkan layanan instruksi bernama start atau menjalankan skrip langsung dari /etc/init.d sebagai /etc/init.d bernama start. Kata mulai adalah argumen yang menunjukkan apa yang kita ingin file skrip ini lakukan. Argumen lainnya adalah status untuk mendapatkan status run-time dari BIND: stop, restart (untuk menghentikan dan kemudian memulai BIND), reload, force-reload, dan try-restart; tiga argumen terakhir mencoba memulai kembali BIND tetapi hanya dalam keadaan tertentu. Dengan status, Anda menerima laporan tentang apa yang dilakukan BIND. Berikut ini adalah contohnya.

```
version: 9.8.2rc1-RedHat-9.8.2-0.30.rc1.el6_6.3
CPUs found: 1
worker threads: 1
number of zones: 17
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is ON
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
named (pid 27407) is running...
```

Perhatikan bahwa untuk menghentikan BIND, yang terbaik adalah menggunakan layanan ini, seperti pada layanan bernama stop. Namun, menggunakan Unix/Linux, perintah kill

dimungkinkan. Anda akan mengeluarkan instruksi berikut. Perhatikan bahwa jika Anda telah mengubah lokasi file pid, Anda harus menggunakan argumen yang berbeda dalam perintah kill.

```
kill -SIGTERM 'cat /var/run/named/named.pid'
```

Utilitas RNDC

Kami dapat mengelola BIND melalui file konfigurasi dan dengan memulai, menghentikan, atau memulai kembali layanan. Namun, BIND hadir dengan utilitas yang sangat nyaman yang disebut rndc (bernama utilitas kontrol server), yang merupakan alat baris perintah. Kami akan menggunakan rndc untuk mengelola server BIND baik dari host lokal atau host jarak jauh.

Untuk tujuan keamanan, rndc menggunakan kunci rahasia untuk berkomunikasi dengan server BIND melalui jaringan. Oleh karena itu, untuk menggunakan rdnc, pertama-tama kita perlu mengonfigurasi kunci rahasia yang akan digunakan oleh BIND dan rndc. Pertama, gunakan perintah rndc-confgen untuk membuat kunci rahasia. Satu-satunya argumen yang diperlukan untuk instruksi ini adalah –a untuk menunjukkan bahwa file kunci akan dibaca oleh rndc dan diberi nama saat startup, untuk digunakan sebagai kunci autentikasi default. Opsi lain memungkinkan Anda menentukan ukuran kunci (-b keysize), lokasi alternatif untuk file kunci (-c nama file), dan nama kunci alternatif (-k name). Lokasi default dan nama file untuk file kunci adalah /etc/rndc.key. Isi file kunci ini akan menggabungkan pernyataan kunci dan arahan. Misalnya, Anda mungkin melihat sesuatu seperti berikut ini. Dalam hal ini, kunci dibuat menggunakan algoritme hmac-md5.

```
key "rndc-key" {
        algorithm hmac-md5;
        secret "rPkvcxZknBxAOMZ5kNy+YA==";
};
```

Karena kami mengharapkan BIND untuk menggunakan kunci ini, kami harus menambahkan arahan penyertaan untuk file ini di file konfigurasi name.conf kami. Kami juga akan menentukan pernyataan kontrol. Di bawah ini, kami melihat entri yang akan kami tambahkan. Alamat dan port inet menunjukkan lokasi di mana pesan rndc akan diterima (yaitu, server BIND kami berada di 10.2.57.28 dan akan mendengarkan perintah rndc melalui port 953), dan kami akan mengizinkan akses ke semua host di subnet 10.2.0.0 /16.

```
include "/etc/rndc.key";
controls {
       inet 10.2.57.28 port 953
       allow { 10.2.0.0/16; } keys { "rndc-key"; };
};
```

Kita juga perlu mengonfigurasi utilitas rndc untuk menggunakan kunci rahasia yang sama. Ada file rndc.conf di /etc yang digunakan untuk mengkonfigurasi rndc (jika tidak ada file konfigurasi, kita harus membuatnya). Entri berikut digunakan untuk mengonfigurasi utilitas ini

untuk menggunakan kunci yang telah kami buat dan untuk berkomunikasi dengan server seperti yang tercantum di bawah pernyataan opsi.

```
include "/etc/rndc.key";
options {
        default-key "rndc-key";
        default-server 10.2.57.28;
        default-port 953;
};
```

Kita perlu menginstal dan mengkonfigurasi rndc untuk setiap host yang ingin kita kirimkan perintah rndc ke server BIND kita. Kami harus menyalin rndc.key ke masing-masing host ini. Gambar 8.6 mencantumkan perintah dan opsi rndc yang lebih umum digunakan. Ini harus relatif cukup jelas. Daftar lengkap perintah dan opsi dapat dilihat dengan menjalankan rndc tanpa argumen.

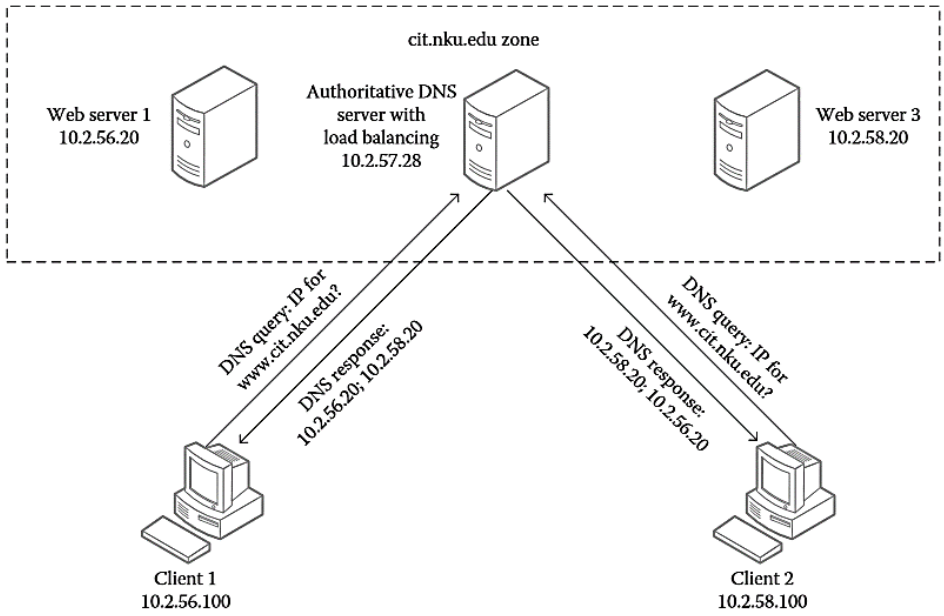
Contoh Konfigurasi BIND Sederhana

Pada subbagian ini, kita melihat contoh untuk konfigurasi BIND. Contoh kami didasarkan pada zona cit.nku.edu yang terdiri dari satu server DNS otoritatif dan dua server web (ws1 dan ws2) yang melayani www.cit.nku.edu. Alamat IP server DNS adalah 10.2.57.28. ws1 dan ws2 terletak di dua subnet, 10.2.56/24 dan 10.2.58/24. Alamat IP ws1 dan ws2 masing-masing adalah 10.2.56.20 dan 10.2.58.20. Gambar 8.2 menunjukkan zona cit.nku.edu.

Tabel 8.6 Perintah dan Opsi Umum Utilitas rndc

Perintah dan Opsi rndc	Arti
dumpdb [-all | -cache | -zone]	Menyebabkan data cache/zona server BIND dibuang ke file dump default (ditentukan oleh opsi file dump); Anda dapat mengontrol apakah akan membuang semua informasi, informasi cache, atau informasi zona, di mana defaultnya adalah semua
flush	Membersihkan cache server BIND
halt [-p] stop [-p]	Hentikan server BIND segera atau dengan anggun; mereka berbeda karena dengan penghentian, perubahan terbaru yang dibuat melalui pembaruan dinamis atau transfer zona tidak disimpan ke file master; opsi –p menyebabkan ID proses yang dinamai dikembalikan
notify zone	Kirim ulang pesan PEMBERITAHUAN ke zona yang ditunjukkan
querylog [on | off]	Mengaktifkan/menonaktifkan pencatatan kueri
reconfig	Memuat ulang file bernama.conf dan zona baru tetapi tidak memuat ulang file zona yang ada meskipun diubah
reload	Muat ulang file bernama.conf dan zona
stats	Menulis statistik server BIND ke file statistik
status	Menampilkan status server BIND saat ini
-c configuration_ file	Menentukan file konfigurasi alternatif
-p port_number	Secara default, rndc menggunakan port 953; ini memungkinkan Anda untuk menentukan port yang berbeda
-s server	Menentukan server selain server default yang terdaftar di /etc/rndc.conf

-y key_name	Menentukan kunci selain kunci default yang terdaftar /etc/rndc.conf.



Gambar 8.2 Server DNS otoritatif

Dalam contoh ini, kami telah menggunakan server BIND kami untuk melakukan penyeimbangan beban antara dua server web, dengan kebijakan penyeimbangan beban berikut.

- Jika klien yang menanyakan www.cit.nku.edu berasal dari subnet yang sama dengan server web, alamat IP server web tersebut akan ditempatkan pertama dalam daftar alamat yang dikembalikan oleh server DNS ke klien, untuk mendorong klien untuk menggunakan server web itu.
- Jika klien meminta www.cit.nku.edu berasal dari subnet selain 10.2.56/24 dan 10.2.58/24, urutan daftar alamat IP yang dikembalikan ke klien akan dalam urutan round-robin .

Kita perlu mengonfigurasi file konfigurasi named.conf dan file zona cit.nku.edu untuk server BIND untuk mengimplementasikan persyaratan di atas. Kita mulai dengan konfigurasi named.conf. Pertama, kita mendefinisikan pernyataan pilihan.

```
options {
  listen-on port 53 { 127.0.0.1; 10.2.57.28;};
  directory         "/var/named";
  allow-query       { localhost; 10.2.0.0/16;};
  recursion no;
  sortlist {
        {10.2.56.0/24;{10.2.56.0/24;10.2.58.0/24;};};
        {10.2.58.0/24;{10.2.58.0/24;10.2.56.0/24;};};
  };
  rrset-order {order cyclic;};
};
```

Kami menggunakan sortlist dan subpernyataan rrset-order untuk menentukan kebijakan load-balancing. Jika klien berasal dari subnet 10.2.56.0/24, 10.2.56.20 (ws1) akan menjadi yang

pertama dalam daftar alamat yang dikembalikan karena alamat ini cocok dengan subnet 10.2.56.0/24. 10.2.58.20 (ws2) akan menjadi yang kedua dalam daftar alamat yang dikembalikan karena cocok dengan 10.2.58.0/24. Demikian pula, jika klien berasal dari subnet 10.2.58.0/24, daftar alamat yang dikembalikan akan mencantumkan 10.2.58.20 terlebih dahulu, diikuti dengan 10.2.56.20. Jika klien berasal dari subnet selain 10.2.56.0/24 dan 10.2.58.0/24, urutan daftar alamat yang dikembalikan akan berada dalam urutan round-robin, yang ditentukan oleh subpernyataan rrset-order (rrset-order {order cyclic;}; cyclic berarti mode round-robin). Kami juga ingin menentukan perilaku logging BIND. Kami menentukan satu saluran dan tiga kategori.

```
logging {
  channel "cit_log" {
    file "/var/log/named/cit.log" versions 3;
    print-time yes;
    print-severity yes;
    print-category yes;
  };
  category "default" { "cit_log"; };
  category "general" { "cit_log"; };
  category "queries" { "cit_log"; };
};
```

Menurut konfigurasi ini, aktivitas BIND akan dikategorikan sebagai default, umum, atau kueri. Semua ini akan dicatat ke cit_log saluran tunggal. File cit_log akan disimpan sebagai /var/log/named/cit.log, dengan hingga tiga versi arsip yang tersedia sekaligus (yaitu, empat file log disimpan: versi saat ini sebagai cit.log dan tiga versi lama sebagai cit.log.1, cit. log.2, dan cit.log.3, dari terbaru ke terlama). Setiap entri yang dicatat akan mencatat tanggal/waktu kejadian, tingkat keparahan, dan kategori aktivitas. Kami juga menentukan zona maju dan mundur, sehingga bernama.conf dapat menunjuk ke file zona mereka.

```
#forward zone
zone "cit.nku.edu" {
        type master;
        file "cit.nku.edu";
};
#reverse zone
   zone "2.10.in-addr.arpa" IN {
        type master;
        file "2.10.rev";
};
```

Nama domain zona tersebut adalah cit.nku.edu. Server BIND dikonfigurasi sebagai server nama otoritatif utama untuk zona tersebut. Nama file zona maju adalah cit.nku.edu, dan nama file zona mundur adalah 2.10.rev. Mereka berada di bawah /var/bernama.

Sekarang, kita harus menentukan zona kita. Kami akan menghilangkan beberapa detail, karena kami telah menyajikan zona di Bab 5. Di sini, kami mendefinisikan elemen file zona maju dan mundur. Zona ini melayani pencarian DNS yang diteruskan. Kami mendefinisikan catatan sumber daya berikut dalam file.

```
$TTL 86400
@     IN      SOA     ns1.cit.nku.edu. root(
      1
      15
      15M
      4W
      1H)
      IN      NS      ns1.cit.nku.edu.
ns1   IN      A       10.2.57.28
www   IN      A       10.2.56.20
www   IN      A       10.2.58.20
```

Anda dapat melihat bahwa ada dua server www, dengan alamat IP 10.2.56.20 dan 10.2.58.20. Mesin ns1 adalah server nama otoritatif kami untuk zona tersebut, dan alamat IP-nya adalah 10.2.57.28.

Kami mendefinisikan catatan PTR (penunjuk) berikut dalam file zona terbalik, yang digunakan untuk menjawab pencarian DNS terbalik.

```
$TTL 86400
$ORIGIN 2.10.IN-ADDR.ARPA.
@   IN  SOA     ns1.cit.nku.edu. root.cit.nku.edu. (
    2011071001  ;Serial
    3600        ;Refresh
    1800        ;Retry
    604800      ;Expire
    86400       ;Minimum TTL
)
@       IN  NS          ns1.cit.nku.edu.
@       IN  PTR         cit.nku.edu.
28.57   IN  PTR         ns1.cit.nku.edu.
20.56   IN  PTR         www.cit.nku.edu.
20.58   IN  PTR         www.cit.nku.edu.
```

Dengan konfigurasi server BIND dan file zona kami siap, kami dapat memulai BIND. Kami akan mengujinya dengan kueri dari dua klien: klien 1 memiliki alamat IP 10.2.59.98 dan tidak berada di salah satu subnet server web kami, sedangkan klien 2 memiliki alamat IP 10.2.58.98 dan berada di subnet yang sama dengan yang kedua server web. Kami akan menggunakan perintah menggali untuk contoh kami.

Pertama, klien 1 mengeluarkan perintah penggalian untuk www.nku.edu. Outputnya ditunjukkan pada Gambar 8.3. Perhatikan bahwa di antara output adalah daftar flag, termasuk aa. Bendera aa menunjukkan jawaban otoritatif yang dikembalikan oleh server nama otoritatif untuk zona tersebut. Hasil dari perintah dig berisi dua alamat IP, tercantum di bagian Jawaban. Yang pertama dari dua alamat IP yang dikembalikan adalah 10.2.56.20, diikuti oleh 10.2.58.20. Jika klien 1 mengeluarkan kembali perintah dig yang sama, hasilnya akan membuat dua alamat IP dibalik. Artinya, 10.2.58.20 akan mendahului 10.2.56.20. Ini menunjukkan bahwa penyeimbangan beban terjadi di mana penjadwal round-robin digunakan, menghasilkan urutan alamat yang dikembalikan yang berbeda.

Kami dapat mengumpulkan lebih banyak informasi dari respons penggalian. Pertama, kita melihat bagian tambahan yang memberi tahu kita alamat server nama yang menanggapi permintaan kita. Kami juga melihat waktu hidup (TTL) untuk entri ini 86400 detik (1 hari),

seperti yang diatur oleh file zona cit.nku.edu. Nilai TTL mengontrol berapa lama entri dalam cache DNS tetap valid.

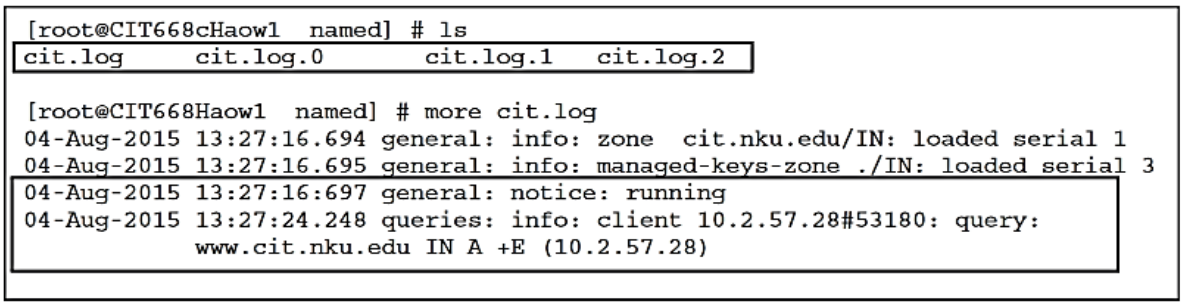
Meskipun tidak ditampilkan di sini, jika klien 2 mengeluarkan perintah dig yang sama, kami melihat respons yang sedikit berbeda. Perintah penggalian pertama akan menghasilkan urutan kedua server web menjadi 10.2.58.20, diikuti dengan 10.2.56.20. Namun, perintah penggalian kedua akan menghasilkan urutan respons yang persis sama. Ini karena klien 2 berbagi subnet yang sama dengan server web kedua dan BIND dikonfigurasi, sehingga perangkat apa pun di subnet ini harus menerima 10.2.58.20 terlebih dahulu.

```
[root@CIT668cHaow1 cit668]# dig @10.2.57.28   www.cit.nku.edu
;  <<>> DiG 9.10.2-P2  <<>> @10.2.57.28  www.cit.nku.edu
;  (1 server found)
 ; ; global options: +cmd
 ; ; Got answer:
; ;   ->>HEADER<<- opcode:   QUERY,   status:  NOERROR,  id:27034
; ;  flags: qr aa rd;  Query:1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 2
; ;  WARNING:  recursion requested but not available

; ;  OPT PSEUDOSECTION:
; ; EDNS:  version: 0,  flags:;   udp:  4096
; ;  Question SECTION:
; ;   www.cit.nku.edu.     IN        A
;
; ;  ANSWER SECTION:
www.cit.nku.edu.        86400   IN        A     10.2.56.20
www.cit.nku.edu.        86400   IN        A     10.2.58.20

; ; AUTHORITY SECTION:
cit.nku.edu.                  86400 IN    NS    ns1.cit.nku.edu.

; ; ADDITIONAL SECTION:
ns1.cit.nku.edu.              86400 IN    A     10.2.57.28

; ; Query time: 0 msec
; ; SERVER: 10.2.57.28#53(10.2.57.28)
; ; WHEN: Tue Aug 04 13:08:15 EDT 2015
; ; MSG SITE revd: 110
```

Gambar 8.3 Meneruskan pencarian DNS.

```
[root@CIT668cHaow1 cit668] # dig @localhost -x 10.2.56.20
;  <<>> DiG 9.10.2-P2  <<>> @localhost  -x  10.2.56.20
;  (1 server found)
 ; ; global options:   +cmd
 ; ; Got answer:
; ;   ->>HEADER<<-  opcode:   QUERY,   status:  NOERROR,  id:51173
; ;  flags: qr aa rd;  Query:1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
; ;  WARNING:  recursion requested but not available

; ;  OPT PSEUDOSECTION:
; ; EDNS:  version: 0,  flags:;   udp:  4096
; ;  Question SECTION:
;    20.56.2.10.in-addr.arpa.     IN            PTR

; ;  ANSWER SECTION:
20.56.2.10.in-addr.arpa.      86400     IN            PTR   www.cit.nku.edu.

; ; AUTHORITY SECTION:
2.10.in-addr.arpa.            86400     IN            NS    ns1.cit.nku.edu.

; ; ADDITIONAL SECTION:
ns1.cit.nku.edu.              86400     IN            A     10.2.57.28

; ;   Query time: 0 msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Wed Aug 05 11:24:47 EDT 2015
; ;  MSG SITE rcvd: 115
```

Gambar 8.4 Membalikkan pencarian DNS.



Gambar 8.5 Memeriksa file log BIND.

Sekarang, mari kita periksa responsnya menggunakan dig untuk pencarian DNS terbalik. Ini akan menguji file zona terbalik untuk konfigurasi yang benar. Gambar 8.4 menunjukkan output yang dihasilkan. Perhatikan bahwa catatan PTR digunakan untuk melacak alamat IP ke alias IP. Selain itu, seperti yang kita lihat pada Gambar 8.3, flag aa dikembalikan lagi, menunjukkan bahwa itu adalah server nama otoritatif yang merespons kueri kita.

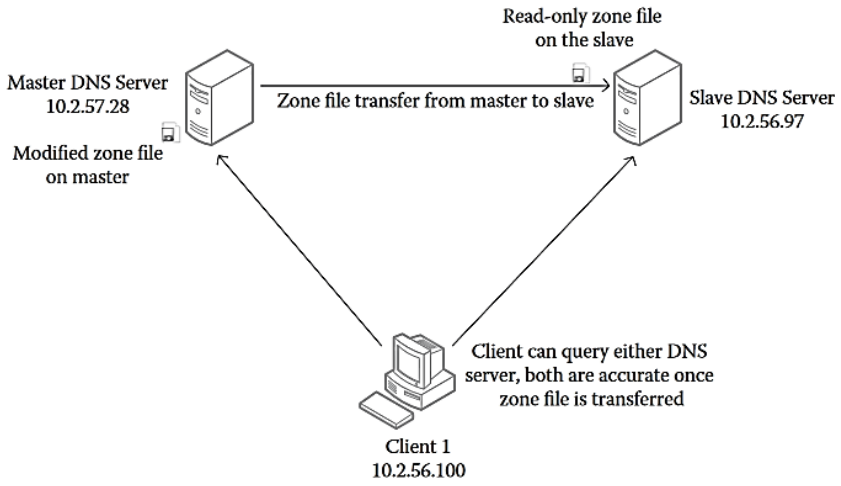
Kami menyelesaikan contoh ini dengan memeriksa file log berdasarkan kueri penggalian sebelumnya. Ingatlah bahwa BIND menyimpan empat file log, cit.log, cit.log.0, cit.log.1, dan cit.log.2. Gambar 8.5 menunjukkan file-file log ini ketika kita memeriksa direktori log dan konten terbaru dari file log yang aktif (cit.log). Disorot dari file log ini adalah kueri terbaru. Ada juga beberapa jalur umum yang merekam informasi status.

Contoh konfigurasi master dan slave BIND

Kami ingin membuat layanan DNS kami toleran terhadap kegagalan perangkat keras. Untuk ini, kami mungkin ingin menawarkan lebih dari satu server. Jika kami memiliki beberapa server nama, kami menetapkan satu sebagai server master dan yang lainnya sebagai server budak. Semua server ini adalah server otoritatif, tetapi kami harus memelihara file zona hanya dari server master. Dengan menyiapkan master dan slave, server master bertanggung jawab untuk mengirim pembaruan ke server slave saat kami mengubah file zona dari server master. Di sisi lain, jika server master turun, server budak akan tersedia untuk menjawab pertanyaan, memberi kita toleransi kesalahan yang diinginkan, dan bahkan jika server master tetap aktif, gabungan server dapat digunakan untuk membantu memuat kueri DNS. menyeimbangkan.

Kami akan mengonfigurasi master dan slave dengan menempatkan semua file zona kami hanya di server nama master. File zona ditransfer ke server nama budak berdasarkan jadwal yang kami sediakan untuk master dan budak melalui awal entri otoritas (SOA).

Kami akan menyempurnakan contoh kami dari subbagian terakhir dengan menambahkan server DNS budak ke zona cit.nku.edu. Karena alasan utama server slave adalah untuk meningkatkan toleransi kesalahan, kami ingin server ini berada di subnet yang berbeda. Server nama kami sebelumnya berada di 10.2.57.28. Kami akan menempatkan server nama ini di 10.2.56.97, yang berada di subnet 10.2.56.0/24. Gambar 8.6 menunjukkan pengaturan master/slave untuk zona cit.nku.edu.



Gambar 8.6 Penataan server DNS Master/slave.

Untuk mengkonfigurasi setup dari Gambar 8.6, kita harus memodifikasi file named.conf pada server master. Kami menambahkan transfer izinkan berikut dan beri tahu subpernyataan ke zona maju dan zona mundur. Subpernyataan memungkinkan-transfer menunjukkan bahwa file zona dapat ditransfer ke host yang terdaftar (10.2.56.97), sedangkan subpernyataan beri tahu menunjukkan bahwa master akan memberi tahu budak berdasarkan jadwal untuk melihat apakah budak memerlukan transfer zona.

```
# zone statement for forward zone file
zone "cit.nku.edu" {
     type master;
     file "cit.nku.edu";
     allow-transfer {localhost;10.2.56.97;};
     notify yes;
};

# zone statement for reverse zone file
zone "2.10.in-addr.arpa" IN {
     type master;
     file "2.10.rev";
     allow-transfer {localhost;10.2.56.97;};
     notify yes;
};
```

Kami juga harus memodifikasi file zona. Secara khusus, kami menambahkan rekor baru untuk server nama baru kami. Kami akan menyebutnya sebagai ns2. Untuk file zona maju, kami menambahkan catatan A. Untuk file zona terbalik, kami menambahkan catatan PTR. Kedua entri tersebut adalah sebagai berikut:

```
# added to the cit.nku.edu zone file
ns2     IN      A       10.2.56.97

# added to the the 2.10.rev zone file
97.56   IN      PTR     ns2.cit.nku.edu.
```

Sekarang, kita harus mengkonfigurasi server slave. Ingat bahwa budak tidak akan dikonfigurasi dengan file zona; ini akan dikirim dari master. Namun, kita harus mengkonfigurasi server slave dengan file name.conf miliknya sendiri. Kami dapat menyalin file ini langsung dari server master dan hanya membuat sedikit perubahan. Faktanya, satu-satunya perubahan yang diperlukan adalah pada pernyataan zona dengan mengubah tipe dari master menjadi budak, menunjukkan lokasi file zona pada budak ini dan menunjukkan siapa pemilik budak ini. Anda akan melihat di sini bahwa kami menyimpan file-file zona ini relatif terhadap direktori kerja BIND di subdirektori yang disebut budak. Ini tidak perlu tetapi dapat membantu pengaturan file kami. Perhatikan juga bahwa ada entri yang disebut master. Ini menyiratkan bahwa mungkin ada banyak master dari setiap budak, tetapi dalam contoh kami, ada satu server master.

```
zone "cit.nku.edu" IN {
     type slave;
     file "slaves/cit.nku.edu";
     masters{10.2.57.28;};
};

zone "2.10.in-addr.arpa" IN {
     type slave;
     file "slaves/2.10.rev";
     masters{10.2.57.28;};
};
```

Dengan konfigurasi master dan slave, kita harus memulai atau me-restart nama di kedua server. Jika BIND sudah berjalan di master, kita bisa saja me-reload file konfigurasi melalui opsi reload di server master.

Karena kami sekarang memiliki dua server nama DNS, kami juga harus memberi tahu klien kami tentang server nama baru. Untuk Unix/Linux, kami memodifikasi file /etc/resolv.conf, yang disimpan di setiap klien, mencantumkan server nama DNS untuk klien. Kami menempatkan master terlebih dahulu, karena kami lebih suka berkomunikasi dengannya jika tersedia.

```
#the master DNS server is the first preference
nameserver 10.2.57.28
#the slave DNSserver is the second preference
nameserver 10.2.56.97
```

Kami akan mencoba percobaan fail-over dengan penyiapan ini dengan menjadikan server master offline. Kita akan melihat server budak mengambil alih permintaan yang dikirim ke master. Pertama, kita jalankan perintah dig www.cit.nku.edu pada client kita (bisa client 1 atau client 2). Server master menjawab kueri, seperti yang diharapkan. Selanjutnya, kami menghentikan layanan bernama di server master. Kami menjalankan kembali perintah menggali pada klien kami. Kali ini, server budak menjawab pertanyaan tersebut.

Kami me-restart master. Sekarang, mari kita uji transfer zona. Pengaturan dua server BIND kami telah menentukan NOTIFY. Saat ada perubahan pada file zona pada master, master akan mengirimkan pesan NOTIFY ke server slave. Dalam situasi di mana ada beberapa budak, tidak semua budak akan menerima notifikasi pada waktu yang bersamaan. Bayangkan sebuah

skenario di mana kita memodifikasi master dan kemudian memodifikasinya lagi segera sesudahnya. Pemberitahuan pertama telah menyebabkan satu budak memperbarui tetapi tidak yang lain.

Akhirnya, budak kedua menerima pesan pemberitahuan kedua dan memperbarui rekamannya. Jika pemberitahuan pertama tiba kemudian, budak tidak boleh memperbarui dirinya lagi, karena itu akan memperbarui catatan zonanya ke versi yang lebih lama dan karenanya salah. Untuk mendukung ini, kami menggunakan nomor seri di antara data yang direkam dalam file zona. Saat administrator memperbarui file zona, dia akan memperbarui nomor seri ini. Angka ini merupakan kombinasi dari tanggal dan jumlah modifikasi yang dilakukan pada hari tersebut. Seorang budak, menerima pesan pemberitahuan, akan membandingkan nomor seri. Jika nomor serinya cocok, maka budak tersebut mengetahui bahwa ia memiliki versi terbaru. Sebagai contoh kami, kami menambahkan satu catatan A baru ke file zona maju di server master. Dengan asumsi bahwa kami melakukan modifikasi ini pada hari yang sama dengan modifikasi kami sebelumnya, kami menambah nomor seri file zona dengan 1 (kami akan mengubah nomor seri jika modifikasi ini dibuat pada hari yang berbeda). Kami menjalankan rndc reload di server master untuk memuat ulang file konfigurasi. Server master sekarang memberi tahu server budak, dan transfer zona terjadi di beberapa titik dalam waktu dekat.

Dalam kasus kami, kami dapat memverifikasi transfer zona dengan melihat file log server budak. Kami menemukan entri berikut, yang menunjukkan bahwa transfer terjadi dengan benar:

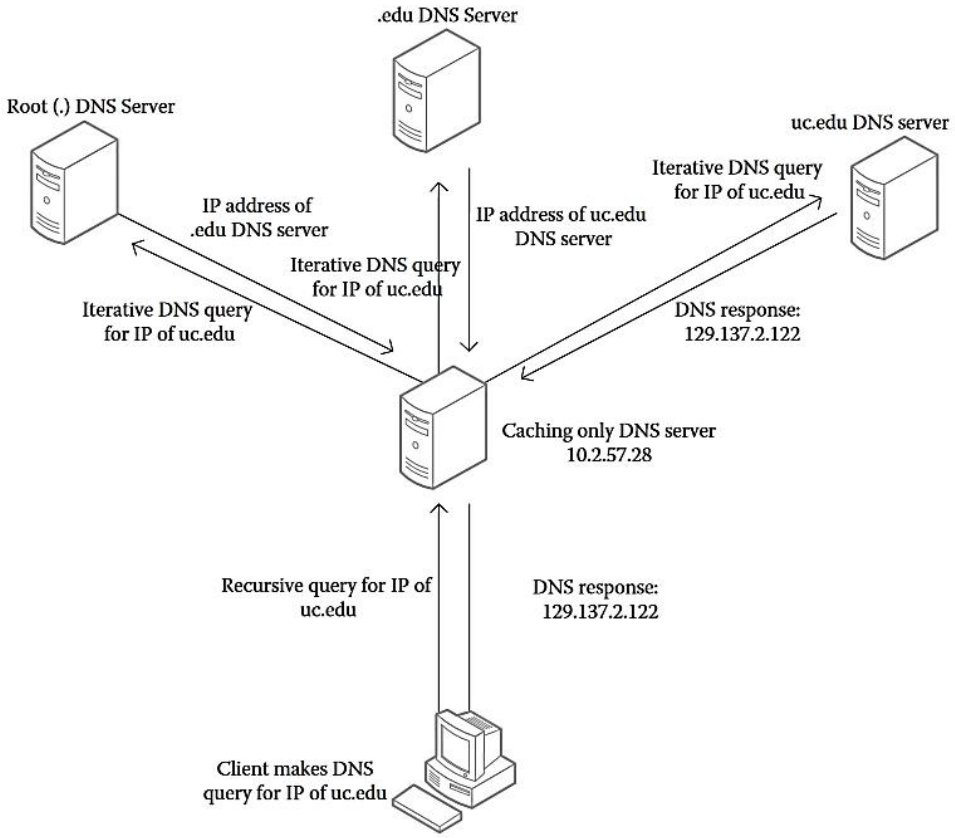
```
06-Aug-2015 02:01:54.392 general: info: zone
     cit.nku.edu/IN: Transfer started.
06-Aug-2015 02:01:54.393 xfer-in: info: transfer of
     'cit.nku.edu/IN' from 10.2.57.28#53: connected
     using 10.2.56.97#52490
06-Aug-2015 02:01:54.394 general: info: zone
     cit.nku.edu/IN: transferred serial 6
06-Aug-2015 02:01:54.394 xfer-in: info: transfer of
     'cit.nku.edu/IN' from 10.2.57.28#53: Transfer
     completed: 1 messages, 9 records, 228 bytes,
     0.001 secs (228000 bytes/sec)
```

Mengkonfigurasi Server DNS Untuk Caching

Pada subbagian ini, kita melihat dua bentuk lain dari server DNS. Kami memisahkan ini menjadi server DNS caching dan server DNS penerusan. Setiap server DNS dapat bertindak sebagai server khusus caching, server khusus penerusan, server penerusan, atau kombinasi dari server master/slave dan server caching dan/atau penerusan.

Server DNS khusus caching adalah jenis server yang dapat menjawab kueri rekursif dan mengambil informasi DNS yang diminta dari server nama lain atas nama klien kueri. Ini beroperasi pertama kali sebagai cache, mencari informasi alamat yang sesuai secara internal. Jika tidak ditemukan, itu kemudian dapat meneruskan permintaan ke server lain untuk mendapatkan informasi tersebut. Bagaimanapun, server DNS khusus caching bukanlah server otoritatif untuk zona mana pun. Cache digunakan untuk membantu meningkatkan kinerja

dengan menyimpan informasi secara lokal untuk membatasi jumlah lalu lintas Internet dan waktu tunggu. Contoh dari server DNS hanya caching ditunjukkan pada Gambar 8.7.



Gambar 8.7 Server DNS khusus caching.

Jika kita berasumsi bahwa alamat IP server kita adalah 10.2.57.28 dan akan mendengarkan antarmuka localhost dan port 53, pernyataan opsi berikut untuk file named.conf akan mengimplementasikan server ini hanya sebagai caching:

```
options {
        listen-on port 53 { 127.0.0.1;10.2.57.28; };
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        allow-query { localhost;10.2.0.0/16; };
        allow-query-cache {localhost;10.2.0.0/16; };
        allow-recursion { localhost;10.2.0.0/16; };
        recursion yes;
      };
```

Subpernyataan allow-query mendefinisikan bahwa localhost, dan setiap host dari subnet 10.2.0.0/16 diizinkan untuk mengakses server BIND ini. Subpernyataan allow-query-cache menetapkan bahwa host lokal dan semua host dari subnet 10.2.0.0/16 diizinkan untuk mengeluarkan kueri yang mengakses cache server BIND ini. Subpernyataan memungkinkan-rekursi menentukan bahwa localhost dan semua host dari subnet 10.2.0.0/16 diizinkan untuk mengeluarkan kueri rekursif ke server BIND ini. Sub-pernyataan rekursi menunjukkan bahwa server BIND akan melakukan kueri rekursif (ketika informasi belum di-cache).

Untuk mendukung kueri rekursif, server BIND perlu mengetahui alamat IP server DNS root, sehingga dapat meneruskan kueri yang diterima. Untuk ini, kita harus menambahkan pernyataan zona untuk zona root. Formatnya ditunjukkan di bawah ini, di mana nama zona yang diwakili hanyalah "." untuk akar Internet. File bernama.ca, yang akan ditempatkan di direktori /var/named berdasarkan konfigurasi instalasi kami sebelumnya, berisi nama dan alamat dari 13 server root Internet.

```
zone "." IN {
      type hint;
      file "named.ca";
};
```

Dengan server kami sekarang dikonfigurasi, kami memulai (atau memulai ulang) bernama. Sekali lagi, kami akan mendemonstrasikan server melalui perintah dig. Pertama kali kami mencoba perintah ini, nama yang diminta (uc.edu) tidak disimpan secara lokal, sehingga server khusus caching kami harus menggunakan kueri rekursif. Saat menerima respons, server nama kami akan meng-cache hasil ini dan mengembalikannya kepada kami. Mengeluarkan perintah dig yang sama sekarang mengirimkan hasilnya dari cache lokal. Interaksi ini ditunjukkan pada Gambar 8.8, di mana bagian atas gambar menunjukkan perintah penggalian pertama kita dan bagian bawah menunjukkan perintah penggalian kedua kita.

Hal pertama yang perlu diperhatikan pada Gambar 8.8 adalah bahwa di bawah flag, "aa" tidak ditampilkan. Ini menunjukkan bahwa jawaban tersebut tidak otoritatif karena server nama DNS kami bukan otoritas untuk domain uc.edu. Kedua instance dari perintah dig mengembalikan alamat IP yang sama yaitu 129.137.2.122, seperti yang kita harapkan. Namun, perbedaan yang mencolok ada pada waktu kueri. Permintaan pertama kami, yang dikirim secara rekursif, membutuhkan waktu 189 ms, sedangkan kueri kedua kami, diambil dari cache lokal, membutuhkan waktu 0 ms (perhatikan bahwa ini tidak memakan waktu 0 ms, tetapi waktunya kurang dari 1 ms).

Mari kita telusuri lebih jauh proses yang terjadi antara klien kita, server DNS kita, server root, dan, akhirnya, server otoritatif untuk uc.edu. Saat menerima permintaan pertama, server nama tidak memiliki data cache. Oleh karena itu, ia harus mengeluarkan kueri berulang ke server nama root, server nama .edu, dan server nama otoritatif untuk zona uc.edu untuk mengetahui alamat IP yang diminta. Kami benar-benar dapat melihat proses ini terjadi dengan menambahkan opsi +trace ke perintah dig kami. Gambar 8.9 memberikan output. Di bagian atas gambar ini, kita melihat kueri diteruskan ke server akar Internet. Kueri kemudian diteruskan ke server domain tingkat atas (TLD) untuk domain edu. Di sini, kita melihat server edu bernama a.edu-servers.net melalui f.edu-servers.net. Ini akan memiliki entri untuk semua subdomain di edu, termasuk uc.edu. Kueri kemudian dikirim ke salah satu server nama untuk zona uc.edu (dalam hal ini, ucdnsb.uc.edu). Tanggapan atas permintaan kami akhirnya dikembalikan ke server nama kami, tempat itu di-cache.

Saat mengirimkan perintah penggalian kedua kami, ini dilayani oleh server nama kami setelah menempatkan data di cache-nya sendiri. Jika kita ingin menjelajahi apa yang di-cache

di server nama caching kita, kita dapat menggunakan perintah rndc dumpdb. Perintah lengkapnya ditampilkan sebagai berikut:

```
rndc dumpdb -cache
```

File bernama cache_dump.db dibuat di bawah direktori /var/named/data/. Kita dapat melihat bahwa alamat IP 129.137.2.122 ada di dalam file tersebut, seperti yang ditunjukkan pada Gambar 8.10. Kueri yang di-cache ini akan kedaluwarsa dalam 883 detik. Waktu kedaluwarsa ditentukan oleh server nama otoritatif dengan menentukan TTL. Dalam hal ini, TTL adalah 900 detik, dan 7 detik telah berlalu sejak data dikembalikan.

```
; [root@CIT668cHaow1 named] # dig @localhost uc.edu
; <<>> DiG 9.10.2-P2 <<>> @localhost uc.edu
; (1 server found)
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:24440
; ; flags: qr rd ra; Query:1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3
; ; OPT PSEUDOSECTION:
; ; EDNS: version: 0, flags:; udp: 4096
; ; Question SECTION:
;  uc.edu.                        IN      A
; ; ANSWER SECTION:
uc.edu.                 900     IN      A       129.137.2.122
; ; AUTHORITY SECTION:
uc.edu.                 172800  IN      NS      ucdnsa.uc.edu.
uc.edu.                 172800  IN      NS      ucdnsb.uc.edu.
; ; ADDITIONAL SECTION:
ucdnsa.uc.edu.          172800  IN      A       129.137.254.4
ucdnsb.uc.edu.          172800  IN      A       129.137.255.4
; ; Query time: 189 msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Wed Aug 05 13:59:07 EDT 2015
; ; MSG SITE rcvd: 125


; [root@CIT668cHaow1 named] # dig @localhost uc.edu
; <<>> DiG 9.10.2-P2 <<>> @localhost uc.edu
; (1 server found)
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:64402
; ; flags: qr rd ra; Query:1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3
; ; OPT PSEUDOSECTION:
; ; EDNS: version: 0, flags:; udp: 4096
; ; Question SECTION:
;  uc.edu.                        IN      A
; ; ANSWER SECTION:
uc.edu.                 770     IN      A       129.137.2.122
; ; AUTHORITY SECTION:
uc.edu.                 172670  IN      NS      ucdnsa.uc.edu.
uc.edu.                 172670  IN      NS      ucdnsb.uc.edu.
; ; ADDITIONAL SECTION:
ucdnsa.uc.edu.          172670  IN      A       129.137.254.4
ucdnsb.uc.edu.          172670  IN      A       129.137.255.4
; ; Query time: 0 msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Wed Aug 05 13:59:07 EDT 2015
; ; MSG SITE rcvd: 125
```

Gambar 8.8 Melakukan caching hasil.

Server DNS penerusan adalah jenis server yang meneruskan semua permintaan ke server DNS rekursif untuk resolusi nama. Seperti server DNS caching, server DNS penerusan dapat menjawab permintaan rekursif dan meng-cache hasil kueri. Namun, server penerusan tidak melakukan rekursi sendiri. Gambar 8.11 mengilustrasikan pengaturan untuk server DNS penerusan.

```
[root@CIT668cHaow1  named]# dig  @localhost uc.edu  +trace
;  <<>> DiG 9. 10. 2-P2  <<>> @localhost  uc.edu  +trace
;    (1 server found)
; ;  global options:  +cmd
             517566             IN     NS     c.root-servers.net.
             517566             IN     NS     l.root-servers.net.
             517566             IN     NS     k.root-servers.net.
             517566             IN     NS     d.root-servers.net.
             517566             IN     NS     i.root-servers.net.
             517566             IN     NS     a.root-servers.net.
             517566             IN     NS     b.root-servers.net.
             517566             IN     NS     f.root-servers.net.
             517566             IN     NS     j.root-servers.net.
             517566             IN     NS     m.root-servers.net.
             517566             IN     NS     e.root-servers.net.
             517566             IN     NS     h.root-servers.net.
             517566             IN     NS     g.root-servers.net.
             517566             IN     NS     NS 8 0 S18400  20150815050000  2015
YX6T28ROcKwE10Y/5pbwfepC
GW2+Lz1KMy7nD14E49aLwQ9sgv2TewQLfHDc66Pnvh3zV6MUyXaPme/t

; ; Received 913  bytes from 127.0.0.1#53(localhost) in 0 ms

edu.         172800             IN     NS     d.edu-servers.net.
edu.         172800             IN     NS     a.edu-servers.net.
edu.         172800             IN     NS     g.edu-servers.net.
edu.         172800             IN     NS     l.edu-servers.net.
edu.         172800             IN     NS     c.edu-servers.net.
edu.         172800             IN     NS     f.edu-servers.net.
edu.         86400        IN    DS     28065 8 2 4172496CDE855345112904
edu.         86400        IN    RRSIG DS 8 1 86400  20150815050000  20150
nccMHkYPsh4nk4B2oMgjjc
w14/Y+XhWXkQuexeUdVu2P5cCyzyGWDEOVHqmlQtzKq5Yv1OUtPuUQ1z
; ; Received 477 bytes from 202.12.27.33#53(m.root-servers.net) in 180 ms

uc.edu.      172800             IN     NS     ucdnsa.uc.edu.
uc.edu.      172800             IN     NS     ucdnsb.uc.edu.
9DHS4EP5G8%PF9NUFK06HEKOO48QGk77.edu.  86400  IN  NSEC3   1   1   0  -
9DKP6R6NRDIVNLACTTSI
9DHS4EP5G8%PF9NUFK06HEKOO48QGk77.edu.  86400  IN  RRSIG    NSEC3  8 2 86400
201508121539
; ; Received 594 bytes from 192.5.6.30#53(a.edu-servers.net) in 79 ms

uc.edu.   900              IN    A     129.137.2.122
; ; Received 51 bytes from 129.137.255.4#53(ucdnsb.uc.edu) in 68 ms
```

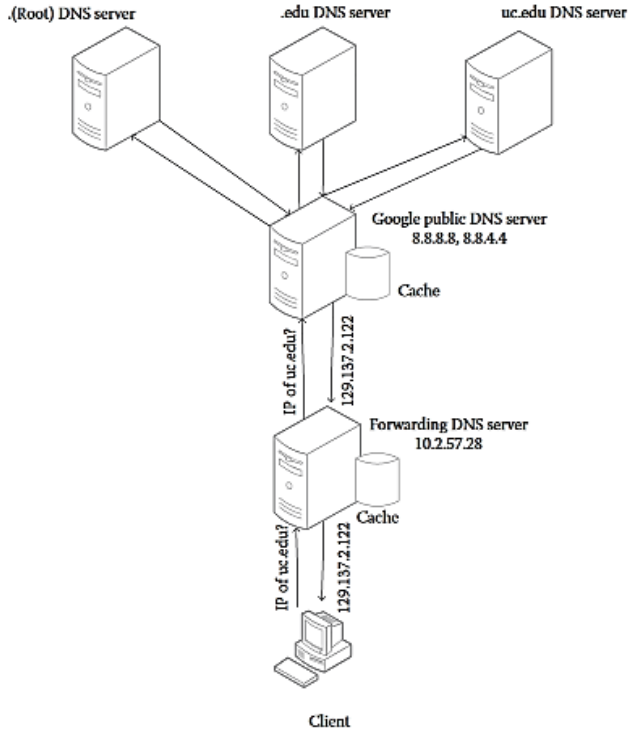
Gambar 8.9 Menelusuri kueri penggalian.

Untuk mengkonfigurasi server DNS menjadi server penerusan, kami memodifikasi lagi file bernama.conf. Dalam hal ini, kami menambahkan dua subpernyataan ke direktif opsi kami: forwarder untuk mencantumkan alamat tujuan yang akan kami teruskan kueri dan teruskan hanya untuk menunjukkan bahwa server nama DNS ini hanya melakukan penerusan.

```
options {
        …
        forwarders {
             8.8.8.8;
             8.8.4.4;
        };
        forward only;
};
```

```
;  glue
edu.                172783      NS    g.edu-servers.net.
                    172783      NS    d.edu-servers.net.
                    172783      NS    c.edu-servers.net.
                    172783      NS    l.edu-servers.net.
                    172783      NS    a.edu-servers.net.
                    172783      NS    f.edu-servers.net.
; additional
                    86383       DS    28065  8  2  (
                                      4172496CDE8554ES1129040355BD04B1FCF
                                      EBAE996DFDDE652006F6F8B2CE76
; additional
                    86383       RRSIG DS 8 1 86400  20150815050000 (
                                      20150805040000    1518  -
                                      lDXYQGagdYN32+nacd921adNdUewh/WlAgJ7
                                      YRVx1T98tubQ9fADbSx81QgrjKpjorKPtGOs
                                      6zpwjOuHk1WN3uEtVGnccMHkYPsh4nk4B2oH
                                      gjjcw14/Y+XhWXkOuexeUdVu2PScCYzyGWDE
                                      OVHqm1QtzKg5YviOUtPuUQIzypA=  )
; glue
uc.edu.             172783      NS    ucdnsa.uc.edu.
                    172783      NS    ucdnsb.uc.edu.
; authanswer
                    883         A     129.137.2.122
```

Gambar 8.10 Cache dump dari server BIND kami.



Gambar 8.11 Meneruskan server DNS.

```
[root@CIT668cHaow1 named] # dig @localhost uc.edu
; <<>> DiG 9.10.2-P2 <<>> @localhost uc.edu
; (1 server found)
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39246
; ; flags: qr rd ra; QUERY:1, ANSWER:1, AUTHORITY:0, ADDITIONAL:1

; ; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; ; QUESTION SECTION:
; uc.edu.                         IN              A

; ; ANSWER SECTION:
uc.edu.                 501     IN              A       129.137.2.122
; ; Query time: 38msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Wed Aug 05 16:48:37 EDT 2015
; ; MSG SIZE rcvd: 51
```

Gambar 8.12 Waktu kueri dicapai dengan meneruskan server DNS.

Pada contoh di atas, penerusan adalah server DNS publik Google, 8.8.8.8 dan 8.8.4.4. Jadi, kami meneruskan permintaan apa pun ke server publik ini. Pernyataan ke depan hanya membatasi server BIND ini untuk menjawab kueri rekursif dan sebagai gantinya meneruskan kueri rekursif apa pun ke depan.

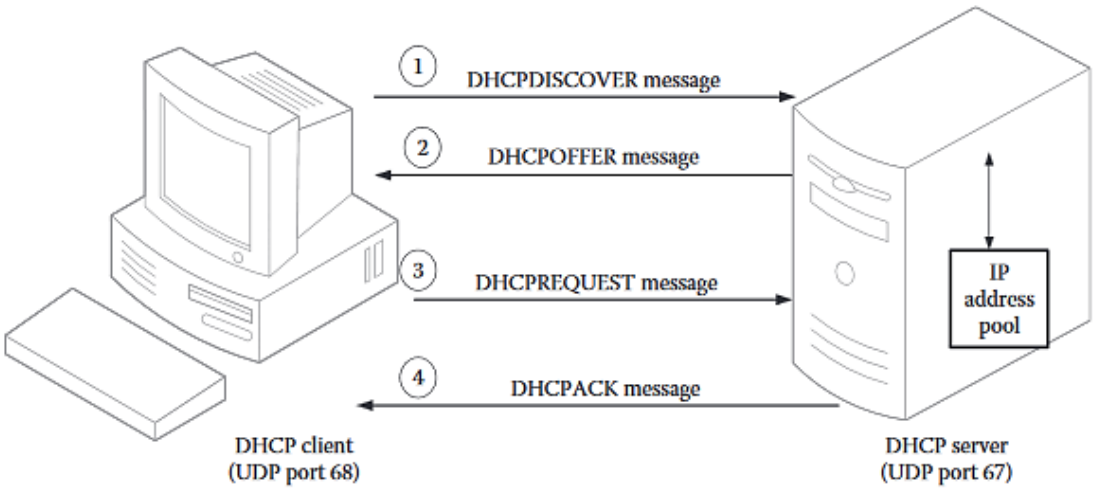
Kami harus memulai atau memulai ulang server kami dengan konfigurasi baru ini. Untuk menjelajahi server khusus penerusan lebih lanjut, kami menggunakan server BIND kami sebelumnya, tetapi kami membersihkan cache untuk mendemonstrasikan proses penerusan. Kami kembali mengeluarkan perintah penggalian. Outputnya ditunjukkan pada Gambar 8.12. Karena tidak ada data yang di-cache, permintaan diteruskan ke salah satu server nama Google. Waktu kueri adalah 38 ms. Ini jauh lebih baik daripada waktu server khusus caching kami (189 md) saat item tidak di-cache, karena Google memiliki infrastruktur yang jauh lebih baik (mis., lebih banyak server nama).

8.2 PENGALAMAT PROTOKOL INTERNET DINAMIS

Dalam contoh yang tercakup dalam Bagian 8.1, kami menambahkan dan/atau mengubah catatan sumber daya file zona secara manual. Meskipun pendekatan manual ini bekerja dengan baik untuk sejumlah kecil host dengan alamat IP statis, ini tidak dapat diskalakan. Jika ada ribuan host dengan alamat IP dinamis di jaringan lokal, pendekatan manual tidak layak untuk server DNS. Dynamic Host Configuration Protocol (DHCP) adalah protokol jaringan yang secara otomatis memberikan alamat IP ke host. Kita dapat mengintegrasikan server DHCP dengan server DNS untuk secara otomatis memperbarui informasi alamat IP dalam file zona server DNS ketika alamat IP baru ditetapkan oleh DHCP ke sebuah host. Pendekatan otomatis ini disebut Dynamic DNS (DDNS). Di bagian ini, pertama-tama kita menjelajahi DHCP untuk melihat apa itu dan bagaimana cara mengkonfigurasi server DHCP. Kami kemudian kembali ke BIND dan melihat bagaimana memodifikasi server BIND kami untuk menggunakan DHCP dan dengan demikian menjadi server DDNS.

Protokol Konfigurasi Host Dinamis

Ide di balik alamat IP dinamis sudah ada sejak tahun 1984, ketika Reverse Address Resolution Protocol (RARP) diumumkan. Ini memungkinkan stasiun kerja tanpa disk untuk secara dinamis mendapatkan alamat IP dari server yang digunakan stasiun kerja sebagai server file mereka. Ini diikuti oleh Protokol Bootstrap, BOOTP, yang ditetapkan pada tahun 1985 untuk menggantikan RARP. DHCP sebagian besar menggantikan BOOTP karena memiliki lebih banyak fitur yang membuatnya lebih kuat dan lebih mudah digunakan. DHCP sudah ada sejak tahun 1993, dengan versi yang diimplementasikan untuk IPv6 sejak tahun 2003.



Gambar 8.13 Proses penugasan alamat DHCP.

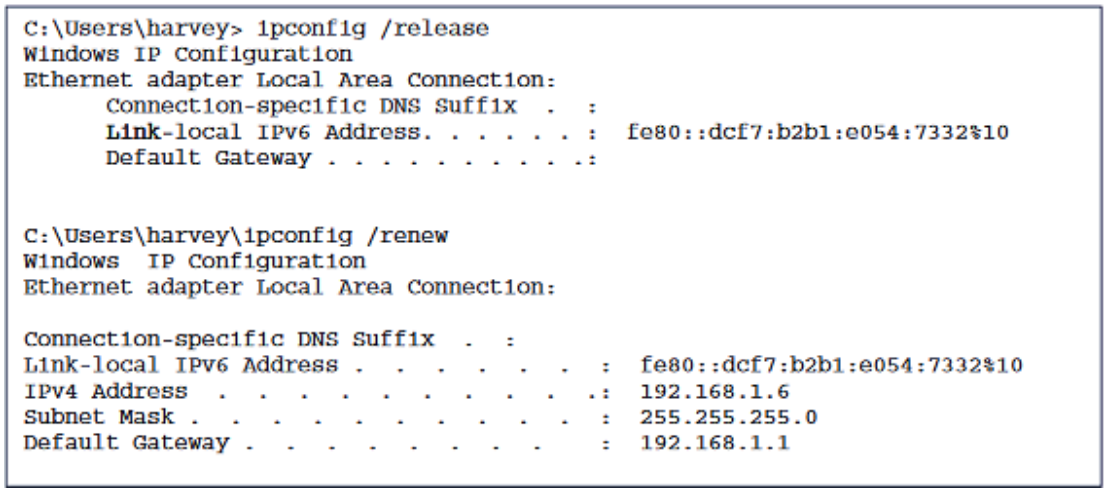
DHCP didasarkan pada model client-server, dimana klien DHCP memulai permintaan penugasan alamat dari server DHCP. Server DHCP memiliki sejumlah alamat IP yang diberikan yang dapat ditetapkan untuk permintaan klien. Selama setidaknya ada satu alamat IP yang tersedia di kumpulan alamatnya, ia mengalokasikan alamat berdasarkan permintaan. Klien DHCP dan server DHCP berkomunikasi satu sama lain melalui pesan User Datagram Protocol (UDP), dengan port default 67 untuk port tujuan server dan 68 untuk port tujuan klien. Gambar 8.13 menunjukkan proses penugasan alamat DHCP.

Proses DHCP, seperti yang ditunjukkan pada Gambar 8.13, dijelaskan sebagai berikut, sebagai urutan empat langkah:

1. Saat mem-boot, me-reboot, atau menghubungkan ke jaringan lokal, klien menyiarkan pesan DHCP DISCOVER untuk meminta alamat IP. Alamat kontrol akses media (MAC) dari kartu antarmuka jaringan disertakan dalam permintaan sehingga server DHCP dapat mengidentifikasi klien sebagai penerima dalam pesan responsnya.
2. Server DHCP, yang berada di jaringan lokal, mengalokasikan alamat IP yang tersedia dari kumpulan alamatnya ke klien. Ini dilakukan melalui respons PENAWARAN DHCP. Penawaran ini mencakup alamat IP yang ditawarkan, alamat MAC klien, dan informasi konfigurasi jaringan lainnya (misalnya, subnet mask, alamat IP server DNS jaringan, alamat IP gateway, dan alamat IP server DHCP). Semua informasi ini dikembalikan dalam pesan UDP ke klien. Sepotong informasi lain dalam penawaran adalah jumlah waktu alamat IP akan tersedia. Ini dikenal sebagai waktu sewa alamat IP.

3. Klien merespons dengan pesan DHCPREQUEST untuk menunjukkan bahwa ia telah menerima alamat IP yang diberikan.
4. Server DHCP mengirimkan pesan DHCPACK, mengakui bahwa klien dapat mulai menggunakan alamat IP.

Anda mungkin bertanya-tanya mengapa langkah 3 dan 4 diperlukan. Saat menyiarkan pesan awalnya di jaringan lokal, klien tidak tahu dari mana ia akan menerima alamat IP. Artinya, sebuah jaringan berpotensi memiliki beberapa server DHCP. Bayangkan dalam situasi seperti itu klien menerima alamat dari dua server DHCP. Yang mana yang harus digunakan? Lebih penting lagi, kedua server DHCP akan percaya bahwa alamat IP yang dikeluarkan sedang digunakan. Tetapi klien hanya akan menggunakan salah satu alamat. Tanpa langkah 3, server tidak akan mengetahui secara pasti bahwa alamat IP yang ditawarkan telah diterima. Dengan menambahkan langkah 3 dan 4, server mengetahui bahwa alamat IP yang ditawarkan sekarang tidak tersedia untuk dibagikan ke permintaan klien lain.



Gambar 8.14 Melepaskan dan memperbarui sewa di windows.

Server DHCP dapat menetapkan alamat IP ke klien dengan tiga cara berikut:

1. Alokasi otomatis: Server memberikan alamat IP permanen ke klien.
2. Alokasi dinamis: Server memberikan alamat IP ke klien untuk jangka waktu tertentu yang disebut sewa. Jika sewa tidak diperpanjang, alamat IP akan diambil kembali oleh server DHCP saat batas waktu habis. Saat mengklaim kembali alamat, server DHCP dapat mengeluarkan alamat itu ke klien lain di masa mendatang.
3. Alokasi manual: Administrator sistem menetapkan alamat IP ke klien. Server DHCP digunakan untuk menyampaikan alamat yang ditetapkan ke klien.

Selain permintaan, klien DHCP juga dapat melepaskan alamat IP yang disewa dan memperbarui alamat IP yang disewa. Dalam kasus sebelumnya, ini mungkin terjadi saat klien dihapus dari jaringan selama beberapa waktu. Alasan untuk memperbarui sewa adalah karena sewa telah mencapai tanggal/waktu kedaluwarsa. Ini sering dilakukan secara otomatis oleh sistem operasi Anda. Anda dapat secara manual melepaskan sewa pada Windows melalui perintah ipconfig /release, yang ditunjukkan di bagian atas Gambar 8.14. Anda dapat

memperbarui kontrak secara manual di Windows melalui perintah ipconfig /renew, yang ditunjukkan di bagian bawah Gambar 8.14.

Di Windows, menggunakan ipconfig, Anda juga dapat memeriksa informasi kedaluwarsa sewa. Ini dapat dilakukan dengan menggunakan opsi /all. Gambar 8.15 mengilustrasikan hasil dari ipconfig /all, menunjukkan tanggal dan waktu sewa diterima dan berakhir. Perhatikan bahwa kedaluwarsa diatur untuk 1 hari dan 1 detik lebih lambat dari waktu yang diperoleh. Biasanya, alamat IP akan disewakan untuk jangka waktu yang lebih lama.

SERVER DHCP ISC

Kita sekarang melihat bagaimana mengatur server DHCP dengan menggunakan server DHCP ISC, yang merupakan perangkat lunak server DHCP sumber terbuka yang paling sering digunakan di Internet.

```
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . :   Realtek PCIe GBE Family Controller
   Physical Address . . . . . . . . . :   C8 9C-DC-F5-95-78
   DHCP Enabled . . . . . . . . . . . :   Yes
   Autoconfiguration Enabled . . . . :   Yes
   Link-local IPv6 Address . . . . . :   fe80::dcf7:b2b1:e054:7332%10(Preferred)
   IPv4 Address . . . . . . . . . . . :   192.168.1.6(Preferred)
   Subnet Mask . . . . . . . . . . . :   255.255.255.0
   Lease Obtained . . . . . . . . . . :   Sunday, August 16, 2015 11:58:18 AM
   Lease Expires . . . . . . . . . . :   Monday, August 17, 2015 11:58:19 AM
   Default Gateway . . . . . . . . . :   192.168.1.1
   DHCP Server . . . . . . . . . . . :   192.168.1.1
   DHCPv6 IAID . . . . . . . . . . . :   248028380
   DHCPv6 Client DUID. . . . . . . . :   00-01-00-01-18-35-3A-FE-C8-9C-DC-F5-95-78
   DNS Servers . . . . . . . . . . . :   192.168.1.1
   NetBIOS over Tcpip . . . . . . . :   Enabled
```

Gambar 8.15 ipconfig /all dieksekusi di Windows menampilkan informasi sewa.

Seperti yang kami lakukan dengan BIND, kami akan menginstal DHCP melalui kode sumber. Pertama, dapatkan paket dhcp-4.3.5.tar.gz dari ftp://ftp.isc.org/isc/dhcp/4.3.5/dhcp-4.3.5.tar.gz. Kemudian, untar dan unzip paket dengan menggunakan tar –xzf. Ini membuat subdirektori bernama dhcp-4.3.5. Sekarang, kami mengeluarkan, seperti yang kami lakukan dengan DHCP, perintah konfigurasikan, buat, dan buat instal. Kita akan mengikuti pola lokasi konfigurasi yang sama seperti yang kita lakukan dengan BIND, menempatkan semua konten di /usr/local/dhcp, kecuali file konfigurasi, yang akan masuk ke /etc/dhcp.

```
./configure --prefix=/usr/local --sysconfdir=/etc/dhcp
make
make install
```

Hasil membangun DHCP adalah pembuatan tiga file yang dapat dieksekusi: dhclient, dhcrelay, dan dhcpd, semuanya terletak di /usr/local/dhcp/sbin. Yang pertama, dhclient, adalah perangkat lunak klien yang memungkinkan host ini meminta alamat IP dari server DHCP. Program dhcrelay digunakan untuk menerima permintaan DHCP pada subnet tanpa server DHCP dan meneruskannya ke server DHCP di subnet lain. Server itu sendiri disebut dhcpd, dan ini adalah program yang akan kami konfigurasikan untuk menetapkan alamat IP ke klien.

File konfigurasi untuk server DHCP disebut dhcpd.conf, dan kami menempatkannya di bawah /etc/dhcp. Seperti halnya named.conf, file konfigurasi ini terdiri dari komentar dan arahan. Komentar dimulai dengan #; direktif dapat diklasifikasikan dalam dua kelompok: pernyataan parameter dan pernyataan deklarasi.

Pernyataan parameter digunakan untuk mengontrol perilaku server DHCP atau menentukan nilai opsi DHCP yang dikirim ke klien DHCP. Pernyataan parameter dapat dikategorikan dalam dua kelompok: pernyataan opsi dan kontrol. Semua pernyataan opsi dimulai dengan opsi kata kunci. Pernyataan opsi menentukan nilai opsi DHCP yang dikirim ke klien DHCP. Pernyataan opsi akan muncul sebagai opsi option_name option_data;. Kami mengeksplorasi pilihan umum pada Gambar 8.7. Untuk daftar opsi lengkap, lihat halaman manual dhcp-options.

Gambar 8.8 mencantumkan beberapa pernyataan non-opsi atau kontrol yang umum digunakan untuk DHCP. Sintaksnya mirip dengan sintaks opsi, kecuali kata opsi dihilangkan, seperti pada default-lease-time 1000;, max-lease-time 86400;, atau lease-file-name /usr/local/dhcp/leases.txt;, di mana setiap pernyataan diakhiri dengan titik koma.

Direktif tambahan digunakan untuk menggambarkan topologi jaringan, klien di jaringan, alamat yang tersedia yang dapat ditetapkan ke klien, dan parameter yang harus diterapkan ke grup klien atau alamat yang ditentukan. Pernyataan deklarasi yang umum digunakan tercantum dalam Gambar 8.9. Contoh pernyataan mengikuti di bawah tabel.

Gambar 8.7 Jenis Opsi Umum untuk DHCP

Nama Opsi dan Data	Arti
broadcast-address ip-address	Menentukan alamat broadcast untuk subnet klien
domain-name text	Menentukan nama domain untuk resolusi nama klien
domain-name-servers ip-address [, ip-address …]	Menentukan server DNS untuk resolusi nama klien, di mana server akan mencoba server dalam urutan yang ditentukan
domain-search domain-list	Menentukan daftar pencarian nama domain yang digunakan klien untuk menemukan tidak sepenuhnya
host-name string	nama domain yang memenuhi syarat
routers ip-address [, ip-address …]	Menentukan nama host klien
subnet-mask ip-address	Menentukan router (berdasarkan alamat IP) pada subnet klien, dicantumkan dalam urutan preferensi

Gambar 8.8 Pernyataan Non-Opsi Umum

Pernyataan dan Parameter	Arti
authoritative	Menunjukkan bahwa server ini adalah server resmi untuk jaringan lokal
ddns-update-style style	Gaya adalah salah satu sementara dan tidak ada (default), di mana sementara berarti bahwa server akan secara dinamis memperbarui server nama DNS dan tidak ada berarti bahwa server tidak akan memperbarui server nama DNS

default-lease-time time	Menentukan waktu sewa dalam hitungan detik (kecuali jika klien menentukan waktu sewanya sendiri dalam permintaan)
fixed-address address [,address...]	Menetapkan satu atau lebih alamat IP spesifik ke klien
hardware hardware-type hardware-address	Menentukan jenis antarmuka dan alamat yang terkait dengan klien, di mana jenis perangkat keras akan menjadi salah satu dari ethernet atau token-ring, dan alamatnya akan menjadi alamat MAC dari antarmuka
lease-file-name name	Menentukan lokasi file yang berisi informasi sewa untuk server
local-port port	Menentukan port yang didengarkan server dengan default 67
local-address address	Menentukan alamat IP yang didengarkan server untuk permintaan masuk; secara default, server mendengarkan semua alamat IP yang ditetapkan
log-facility facility	Menentukan tingkat log untuk peristiwa dengan menggunakan salah satu dari local0 (darurat), local1 (peringatan), ..., local7 (debug), dengan default menjadi log0
max-lease-time time min-lease-time time	Menentukan waktu maksimum dan minimum, dalam detik, untuk sewa
pid-file-name filename	Menentukan file ID proses, dengan nama file default /var/run/dhcpd.pid
remote-port port	Mengarahkan server untuk mengirim responsnya ke nomor port yang ditentukan untuk klien; standarnya adalah 68

Gambar 8.9 Arahan Umum untuk Menentukan Kondisi Jaringan

Pengarahan	**Arti**

group {
[parameters] [
declarations]
}
	Menerapkan parameter ke grup host yang dideklarasikan, jaringan, subnet, atau grup lain yang ditentukan

host hostname {
[parameters]
[declarations]
}
	Menentukan informasi konfigurasi untuk klien, termasuk, misalnya, alamat IP tetap yang akan selalu ditetapkan ke host tersebut

range [dynamic-bootp] low-address [high-address]	Menentukan alamat IP terendah dan tertinggi dalam suatu rentang, membuat kumpulan alamat IP yang tersedia; perhatikan bahwa kami dapat menetapkan banyak rentang; jika flag dynamic-bootp disetel, maka klien BOOTP juga dapat diberi alamat IP dari server DHCP ini; jika menentukan satu alamat, alamat tinggi dihilangkan

shared-network name {
[parameters]
[declarations]
}
	Mendeklarasikan bahwa subnet dalam pernyataan berbagi jaringan fisik yang sama dan secara opsional dapat menetapkan parameter ke subnet yang ditentukan

subnet subnet-number netmask netmask { [parameters]	Mendeklarasikan subnet dan kisaran alamat IP yang tersedia yang dapat dikeluarkan, bersama dengan parameter opsional seperti router, subnet mask, dan server DNS

[declarations] }	

Kami mungkin mendefinisikan sekelompok host sebagai berikut:

```
group {
      …
      host s1 {
         option host-name "s1.cit.nku.edu";
           hardware ethernet C8:9C:DC:F5:95:78;
           fixed-address 10.10.10.10;
      }
      host s2 {
         …
      }
   }
```

Berikut ini mendeklarasikan jaringan bersama, di mana cit menggabungkan subnet 10.10.10.0 dan 10.10.20.0:

```
shared-network cit {
  ...
 subnet 10.10.10.0 netmask 255.255.0.0 {
   ...
    }
 subnet 10.10.20.0 netmask 255.255.0.0 {
   ...
 }
}
```

Pernyataan subnet berikut menentukan parameter untuk subnet 10.10.10.0. Di sini, kita melihat subnet mask, alamat router untuk subnet, nama domain, dan server nama DNS, diikuti dengan kisaran alamat IP yang dapat dikeluarkan untuk klien subnet ini.

```
subnet 10.10.10.0 netmask 255.255.255.0 {
  option routers 10.10.10.1;
   option domain-name "cit.nku.edu";
   option domain-name-servers 10.10.10.2;
   range 10.10.10.20 10.10.10.80;
}
```

Perhatikan bahwa beberapa detail dihilangkan dan diganti dengan … pada contoh di atas. Di sini, kita melihat contoh yang lebih kompleks untuk file dhcp.conf. Sekali lagi, untuk melihat daftar lengkap dari arahan yang tersedia, lihat man dhcpd.conf.

```
option domain-name "cit.nku.edu";
option domain-name-servers ns1.cit.nku.edu;
default-lease-time 600;
lease-file-name "/var/lib/dhcpd/dhcpd.leases";
max-lease-time 7200;
log-facility local7;
local-address 10.2.57.28;
subnet 10.2.57.0 netmask 255.255.255.0 {
   range 10.2.57.100 10.2.57.110;
   option subnet-mask 255.255.255.0;
}
```

File / database sewa harus ada sebelum server DHCP dimulai. File sewa tidak dibuat selama instalasi server DHCP. Administrator harus membuat file sewa kosong secara manual. Ini dapat dilakukan dengan menggunakan perintah sentuh Unix/Linux. Anda juga ingin menempatkan file sewa di subdirektori. Berikut ini dapat digunakan untuk mengatur file sewa Anda:

```
mkdir /var/lib/dhcpd/
touch /var/lib/dhcpd/dhcpd.leases
```

Sekarang, kita mulai server DHCP sebagai berikut:

```
/usr/local/sbin/dhcpd -cf /etc/dhcp/dhcpd.conf
```

Perintah dhcpd mengizinkan banyak opsi. Dalam perintah kami di atas, kami menggunakan –cf untuk menentukan lokasi file konfigurasi kami. Gambar 8.10 mengilustrasikan pilihan umum yang tersedia. Pilihan lain dapat dilihat melalui man dhcpd.

Kami sekarang memeriksa cara menguji server DHCP kami. Kami menggunakan sistem operasi CentOS 6 Linux untuk klien DHCP kami. Pertama, kita harus mengkonfigurasi klien untuk menggunakan DHCP. Ini dilakukan dengan mengedit file konfigurasi antarmuka /etc/sysconfig/network-scripts/ifcfg-eth0. Kami menentukan entri berikut untuk file ini. Intinya, kami menyatakan bahwa klien ini harus mencari alamat IP-nya dari server DHCP pada saat melakukan booting.

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

Kami menyimpan file konfigurasi ini dan memulai ulang komputer (atau memulai ulang layanan jaringan melalui restart jaringan layanan). Dengan klien sekarang memperoleh alamat IP dari server DHCP, jika kita menjalankan perintah ip addr (atau ifconfig), kita dapat melihat alamat yang telah diberikan padanya.

Gambar 8.10 Opsi Baris Perintah umum dhcpd

Pilihan	Arti
–cf config-file	Lokasi file konfigurasi
–d	Mengirim pesan log ke deskriptor kesalahan standar (tujuan default adalah monitor); opsi ini menyiratkan opsi "–f".

–f	Menjalankan dhcpd sebagai proses latar depan
–lf lease-file	Lokasi file sewa
–p port	Arahkan dhcpd untuk mendengarkan nomor port UDP yang ditentukan
–q	Menekan pesan hak cipta selama startup
–t	Menguji sintaks file konfigurasi, tanpa memulai DHCP
–T	Menguji sintaks file sewa, tanpa memulai DHCP

Di server DHCP, kami memeriksa file sewa untuk melihat apakah server DHCP mengadakan sewa untuk mesin klien. Setiap entri sewa dalam file sewa memiliki format berikut:

```
lease ip-address { statements... }
```

Di sini, ip-address adalah alamat IP yang telah disewakan kepada klien. Beberapa pernyataan yang umum digunakan terdaftar sebagai berikut:

- mulai tanggal; pernyataan ini menunjukkan waktu dimulainya sewa. Tanggal ditentukan dalam format ini: “weekday year/month/day hour:minute:second”. Hari kerja dinyatakan sebagai angka dari 0 sampai 6, dengan 0 adalah hari Minggu. Tahun dinyatakan dengan empat digit. Bulan adalah angka antara 1 dan 12. Hari adalah angka antara 1 dan 31. Jam adalah angka antara 0 dan 23. Menit/detik adalah angka antara 0 dan 59. Waktu sewa ditentukan dalam Universal Coordinated Waktu (UTC).
- tanggal berakhir; pernyataan ini menunjukkan waktu berakhirnya sewa. Untuk sewa tak terbatas, tanggal ditetapkan menjadi tidak pernah.
- alamat mac tipe perangkat keras perangkat keras; pernyataan ini menunjukkan alamat MAC klien.
- pengidentifikasi klien uid; pernyataan ini mencatat pengidentifikasi klien yang digunakan oleh klien untuk memperoleh sewa.
- nama host klien-nama host; pernyataan ini menunjukkan nama host klien.

Untuk daftar lengkap pernyataan yang didukung oleh dhcpd, gunakan halaman manual untuk dhcpd.leases.

Berikut ini adalah satu entri dari file /var/lib/dhcpd/dhcpd.leases untuk mendemonstrasikan sewa yang diberikan kepada klien 192.168.42.1.

```
lease 192.168.42.1 {
      starts 0 2015/08/20 08:02:50;
      ends 5 2015/08/21 08:02:50;
      hardware ethernet 00:50:04:53:D5:57;
      uid 01:00:50:04:53:D5:57;
      client-hostname "PC0097";
}
```

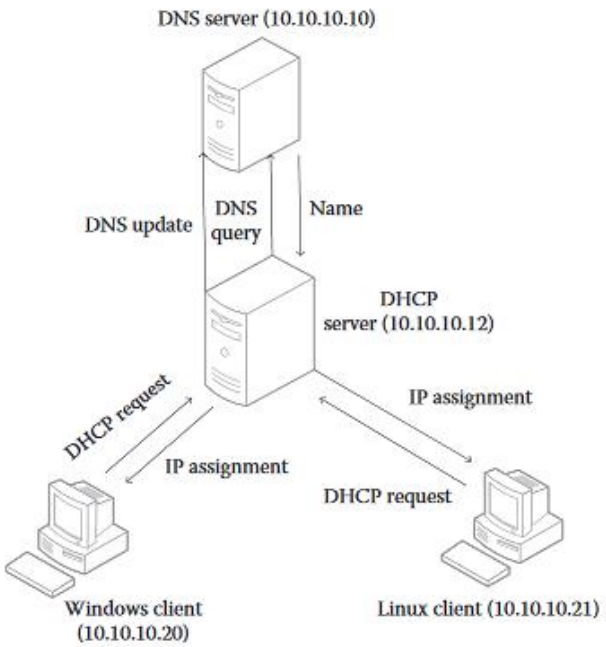
MENGINTEGRASIKAN SERVER DHCP ISC DENGAN SERVER DNS BIND

Server DHCP dapat secara dinamis memperbarui file zona server DNS (file zona maju dan file zona mundur) saat server DNS sedang berjalan. Dalam subbagian ini, kita melihat bagaimana mengkonfigurasi server ISC DHCP dan BIND DNS yang telah dikonfigurasi sebelumnya untuk mengizinkan kerja sama ini. Hasil akhirnya adalah server DNS kami menjadi

server DDNS. Gambar 8.16 menunjukkan setup. Alamat IP server DNS adalah 10.10.10.10. Alamat IP server DHCP adalah 10.10.10.12. Kumpulan alamat IP-nya berkisar dari 10.10.10.20 hingga 10.10.10.50.

Pertama, kita mengkonfigurasi server DHCP dengan mengedit file konfigurasi dhcpd.conf. Di bawah ini, kami menampilkan versi baru dari file ini. Perhatikan direktif baru, ddns-update-style interim, seperti yang disebutkan pada Tabel 6.8 dari subbagian sebelumnya. Pernyataan ini memungkinkan server DHCP kami untuk memperbarui server DNS kami. Server DHCP sudah mengetahui alamat IP server DNS melalui opsi server nama domain di klausa subnet.

```
ddns-update-style interim;
option domain-name "cit.nku.edu";
default-lease-time 600;
lease-file-name "/var/lib/dhcpd/dhcpd.leases";
max-lease-time 7200;
log-facility local3;
local-address 10.10.10.12;
authoritative;
subnet 10.10.10.0 netmask 255.255.255.0 {
      range dynamic-bootp 10.10.10.20 10.10.10.50;
      option subnet-mask 255.255.255.0;
      option broadcast-address 10.10.10.255;
      option routers 10.10.10.1;
      option domain-name-servers 10.10.10.10;
}
```



Gambar 8.16 Pengaturan DDNS.

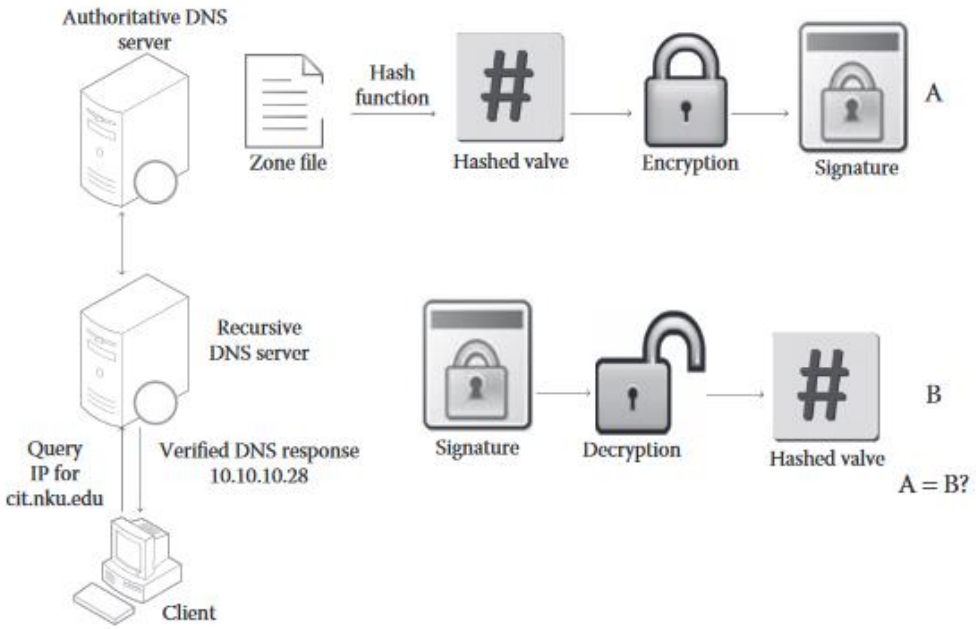
Kami juga harus mengonfigurasi server DNS kami dengan menambahkan arahan perbarui untuk menunjukkan siapa yang dapat memperbarui file server ini. Mesin pemutakhiran akan menjadi server DHCP kami, 10.10.10.12. Kami mengedit file name.conf server DNS dengan memperbarui semua entri zona.

```
zone "cit.nku.edu" {
        type master;
        file "cit.nku.edu";
        allow-update { 10.10.10.12; };
};

zone "2.10.in-addr.arpa" IN {
        type master;
        file "2.10.rev";
        allow-update { 10.10.10.12; };
};
```

Kami juga harus mengonfigurasi salah satu klien kami untuk menggunakan DHCP. Kami menjelajahi cara melakukan ini di Unix/Linux dengan memodifikasi file data antarmuka.

8.3 MENGKONFIGURASI DNSSEC UNTUK SERVER BIND

DNSSEC adalah ekstensi ke DNS yang dapat membantu klien DNS mendeteksi jika respons DNS telah diubah. Mari kita lihat cara kerja DNSSEC. Pada Gambar 8.17, kita melihat server DNS otoritatif, di mana setiap catatan DNS di file zona disimpan berdasarkan fungsi hash. Nilai yang dikembalikan oleh fungsi hash disebut nilai hash. Nilai hash dienkripsi dengan kunci pribadi dari server DNS otoritatif. Nilai hash terenkripsi ini disebut tanda tangan digital. Fungsi hash yang digunakan untuk menghasilkan tanda tangan termasuk dalam tanda tangan. Saat server otoritatif menjawab kueri, tanda tangan digital disertakan dalam responsnya. Server DNS rekursif, yang menanyakan server otoritatif atas nama klien, menggunakan tanda tangan digital untuk memeriksa apakah respons kueri telah diubah. Pertama, ia menggunakan kunci publik dari server otoritatif untuk mendekripsi tanda tangan dan mendapatkan nilai hash yang dihasilkan oleh server otoritatif. Nilai hash ini dilambangkan sebagai A pada gambar. Server DNS rekursif kemudian menggunakan fungsi hash yang disertakan dalam tanda tangan untuk mencirikan pesan teks biasa dalam respons, untuk menghasilkan nilai hash. Nilai hash ini diberi label sebagai B pada gambar. Jika A dan B sama, maka server DNS rekursif mengetahui bahwa responsnya belum diubah. Jika tidak, harus diasumsikan bahwa respons diubah di beberapa titik.



Gambar 8.17 Memverifikasi kueri DNS dengan DNSSEC.

Untuk mengonfigurasi BIND agar beroperasi di bawah DNSSEC, kita memerlukan openssl, perangkat lunak sumber terbuka lapisan soket aman. Ingatlah bahwa ketika kita melewati konfigurasi untuk BIND di awal bab ini, kita mengonfigurasi BIND tanpa openssl dengan menggunakan opsi configure --with-openssl. Sekarang, kita harus menginstal versi dengan openssl. Kita harus menjalankan kembali langkah-langkah konfigurasi, buat, dan buat instal seperti yang ditunjukkan di bawah ini.

```
./configure --prefix=/usr/local --sysconfdir=/etc --with-openssl
make
make install
```

Jika versi BIND kami memiliki openssl, kami sekarang dapat melanjutkan untuk mengonfigurasi BIND ke permintaan layanan dengan DNSSEC. Kita harus membuat Zone Signing Key (ZSK) dan Key Signing Key (KSK). Kami akan menyimpan kunci kami di direktori. Dalam hal ini, kami akan membuat direktori ini di bawah /etc/bind. Untuk menghasilkan kunci, kami akan menggunakan program dnssec-keygen. Empat instruksi berikut akan menyelesaikan tugas kita. Perhatikan bahwa kita meng-hardcoding nama zona (cit.nku.edu) dengan data untuk kunci kita. Kami menggunakan RSA SHA 256 sebagai algoritme enkripsi kami (RSA dinamai menurut pembuatnya, Rivest, Shamir, dan Adelman, SHA adalah Algoritma Secure Hash).

```
mkdir -p /etc/bind/keys/cit.nku.edu
cd /etc/bind/keys/cit.nku.edu/
dnssec-keygen -a RSASHA256 -b 1024 -n ZONE cit.nku.edu
dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE cit.nku.edu
```

Kami sekarang telah menghasilkan dua kunci. Satu, ZSK, adalah 1024 bit. Yang kedua, KSK, adalah 2048 bit. Hasil dari perintah kami di atas adalah pembuatan empat kunci. Kcit.nku.edu.+008+50466. private adalah kunci privat untuk ZSK kita. Kunci ini digunakan oleh

server DNS otoritatif untuk membuat catatan sumber daya tanda tangan digital (RRSIG) untuk file zona cit.nku.edu. Kunci Kcit.nku.edu.+008+50466.key adalah kunci publik untuk ZSK kami. Kunci ini diberikan ke server DNS rekursif apa pun, sehingga mereka dapat memverifikasi catatan sumber daya zona apa pun yang dikirim dari server otoritatif kami kepada mereka. Kunci ketiga adalah Kcit.nku.edu.+008+15511.private. Kunci ini adalah kunci privat untuk KSK kita. Kunci ini digunakan oleh server DNS resmi kami untuk membuat catatan RRSIG. Kunci terakhir, Kcit.nku.edu.+008+15511.key, adalah kunci publik untuk KSK kita. Kunci ini juga dikirim ke server DNS rekursif mana pun oleh server otoritatif kami untuk memvalidasi catatan sumber daya DNSKEY. Kunci pribadi disimpan di server DNS otoritatif kami. Kunci publik diterbitkan dalam catatan sumber daya DNSKEY dari file zona cit.nku.edu.
Berikut ini adalah catatan DNSKEY untuk zona cit.nku.edu. Dua entri berkaitan dengan kunci publik ZSK dan kunci publik KSK.

```
86400   DNSKEY  256 3 8 (
            AwEAAcO5xorSjIW+cvJ0fQ13Gp/U1Yym68cE
            RtnIESSZ41k8oMEgkITJAP6WVoxGCWGuFaGL

            +FD3eZu3JJghebdjU97HQts0M4W+df7y6xtZ
            jTccXiWRDyKwPiwyxd6oxf8Yywgaa/LuWpG+
            lYAyd27XUnQreBgvlTInt7jjoz0m9qxV
      ) ; ZSK; alg = RSASHA256; key id = 50466

86400 DNSKEY 257 3 8 (
            AwEAAb7Tfr0Uelc0+D25MncXvGTCcV7duTvQ
            4eowU0/J3M3f+CTqQY0GdqaFVLL219b3jSoU
            QTLAnqr5tjUdETi7tZenoDYJzQu54gYan5yj
            pUtsY37DGcXaDcdpN6X4W1D20RmrLapHjGEZ
            WWYDRM8xr97q/mVslyhi5MYC49tx4IRZBx//
            zt5BhinkIH2YEs1i6F4PeAGZenDGkVqllM76
            ExflDXX6qBApImXBd+VnsMCDPGlrrWeTdEWW
            ckEmOXkad2X52W98ebVqIVVSz7EvVASUwBra
            0+H1OESl+zibuNkftzEUxfkyRKsgastdKyVM
            KbIQx0Jb3gDpYg3YDKdFOyM=
      ) ; KSK; alg = RSASHA256; key id = 15511
```

Anda mungkin memperhatikan bahwa kunci kedua dua kali lebih panjang dari yang pertama karena kami menetapkan bahwa KSK harus dua kali lebih panjang (2048 bit hingga 1024 bit). Dalam melihat catatan di atas, nilai 86400 adalah TTL untuk kunci ini dalam hitungan detik, sama dengan 1 hari. Nilai 256 dan 257 menunjukkan jenis kunci apa yang menunjukkan jenis masing-masing kunci, di mana 256 berarti ZSK dan 257 berarti KSK. Kami telah meminta protokol 3 dan algoritme kunci 8 untuk kedua kunci. Ini adalah nilai untuk algoritma RSA SHA 256. Kunci sebenarnya ditampilkan di dalam tanda kurung. Kunci publik ini dikodekan menggunakan format Base64.

Sekarang setelah kunci kami tersedia, kami perlu menambahkannya ke file zona kami. Kami melakukan ini dengan menggunakan direktif include. Kami akan menempatkan dua baris berikut ke dalam file zona kami.

```
include /etc/bind/keys/cit.nku.edu/Kcit.nku.edu.+008+15511.key
include /etc/bind/keys/cit.nku.edu/Kcit.nku.edu.+008+50466.key
```

Kita juga harus menandatangani zona kita dengan mengeluarkan perintah zona tanda. Ini adalah bagian dari peranti lunak DNSSEC. Perintah sebenarnya adalah dnssec-signzone. Kami mungkin menentukan instruksi berikut:

```
dnssec-signzone -S -K /etc/bind/keys/cit.nku.edu
     -e +3024000 -N INCREMENT cit.nku.edu
```

Kami akan menerima verifikasi bahwa zona kami telah ditandatangani menggunakan RSA SHA 256, sebagai berikut:

```
Zone fully signed:
Algorithm: RSASHA256:
      KSKs: 1 active, 0 stand-by, 0 revoked
      ZSKs: 1 active, 0 stand-by, 0 revoked
cit.nku.edu.signed
```

Hasilnya adalah file zona yang baru dibuat untuk cit.nku.edu zona kita, bernama cit.nku.edu.signed. Catatan sumber daya RRSIG dibuat dalam file. Contoh record RRSIG ditunjukkan pada Gambar 8.18.

Langkah terakhir kita adalah memodifikasi file konfigurasi named.conf kita untuk menyertakan file zona yang ditandatangani. Untuk ini, kami menambahkan direktif file ke pernyataan zona cit.nku.edu kami untuk menunjukkan file baru ini. Entri zona kami yang telah direvisi ditunjukkan sebagai berikut:

```
www.c1t.nku.edu. 86400 IN  A  10.2.56.220
                 86400 RRSIG  A 8 4 86400 (
                              20151026143003  20150921143003  50466
c1t.nku.edu.
                              PKegHWFYtssXoIKDdPuw7rE1/Ym/gX6PUHRs
                              fFM2KaSvF+PyB+FJ2k4YDHLZI + bFRmjqjnD
                              /0BbxVJxXcNp1K3jyMGBfWcx1/ZX7gyK+hFZ
                               Cs755PZ1sG9K7sUNzkKRsMD4CYhDFs0Tu7er
                              OnEH4Em79a26gt0wMFQXAprQ3GU=    )
                 86400 NSEC  c1t.nku.edu.  A RRSIG  NSEC
                 86400 RRSIG NSEC  8   4   86400   (
                              20151026143003  20150921143003  50466
c1t.nku.edu.
                              BxE1TkqW3zMVp8O9LmotfF1Kmn+FkfpsScJg
                              WXYAW811/N2gphx7k7JDTCSoPkmLT8SOfae
                              Yk1YoII+EQNFCAM6+qDpqtoPV1/mSaSQxZbz
                              N3DpwRZ/gMVor3CyZFMS4K3joX6uXnkUJp9+
                              SvNQM47zJN6sR58+6MtML8t10e0=   )
```

Gambar 8.18 Rekam RRSIG dalam file zona bertanda.

Kami harus me-restart server BIND kami agar perubahan ini berlaku. Kami dapat menguji konfigurasi DNSSEC kami dengan menggunakan perintah dig. Pertama, kami mengambil catatan DNSKEY, seperti yang ditunjukkan pada Gambar 8.19. Kami sekarang

dapat mencari catatan melalui server rekursif dan memastikan bahwa catatan tersebut belum diubah. Pada Gambar 8.20, kita melihat ini melalui perintah dig untuk test.cit.nku.edu. Jika Anda memeriksa gambar ini, Anda akan menemukan bahwa bagian jawaban dan bagian otoritas memiliki kunci yang cocok, memastikan bahwa respons berasal dari otoritas, tanpa diubah dalam perjalanan oleh server rekursif. Dengan demikian, kita dapat mempercayai jawabannya.

```
[root@CIT436001cTemp  named]# dig @127.0.0.1 cit.nku.edu +multiline DNSKEY
;  <<>> DiG  9.10.2-P2  <<>>  @127.0.0.1  cit.nku.edu  +multiline DNSKEY
;  (1 server found)
; ; global options: +cmd
; ; Got answer:
; ;  ->>HEADER<<-opcode: QUERY, status: NOERROR, id:38342
; ; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

; ; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: ; udp: 4096
; ; QUESTION SECTION:
;cit.nku.edu.      IN    DNSKEY

; ; ANSWER SECTION:
cit.nku.edu. 86400 IN DNSKEY  257 3 8 (
                      AwEAAb7Tfr0Uelc0+D25MncXvGTCcV7duTvQ4eowUO/J
                      3M3f+CTqQY0GdgaFVLL219b3jSoUQTLAnqr5tjUdETi7
                      tZenoDYJzQu54gYan5yjpUtsY37DGcXaDcdpN6X4W1D2
                      0RmrLapHjGEZWWYDRM8xr97q/mVslyhi5MYC49tx4IRZ
                      Bx//zt5BhinkIH2YEsli6F4PeAGZenDGkVqllM76Exf1
                      DXX6qBApImXBd+VnsMCDPG1rrWeTdEWWckEmOXkad2X5
                      2W98ebVqIVVSz7EvVASUwBra0+H10ES1+zibuNkftzEU
                      XfkyRKsgastdKyVMKbIQx0Jb3gDpYg3YDKdFoyM=
                      ) ;  KSK;  alg = RSASHA256;  key id = 15511
cit.nku.edu. 86400 IN DNSKEY  256 3  8 (
                       AwEAAc05xorSjIW+cvJ0fQ13Gp/UIYym68cERtnIESSZ
                       41k8oMEgkITJAP6WVoxGCWGuFaGL+FD3eZu3JJghebdj
                      U97HQts0M4W+df7y6xtZjTccXiWRDyKwPiwyxd6oxf8Y
                      ywgaa/LuWpG+1YAyd27XUnQreBgv1Tint7jjoz0m9qxV
                      ) ; ZSK; alg = RSASHA256; key id= 50466

; ; Query time: 3msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Mon Sep 21 14:44:11  EDT 2015
; ;  MSG SIZE rcvd: 464
```

Gambar 8.19 Mengambil record DNSKEY melalui perintah dig.

```
[root@CIT436001cTemp named]# dig @127.0.0.1 test.cit.nku.edu +multiline +dnssec
;  <<>> DiG  9.10.2-P2  <<>>  @127.0.0.1  test.cit.nku.edu  +multiline +dnssec
;  (1 server found)
; ; global options: +cmd
; ; Got answer:
; ;  ->>HEADER<<-opcode: QUERY, status: NOERROR, id:3155
; ; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

; ; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; ; QUESTION SECTION:
;test.cit.nku.edu.            IN    A

; ; ANSWER SECTION:
test.cit.nku.edu. 86400  IN  A  10.2.56.221
test.cit.nku.edu. 86400  IN  RRSIG A 8 4 86400 (
                             20151026143003  20150921143003  50466  cit.nku.edu.
                             NeXr0a/vy/BIbZ7QhW+//5vRCc4o31eseg3RJW1tI/Qo
                             ep3+GOZAGfBvPukyVRjfASYZDZCPsSFaEM4cty7059u9
                             WuGXjyHvfwAJ2MfNFmQ8O19PgwhhAQ2nPvgtstSoIZ7N
                             YEChcmgsjQHjWR36zxvRxMyEEIMgTXON+moO7r8=  )

; ; AUTHORITY SECTION:
cit.nku.edu.      86400  IN   NS   masterdns.nku.edu.
cit.nku.edu.      86400  IN   NS   secondarydns.nku.edu.
cit.nku.edu.      86400  IN   RRSIG  NS  8 3 86400(
                             20151026143003  20150921143003 50466  cit.nku.edu.
                             wmQzEksz1Tw/6gjfO61qIUPtXJgZ1aR94ZMZ2OV+uucV
                             o+y5qAN99bGJo)7pLpuwGl0X3A+ZCLD5nS834oYG/Ge0
                             RMm9LgFEZ1SsSWtLobHwNBL6rUvZL/sAf1qJ5X/bp3kQ
                             CRSY3m3T2a2ACj1x93AT4RcsKBIAI5rDxaZObIY=  )

; ; Query time: 0 msec
; ; SERVER: 127.0.0.1#53(127.0.0.1)
; ; WHEN: Mon Sep 21 14:44:11  EDT 2015
; ;  MSG SIZE rcvd: 454
```

Gambar 8.20 record DNSSEC melalui perintah dig.

DAFTAR PUSTAKA

A. Tanenbaum. 1992. Jaringan Komputer Jilid 1 dan 2, Ketiga. Yogyakarta: Salemba Teknika.

Anjik Sukmaaji & Rianto. 2008. "Jaringan Komputer Konsep Dasar Pengembangan Jaringan & Keamanan Jaringan". Yogyakarta: Andi.

Anonim. (2003). Konsep Jaringan Komputer dan Pengembangannya. Salemba Infotek.

Ariawal, Dian & W Purbo, Ono. 2016. Simulasi Jaringan Komputer dengan Cisco Packet Tracer, Jakarta: PT Elex Media Komputindo.

Arifin, Z. (2017). Pengertian dan Ruang Lingkup Infrastruktur Teknologi Informasi. Pustaka.Ut.Ac.

Asosiasi Penyedia Jasa Internet Indonesia. (2015). Profil Pengguna Internet Indonesia 2014. (Puskakom UI Jakarta, Ed.). Jakarta: Asosiasi Penyedia Jasa Internet Indonesia.

Curran, J., Fenton, N., & Freedman, D. (2016). Misunderstanding The Internet. Pedagoges: An International Journal, Vol. 11(No. 2),

D. Sopandi. 2010. Instalasi dan Konfigurasi Jaringan Komputer. Bandung: Informatika.

Dodi Hriadi. 2012. Solusi Cerdas Menguasai Internetworking Pacet Tracer. Yogyakarta: ANDI.

Eddy Sutanta, 2005. Komunikasi Data & Jaringan Komputer, Jakarta: Graha Ilmu.

Hikmah, N. N. (2020). Mengelola Infrastruktur Bisnis Digital. OSF Preprints.

I. Sofana. 2013. Membangun Jaringan Komputer Mudah Membuat Jaringan Komputer (Wire & Wireless) Untuk Pengguna Windows dan Linux. Bandung: Informatika, 2013.

Ir. Lukas Tanutama dan Ir Hosea., Mengenal Lokal Area Network. PT. Elek Media Komputindo. Jakarta.

Iwan. 2012. CISCO CCNA & Jaringan Komputer. Bandung: Informatika, 2012.

MADCOMS. 2015. Panduan Lengkap Membangun Sendiri Sistem Jaringan Komputer. Yogyakarta: Penerbit Andi.

Munawar, Z., & Putri, N. I. (2020). Keamanan Jaringan Komputer Pada Era Big Data. J-SIKA| Jurnal Sistem Informasi Karya Anak Bangsa, 2(01), 14-20.

Nial Mansfield, 2002, Practical TCP/ IP Designing, Using and Troubleshooting TCP/ IPNetwork on Linux® and Windows®, Pearson Education, Inc.

Nugroho, K. 2016. Jaringan Komputer Menggunakan Pendekatan Praktis . Purwokerto: Penerbit Mediatera.

Onno W. Purbo, 1999, TCP/ IP Standar, Desain dan Implementasi, Jakarta, ISBN no. 979-20-0759-8.

R. Rafiudin. 2006. Sistem Komunikasi Data Mutakhir, C.V ANDI OFFSET, Yogyakarta.

Stalling, William. 2007. “Komunikasi & Jaringan Nirkabel”. Jakarta: Erlangga.

Stallings. 2011. William Network Security Essentials: Apllications and Standards, 4 th edition, Prentice Hall.

Sugeng, Winarno, 2010, Jaringan Komputer dengan TCP/IP. Bandung: Penerbit Modula.

Supriadi, D., Fahmi, H., & Imtihan, K. (2018). Analisa dan Perancangan Infrastruktur Jaringan Wireless Local Area Network (WLAN) Pada Dinas Perindustrian dan Perdagangan Kabupaten Lombok Tengah. Jurnal Informatika dan Rekayasa Elektronik, 1(2), 1-6.

Tanenbaum, Andrew.S. (1997). “Jaringan Komputer”. Jilid 1. Prenhallindo. Jakarta.

Tittel, Ed. 2002. “Schaum’s Outlines Computer Networking”. Jakarta: Erlangga.

Wendel Odom, 2004, Computer Networking First-Step, Cisco Systems, Inc.

Winarno Sugeng. 2010. “Jaringan Komputer dengan TCP/IP”. Bandung: Modula.

INFRASTRUKTUR INTERNET

Jilid 1

Dr. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-51-2 (no.jil.lengkap PDF)
ISBN 978-623-8120-52-9 (jil.1 PDF)