Chapter 8

Understanding Networking Protocols

A network *protocol* is a set of rules that data communications over a network follow to complete various network transactions. For example, TCP/IP defines a set of rules used to send data from one node on a network to another node. SMTP is a set of rules and standards used to transfer e-mail and attachments from one node to another. DHCP is a set of rules and standards used to allocate IP addresses dynamically for a network, so they do not need to be set manually for each workstation.

Many protocols are used in networking. In fact, in a sense, almost *every* activity on a network follows a protocol of one sort or another. Some protocols function at a low level in the OSI network model, others operate at a high level, and some operate in between.

In this chapter, you learn about the essential networking protocols used to transmit and receive data across a network.

Understanding TCP/IP and UDP

As its name suggests, TCP/IP is actually two protocols used in concert with one another. The Internet Protocol (IP) defines how network data is addressed from a source to a destination and in what sequence the data should be reassembled at the other end. IP operates at the network layer in the OSI model. The Transmission Control Protocol (TCP) operates one layer higher than IP, at the transport layer. TCP manages connections between computers. TCP messages are carried (encapsulated) in IP datagrams.

The User Datagram Protocol (UDP) serves the same role as TCP but offers fewer features. Both TCP and UDP packets are carried within IP packets, but the only reliability feature that UDP supports is the resending of any packets not received at the destination. (UDP is called a *connectionless* protocol.) The chief advantage to UDP is that it is much faster for trivial network communications, such as sending a web page

DEFINE-IT! Datagrams, Frames, and Packets

A *packet* is any collection of data sent over a network, and the term is usually used generically to refer to units of data sent at any layer of the OSI model. For instance, people talk about *IP packets*, even though technically the correct term is *IP datagrams.* In this book, *packet* is used generically. The persnickety definition of packet applies only to messages sent at the top layer of the OSI model, the application layer.

Network layer units of data, such as those carried by IP, are called *datagrams.* Units of data carried at the data-link layer (layer 1) are called *frames.*

All of these terms to refer to a collection of data that is transmitted as a single unit.

to a client computer. Because UDP doesn't offer many error-checking or error-handling features, it should be used only when it isn't that important if data occasionally gets mangled between points and needs to be resent, or when an application program provides its own extensive error-checking and error-handling functions.

TCP and UDP Ports

Both TCP and UDP support the concept of *ports*, or application-specific addresses, to which packets are directed on any given receiving machine. For example, most web servers run on a server machine and receive requests through port 80. When a machine receives any packets that are intended for the web server (such as a request to serve up a web page), the requesting machine directs those packets to that port number. When you request a web page from a web server, your computer sends the request to the web server computer and specifies that its request should go to port 80, which is where HTTP requests are directed.

Hundreds of different ports have standardized uses. Defining your own ports on a server for specific applications is easy. A text file called SERVICES defines the ports on a computer. An example of a portion of a Windows SERVICES file follows. (Only selected entries are shown due to space constraints; the following is not a complete SERVICES file, but it illustrates what the file contains.)

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This file contains port numbers for well-known
# services as defined by
# RFC 1700 (Assigned Numbers).
#
# Format:
#
# <service name>port number></protocol> [aliases ...][# <comments>]
#
echo        7/tcp
echo        7/udp
discard     9/tcp sink null
discard     9/udp sink null
systat      11/tcp      users #Active users
daytime     13/tcp
daytime     13/udp
chargen     19/tcp      ttytst source     #Character generator
chargen     19/udp      ttytst source     #Character generator
ftp-data    20/tcp            #FTP, data
ftp         21/tcp            #FTP. control
telnet      23/tcp
smtp        25/tcp      mail  #SMTP
time        37/tcp      timserver
time        37/udp      timserver
```

```
tftp        69/udp            #Trivial File Transfer
gopher      70/tcp
finger      79/tcp
http        80/tcp      www www-http      #World Wide Web
kerberos-sec      88/tcp      krb5  #Kerberos
kerberos-sec      88/udp      krb5  #Kerberos
rtelnet     107/tcp           #Remote Telnet Service
pop2        109/tcp     postoffice  #POP-V2
pop3        110/tcp           #POP v3-
nntp        119/tcp     usenet      #NNTP
ntp         123/udp           #Network Time Protocol
snmp        161/udp           #SNMP
snmptrap    162/udp     snmp-trap   #SNMP trap
print-srv   170/tcp           #Network PostScript
irc         194/tcp           #Relay Chat Prot
ipx         213/udp           #IPX over IP
ldap        389/tcp           #Lightweight DAP
https       443/tcp     MCom
https       443/udp     MCom
who         513/udp     whod
cmd         514/tcp     shell
syslog      514/udp
printer     515/tcp     spooler
router      520/udp     route routed
netnews     532/tcp     readnews
uucp        540/tcp     uucpd
wins        1512/tcp          #Windows Name Service
```

As you can see, most of the Internet services that you might be familiar with actually work through the use of TCP and/or UDP ports, such as HTTP for the Web, SMTP for e-mail, NNTP for Usenet, and so forth. The use of ports ensures that network communications intended for a particular purpose are not confused with others that might also be arriving at the same machine.

Ports allow the receiving machine to direct arriving data appropriately. An example is a server that hosts web pages and also receives and processes e-mail. Packets arriving at port 80 will be sent to the web-serving software, while those that arrive at port 25 will go to the e-mail software. Other services on the machine, such as Telnet and FTP, can also function concurrently through this mechanism.

IP Packets and IP Addressing

IP packets include addresses that uniquely define every computer connected to the Internet (see Figure 8-1). These addresses are used to route packets from a sending node to a receiving node. Because all the routers on the Internet know the network addresses to which they are connected, they can accurately forward packets destined for a remote network.

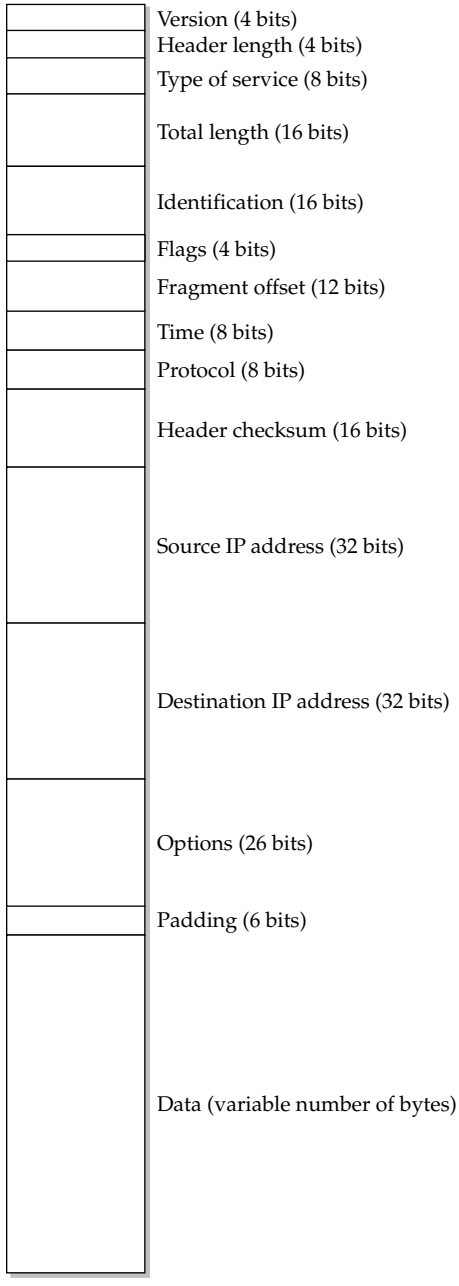


Figure 8-1. A schematic showing the layout of an IP packet

In addition to carrying its data, each IP packet contains a number of fields, which are organized in the following order:

- **Version** This field indicates the version of the IP protocol being used.
- **Header length** This field indicates the length of the header information before the data begins in the packet.
- **Type of service** This field is used for different purposes by different vendors. It can be used for features such as requesting high-priority routing, requesting highest possible reliability, and so forth.
- **Total length** This field indicates the total length of the packet.
- **Identification, flags, and fragment offset** These three fields are used to reassemble an IP packet that was disassembled at some point during transmission. They include all the information necessary for the correct reassembly of the packet at the receiving end.
- **Time to live** This field (called "Time" in Figure 8-1) defines how many network hops the packet can traverse before it is declared dead and the routers stop forwarding it to other routers. This number is set when the packet is sent, and each router that handles the packet decrements the value by one. When the number reaches zero, the packet is dead and is no longer transmitted. If there is a routing configuration error on the path to the destination that causes the packet to go into an endless loop between routers, this is the feature that will stop it after a period of time.
- **Protocol** This field indicates whether the IP packet is contained within a TCP or a UDP packet.
- **Header checksum** The header checksum is used to help ensure that none of the packet's header data (the fields discussed in this list) is damaged.
- **Source IP address** This field contains the address of the sending computer. It is needed in case a packet must be retransmitted, to tell the receiving node (or, in some cases, a router) from which node to request a retransmission.
- **Destination IP address** This field contains the address of the receiving node.
- **Options and padding** These final two fields of the header of the IP packet are used to request any required specific routing instructions or to specify the time that the packet was sent.
- **Data** The final field of an IP packet is the actual data being sent.

IP addresses are 32 bits long, allowing for a theoretical maximum number of addresses of 2^{32}, or about 4.3 billion addresses. To make them easier to work with and to help route them more efficiently, they are broken up into four *octets,* which are each 1 byte long. Thus, in decimal notation, IP addresses are expressed as *xxx.xxx.xxx.xxx,* where each *xxx* represents a base-10 number from 0 to 255. The numbers 0, 127, and 255 are usually reserved for special purposes, so they are typically unavailable for

Help! We're Almost Out of Addresses!

The current implementation of IP, called IP version 4 (IPv4), is approaching the point where running out of addresses is becoming a real possibility. In 1994, a proposal was issued to address this limitation. Called IP Next Generation (IPng, now IP version 6, or IPv6), the new version of IP takes care of the addressing limitation by bumping up the address length from 32 bits to 128 bits. This allows 3.4×10^{38} (34 followed by 37 zeros, or around 340 trillion, trillion, trillion) unique addresses, which should leave plenty of room for all anticipated Internet addresses, even allowing for refrigerators, toasters, and cars to have their own IP addresses!

assignment to nodes. The remaining 253 unique addresses are available for assignment in each octet.

Addresses on the Internet are guaranteed to be unique through the use of an address registration service, presently administered by the Internet Corporation for Assigned Names and Numbers (ICANN). Actual registrations of domain names and addresses are handled through one of many *registrars,* which include companies such as InterNIC, Network Solutions, and many others. ICANN is the overall authority.

ICANN assigns three major classes of addresses, called Class A, B, and C, as follows:

- For a Class A address, ICANN assigns the owner a number in the first octet. The owner is then free to use all possible valid combinations in the remaining three octets. For example, a Class A address might be 57.*xxx*.*xxx*.*xxx*. Class A addresses enable the owner to address up to around 16.5 million unique nodes.
- Class B addresses define the first two octets, leaving the remaining two open for the address's owner to use. For instance, 223.55.*xxx*.*xxx* would be a valid Class B address assignment. Class B addresses enable the holder to have about 65,000 unique nodes.
- Class C follows this progression, defining the first three octets and leaving only the last octet available for the Class C owner to assign. The owner can assign up to 255 unique addresses.

An Internet service provider (ISP) might own either a Class A or a Class B address, and then can handle a number of Class C addresses within its own address structure. Changing ISPs, even for a company that has a valid Class C address, means changing the company's address from a Class C address available through the first ISP to a Class C address available from the new ISP.

As mentioned earlier, the addresses 0, 127, and 255 are reserved. Usually, address 0—as in 123.65.101.0—refers to the network itself, and the router that connects the network to other networks handles this address. The address 127 is a special *loopback address* that can be used for certain kinds of testing. The address 255 refers to all

computers on the network, so a broadcast message to address 223.65.101.255 would go to all addresses within 223.65.101.*xxx*.

IP addresses are made up of two main components. The first, or leftmost, is the *network ID,* also called the *netid*. The other is the *host ID,* usually referred to as *hostid*. The netid identifies the network, while the hostid identifies each node on that network. (In IP parlance, every node is called a *host*, regardless of whether it's a server, client computer, printer, or whatever.) For a Class C address, for instance, the netid is set in the first three octets, and the hostids use the fourth octet. For a Class B address, the first two octets are the netid, and the final two octets are hostids. These address parts are important for subnetting, as described next.

IP Subnetting

Suppose that a company has three networks in three different buildings, all connected by a 64 Kbps ISDN link. Each network has about 25 nodes. Each building has its own set of servers and printers for the workers in that building. The ISDN link between the networks is for the occasional need to transmit information between buildings, such as e-mail messages or accounting transactions. How should the company assign IP addresses in this situation?

The company could request a single Class C set of addresses, and then assign those addresses across the three networks in some fashion. This seems like a simple solution, but it's actually a poor idea for a couple of reasons. Typically, a lot of network traffic is sent to each hostid within a single netid. The slow ISDN link between the buildings would become a tremendous bottleneck in this situation, and the entire network would function very poorly.

Another idea is to use separate Class C addresses (netids) for each building. This is a relatively simple solution, and it would work just fine, except that the ISP might not be able to assign three separate Class C addresses. Also, it would be terribly wasteful of the available pool of IP addresses. In this situation, each building would be wasting more than 200 addresses for no good reason.

What if there were a way to divide a Class C address so that each building could have its own *virtual* netid? Such a solution is what subnetting is all about. *Subnetting* allows you to subdivide a *hostid* range (usually that of a Class C address, but such subnetting can also be done with Class A or B addresses) across two or more networks. Subnetting is done through the use of subnet masks, which are discussed in the next section.

NOTE To understand subnetting, you first need to understand the binary representation of IP addresses. For a quick overview of how binary numbers work, see Chapter 2.

Subnet Masks

If you look at a computer's IP configuration, you'll see that the computer always has both an IP address (such as 205.143.60.109) and a *subnet mask* (such as 255.255.255.0). The subnet mask defines which part of the computer's IP address is the netid and

which part is the hostid. To see this clearly, you need to represent the addresses in binary form:

```
Computer IP Address (Dec):      205        143        60         109
Computer IP Address (Bin):      11001101   10001111   00111100   01101101
Subnet mask (Dec):              255        255        255        0
Subnet mask (Bin):              11111111   11111111   11111111   00000000
```

The netid of an address, defined by the subnet mask, is whatever portion of the address has a binary 1 set in the corresponding subnet mask. In the preceding example, the netid is the full first three octets (the first 24 bits), and the hostid is the last octet (the last 8 bits). Now you can see why 255 (decimal) is used so frequently in subnet masks: 255 corresponds to having all bits set to 1 in an 8-bit number.

NOTE Subnet masks should always use contiguous 1s, starting from the left and working to the right. The hostid portion should contain all contiguous 0s, working backward from the right to the left. While it is theoretically possible to build subnet masks that have interspersed 1s and 0s, it is never done in practice because it would quickly become too complicated to manage properly and because there's no real reason to do so. Also, the portion of the hostid that is subnet-masked cannot consist of all 0s or all 1s. While certain implementations of IP do allow all 0s, such a configuration is not part of the accepted standard IP rules, and thus using such a hostid is risky because some devices on the network might not understand it.

Let's now return to the example of the company with three buildings. What if the company could divide a single Class C address so that each building could use its own portion, and the routers connecting the buildings would understand which transmissions should be forwarded to the other buildings and which ones should not be? Such a configuration is where subnet masks are useful.

A subnet mask allows you to "borrow" some bits from your hostids and then use those bits to create new netids. For the example, you would need to borrow three bits from the Class C address (the fourth octet) and use that address to create four separate netids. Examine how this configuration would work in binary format:

```
Subnet mask (Bin):              11111111   11111111   11111111   111xxxxx
Bldg. 1 IP addresses:           11001101   10001111   00111100   100xxxxx
Bldg. 2 IP addresses:           11001101   10001111   00111100   011xxxxx
Bldg. 3 IP addresses:           11001101   10001111   00111100   101xxxxx
Subnet mask (Dec):              255        255        255        224
Bldg. 1 IP addresses:           205        143        60         129 - 158
Bldg. 2 IP addresses:           205        143        60         97 - 126
Bldg. 3 IP addresses:           205        143        60         161 - 190
```

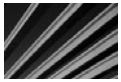
Using this configuration, the company can create up to 6 netids, and each building can be provided with 30 available hostid addresses. By using subnetting to designate each separate netid, the company can program the routers to send packets between networks only when the packets are supposed to be routed.

Binary Mask	Decimal Equivalent	Number of Subnets	Number of Hostids per Subnet
00000000	0	1	254
10000000	128	2	126
11000000	192	4	62
11100000	224	8	30
11110000	240	16	14
11111000	248	32	6
11111100	252	64	2
11111110	254	N/A	N/A
11111111	255	N/A	N/A

Table 8-1. Most Common Subnet Masks

Because subnet masks are usually created using contiguous bits for the mask itself, only nine subnet masks are commonly used, as shown in Table 8-1.

In Table 8-1, some configurations are marked as N/A, for not applicable. These subnet masks would result in no available addresses, because of the rule that the subnet portion of the netid cannot be all 0s or all 1s. For example, consider the subnet mask of 224, which uses three hostid bits for the subnetid. In theory, this configuration should result in eight subnets. However, the subnets represented by 000 and 111 are not valid. Likewise, 128 is not a valid subnet mask because that one bit would always be either a 1 or a 0.



TIP If you need to implement subnets, you should initially work through the project with an experienced network engineer, who can help you avoid pitfalls (which were not explicitly described in the preceding section). You might also want to learn more about TCP/IP through resources devoted to detailed coverage of the concepts introduced here.

Understanding Other Internet Protocols

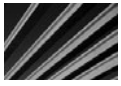
Quite a few other protocols used on the Internet either rely on or make use of TCP/IP. In this section, you learn about these different protocols.

Domain Name System (DNS)

If you had only IP address numbers to address computers over the Internet, trying to keep track of them and using their correct addresses might make you a little crazy. To go to the web site for Google, for example, you would need to remember to type

the address http://209.85.171.100. To solve this problem, a system called the Domain Name System (DNS) was developed.

DNS enables people to register domain names with ICANN and then use them to access a particular node over the Internet. Therefore, DNS is the service that allows you to open a web browser and type http://www.google.com to connect to a particular computer over the Internet. In this case, google.com is the full domain name.



NOTE Domain names are given out on a first-come, first-served basis. However, ICANN gives preference to a holder of a valid registered trademark if a conflict develops. ICANN, upon being presented with valid trademark information and notice of the domain name that infringes on that trademark, goes through a process to assess the truth of the claim and, if necessary, takes a domain name away from its present holder and transfers the name to its rightful owner.

Domains are organized in a tree arrangement, like a directory tree on a disk drive. The top level defines different *domain types*, called *top-level domain names* (TLDs). The most common is the .com domain type, usually used with for-profit commercial entities. The following are other common domain types:

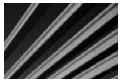
- .edu for educational institutions
- .gov for governmental entities
- .mil for military entities
- .net for Internet-related entities
- .org for nonprofit entities
- **.***xx* for different countries, such as .it for Italy and .de for Germany (Deutschland)

NOTE In recent years, a number of other TLDs have been added to the system, such as .biz, .info, and .name. You can find a complete list of the TLDs at http://www.icann.org.

Within a domain name, entities are free to add other names before the beginning of the domain name, and these usually refer to a particular host or server, or sometimes to a particular type of service for that domain. For example, if you had the domain bedrock.gov, you would be free to create additional names, such as quarry.bedrock.gov and flintstone.bedrock.gov.

As a matter of standards, the first portion of a domain name preceding the actual domain name indicates what type of service is being connected. For instance, www .bedrock.gov would be used for a World Wide Web server for the domain bedrock.gov and ftp.bedrock.gov would be used for an FTP server. The standards for service types within the domain name are usually followed, but not always. The owners of domain names are free to invent their own service types that meet their particular needs. For example, some domain name holders refer to their e-mail servers as smtp.domain.org; others might prefer to use mail.domain.org.

Domain names are resolved to IP addresses through the use of *domain name servers* (DNS servers), which are servers that accept the typed domain name, perform a database query, and then return the actual address that should be used for that domain name. Generally, each ISP maintains its own DNS servers (and many companies and organizations maintain their own DNS servers as well). Any changes are propagated throughout all the Internet's DNS servers within about an hour.



NOTE Changes to DNS entries used to take up to several days to propagate throughout the Internet, but updates to the system now allow changes to propagate much more quickly—often within minutes of the change being posted.

Dynamic Host Configuration Protocol (DHCP)

In the early days of TCP/IP-based networks, administrators defined each node's address in a text file or dialog box. From then on, the address was fixed unless someone changed it. The problem was that administrators occasionally would mistakenly put conflicting addresses into other nodes on the network, causing a network's version of pandemonium. To resolve this problem and to make it easier to assign TCP/IP addresses, a service called Dynamic Host Configuration Protocol (DHCP) was invented.

DHCP services run on a DHCP server, where they control a range of IP addresses called a *scope*. When nodes connect to the network, they contact the DHCP server to get an assigned address that they can use. Addresses from a DHCP server are said to be *leased* to the client that uses them, meaning they remain assigned to a particular node for a set period of time before they expire and become available for another node to use. Often, lease periods are for just a few days, but network administrators can set any time period they want.

You should not use DHCP for nodes that provide network services, particularly for servers that provide services over the Internet. This is because changing a TCP/IP address would make reliably connecting to those computers impossible. Instead, use DHCP to support client workstations that do not need to host services for other nodes.

DEFINE-IT! Host

You might think a *host* is a server, and in some networking contexts, you would be right. However, in the jargon of Internet names and addresses, every computer that has an IP address is called a *host*, thus the name, Dynamic Host Configuration Protocol. Remembering that every computer is called a host is particularly important in the UNIX and Linux worlds, where the term is much more common than in the Windows or Macintosh worlds.

Hypertext Transfer Protocol (HTTP)

The World Wide Web is made up of documents that use a formatting language called Hypertext Markup Language (HTML). These documents are composed of text to be displayed, graphic images, formatting commands, and hyperlinks to other documents located somewhere on the Web. HTML documents are displayed most often using web browsers, such as Mozilla Firefox or Microsoft Internet Explorer.

A protocol called Hypertext Transfer Protocol (HTTP) controls the transactions between a web client and a web server. HTTP is an application-layer protocol. The HTTP protocol transparently makes use of DNS and other Internet protocols to form connections between the web client and the web server, so the user is aware of only the web site's domain name and the name of the document itself.

HTTP is fundamentally an insecure protocol. Text-based information is sent "in the clear" between the client and the server. To address the need for secure web networking, alternatives are available, such as HTTP Secure (HTTPS) and Secure Sockets Layer (SSL).

Requests from a web client to a web server are connection-oriented, but they are not persistent. Once the client receives the contents of an HTML page, the connection is no longer active. Clicking a hyperlink in the HTML document reactivates the link, either to the original server (if that is where the hyperlink points) or to another server somewhere else.

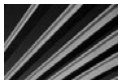
File Transfer Protocol (FTP)

The acronym FTP stands for two things: File Transfer Protocol and File Transfer Program (which makes use of the File Transfer Protocol). It's sort of like, "it's a dessert topping *and* a floor polish," (from the *Saturday Night Live* TV show). Because FTP (the program) makes use of FTP (the protocol), it can become confusing to know which is being discussed. This section discusses the protocol. (When I'm referring to the program, I'll say so.)

FTP is an application-layer protocol used to send and receive files between an FTP client and an FTP server. Usually, this is done with the FTP program or another program that can also use the protocol (many are available). FTP transfers can be either text-based or binary-based, and they can handle files of any size.

When you connect to an FTP server to transfer a file, you log in to the FTP server using a valid username and password. However, some sites are set up to allow *anonymous FTP*, where you enter the username *anonymous* and then enter your e-mail address as the password. For example, Microsoft maintains an FTP site you can use to download updates to its products, located at ftp.microsoft.com, which allows anonymous FTP.

To use the FTP program, on most platforms you type the command **ftp** followed by the address to which you want to connect. So, to use the Microsoft example, you would type **ftp.microsoft.com**, press ENTER, and then log in. Then you can use all of the FTP commands—PUT, GET, MGET, and so forth. Most FTP program implementations have online help to assist you with the various commands. Type **?** or HELP to access this feature.



TIP Recent versions of Windows also support FTP connections using Internet Explorer. Just open Internet Explorer and instead of entering an http:// address in the address bar, type an address preceded by ftp://. For example, to connect to Microsoft's FTP server, you would use the address ftp://ftp.microsoft.com. This trick also works in most other current web browsers, such as Mozilla Firefox. Note that for FTP sites that require a login, the browser must support logging in. In Internet Explorer, a Logon As option is available on the File menu after you browse to an FTP site.

Network News Transfer Protocol (NNTP)

Usenet (NetNews) is a set of discussion groups devoted to an extremely wide variety of topics. There are well over 100,000 such such groups in existence. Usenet conversations are posted to Usenet servers, which then echo their messages to all other Usenet servers around the world. A posted message can travel to all the Usenet servers in a matter of hours, and then be available to users accessing any particular Usenet server.

Usenet discussion groups are loosely organized into the branches of a tree. The following are some of the main branches:

- Alt, for discussions about alternative lifestyles and other miscellaneous topics
- Comp, for computer-oriented discussions
- Gov, for government-oriented discussions
- Rec, devoted to recreational topics
- Sci, for science-based discussions

Usenet groups can either be public, which are echoed to other Usenet servers, or private, which are usually hosted by a particular organization and require the user to enter appropriate login credentials before reading and posting messages.

The NNTP protocol is what makes Usenet possible. It allows for a connection between a Usenet reader (also called a *news reader*) and a Usenet server. It also provides for message formatting, so messages can be text-based or can also contain binary attachments. Binary attachments in Usenet postings are usually encoded using Multipurpose Internet Message Encoding (MIME), which is also used for most e-mail attachments. Some older systems use different methods to encode attachments, including one method called UUEncode/UUDecode and, on the Macintosh, a method called BinHex.

Telnet

Telnet defines a protocol that allows a remote terminal session to be established with an Internet host, so remote users have access similar to using a terminal connected directly to the host computer. Using Telnet, users can control the remote host, performing tasks such as managing files, running applications, or even (with appropriate permissions) administering the remote system. Telnet is a session-layer protocol in the OSI model.

For Telnet to work, Telnet software must be running on both the server and client computer. You run the program Telnet on a client computer and run the program Telnetd on the server computer to allow the connection. Telnet is specific to the TCP protocol

and typically runs on port 23 (although it can run on any port that has been enabled on the server system). Once users connect using Telnet, they must log in to the remote system using the same credentials they would use if they were working from a directly connected terminal.

Simple Mail Transfer Protocol (SMTP)

E-mail had a somewhat rocky start on the Internet, with early e-mail programs sharing few standards with other e-mail programs, particularly in the handling of attached binary data. The good news is that the situation is now resolved, and all current e-mail software supports all the widely accepted standards.

The Simple Mail Transfer Protocol (SMTP) is used to send and receive e-mail messages from one e-mail server to another. The SMTP protocol defines a dialog between a sending system and a receiving system.

An SMTP dialog starts when a sending system connects to port 25 of a receiving system. After the connection is established, the sending system sends a HELO command, followed by its address. The receiving system acknowledges the HELO command along with its own address. The dialog then continues, with the sending system issuing a command indicating that the system wants to send a message and identifying the recipient for whom the message is intended. If the receiving system knows of the recipient, it acknowledges the request, and then the sending system transmits the body of the message along with any attachments. Finally, the connection between the two systems is terminated once the receiving system acknowledges that it has received the entire message. Figure 8-2 illustrates this process.

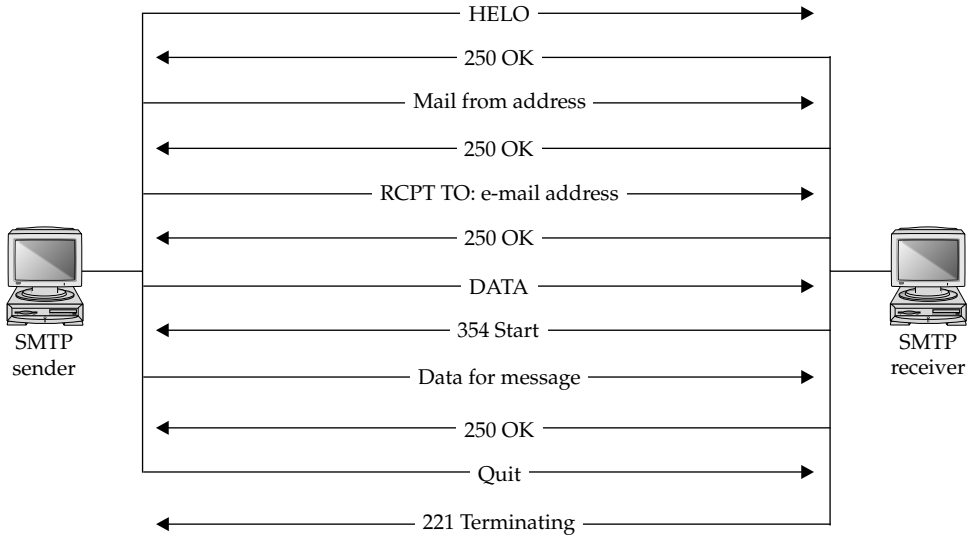
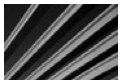


Figure 8-2. Part of an SMTP dialog between systems



TIP Details on SMTP can be found in RFC 821 (http://www.faqs.org/rfcs/rfc821.html).

Voice over IP (VoIP)

An important emerging set of IP protocols concerns the transmission of voice and facsimile information over IP-based networks, called *Voice over IP,* or *VoIP* for short (pronounced "voyp"). VoIP is a protocol that allows analog voice data—for telephone calls—to be digitized and then encapsulated into IP packets and transmitted over a network. VoIP can be used to carry voice telephone calls over any IP network, such as a company's local area network (LAN) or wide area network (WAN), or the Internet.

Sending voice data over IP networks has some very attractive possible payoffs. One is more efficient use of available connections.

Consider a large company with two main offices. At any given time, hundreds of voice conversations might be occurring between those two offices. Each traditional voice connection consumes one DS0 line, capable of carrying up to 56 Kbps of data if the line were used digitally. Each conversation does not use all of the available bandwidth on the line. Part of this is because most conversations have a lot of silent spaces—time between words or sentences, time where one party stops speaking and the other starts, and so forth. Plus, most conversations, were they encoded digitally, could be significantly compressed. Add all of this up, and each voice conversation is likely to use only one-third to one-half of the available bandwidth on a single DS0 circuit.

If you were able to carry all of these voice conversations digitally, much less bandwidth would be required. Instead of 100 DS0 lines for 100 conversations, for example, the same conversations might use up only 25 to 33 DS0 lines if they were digitally packaged. Many companies can save a significant amount of money by using VoIP.

Another advantage of VoIP is that the connections are packet-oriented. When the user places a call, a single connection is formed between the caller and the receiver. This connection is static for the duration of the call. If the conversation were digitized and sent over a packet-oriented network, however, many possible paths would be available for each packet, and much more redundancy would be automatically available. For instance, if some portion of the network between the two points went down, the packets could still arrive at their destination through an alternate route, just as data packets do over the Internet. Also, available circuits would be used more efficiently, allowing more calls to be routed within a particular geographic area.

VoIP also has some disadvantages that you need to consider:

- **No guaranteed delivery** VoIP does not guarantee delivery of IP packets over the Internet. For a digital transmission of data, this is no big deal; if a packet isn't confirmed as being received, it is simply retransmitted. For a real-time voice conversation, the loss of packets directly inhibits the conversation, and you can't go back in time to retransmit missing packets.

- **Out-of-sequence packets** Not only can IP packets simply fail to arrive at their destination on occasion, but sometimes they arrive out of sequence due to other Internet traffic and other reasons. This is fine for transmitting things such as files, because the packets can be reassembled on the other end in the proper sequence once they are all received. For a real-time application such as voice, however, having packets arrive out of sequence results in a hopelessly jumbled, and thus useless, transmission.
- **QoS not widely implemented** Real-time uses of the Internet, such as VoIP or multimedia streaming and time-sensitive transmissions, should be given priority over transmissions that are not particularly time-sensitive, such as the transmission of an e-mail message. Fortunately, IP has a quality of service (QoS) field that enables the user to prioritize traffic for such reasons. However, QoS is not widely implemented in all parts of the Internet.

VoIP is a hot, emerging technology that is virtually certain to become an important part of the Internet and most companies' networks. However, there is still much work to be done toward actually implementing this technology widely and solving the problems outlined in this section. In other words, if you're learning about networking, you should be aware of VoIP—what it is and what it does—although the technology is still relatively early on the adoption curve.

NOTE There are a number of companies offering VoIP services for residential customers, including AT&T, Vonage, Verizon, and Time Warner Cable. These companies provide packages that allow virtually unlimited calling over an existing high-bandwidth Internet connection for as little as $30 additional per month. They often package the necessary VoIP hardware with a subscription agreement.

Comparing Important Proprietary Protocols

While Microsoft-based, Novell-based, and Apple-based networks can work with TCP/IP and all the previously discussed protocols, each type of network got its start supporting proprietary protocols unique to the company, and each of these protocols can still be found in current networks. All these companies have embraced TCP/IP and support it fully, both for servers and for network clients.

Microsoft and Novell networks (as of Windows NT 4 and Novell NetWare 5) can be easily deployed using only TCP/IP. In theory, you could do the same thing with an Apple-based network, but you would lose a good deal of the Macintosh's network functionality if you did so. Because of this, an Apple-based network should support both AppleTalk (Apple's proprietary protocol) and TCP/IP.

Novell networks originally used the Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) protocols. These are not the same as TCP/IP, but they are comparable. IPX is analogous to IP, and SPX is analogous to TCP.

Microsoft networks were originally based on an IBM-developed protocol called Network Basic Input/Output System (NetBIOS). NetBIOS is a relatively high-level protocol that, in essence, extends the functionality of DOS to a network. Microsoft also used IBM's NetBIOS Extended User Interface (NetBEUI), an enhancement to NetBIOS.

Apple Macintosh computer networks originally supported only AppleTalk. The protocol was designed expressly for the purpose of sharing Apple LaserWriter printers within small workgroups using a low-bandwidth (230 Kbps originally) network media called LocalTalk. Over time, Apple extended AppleTalk somewhat to enable file sharing and other network functions. However, AppleTalk is still an extremely inefficient network protocol that, even over Ethernet (called EtherTalk in Apple's implementation), works slowly.

Novell's IPX/SPX

Novell's IPX protocol was originally a derivative of the Xerox Network Systems (XNS) architecture and closely resembles it. While IPX can be used on any of the popular network media (Ethernet, Token Ring, and so forth), it was originally designed for Ethernet networks and works best with that media. In fact, the IPX protocol depends on Ethernet MAC addresses for part of its own addresses. IPX addresses are dynamic and are automatically negotiated with the server at login, rather than being statically set, as is the case with TCP/IP without DHCP services.

An IPX network address is composed of both a 32-bit network address and a 48-bit node address. In addition, another 16 bits are used for a connection ID, which allows up to 65,000 unique connections between a client and a server. The address design of IPX theoretically allows for about 281 trillion nodes on each of 16 million networks.

IPX was originally designed only for LANs, but it has been enhanced to support WAN connections. While typically considered a "chatty" protocol that requires a lot of send/acknowledgment transactions, IPX has been enhanced with burst-mode capabilities, which increase the size of packets destined for a WAN and decrease the number of back-and-forth communications required. IPX can be routed, but only if the network includes an IPX-capable router.

NetBIOS and NetBEUI

IBM originally developed NetBIOS and NetBEUI to support small networks. Microsoft adopted the protocols as part of LAN Manager, a network operating system built on top of early versions of the OS/2 operating system.

Neither protocol is routable, so each is suitable only for small LANs that do not rely on routers between different LAN segments. However, NetBIOS can be encapsulated within TCP/IP packets on Windows networks using a service called NetBIOS over TCP/IP (abbreviated as NBT).

Microsoft LANs (prior to Windows 2000) rely on a NetBIOS service called NetBIOS Names to identify each workstation uniquely. In a simple NetBIOS implementation, names are registered with all workstations through a broadcast message. If no computer has already registered a particular name, the name registration succeeds. In a more

complex Windows–based network that also uses TCP/IP, however, the NetBIOS names resolve to TCP/IP addresses through the use of Windows Internet Name Service (WINS). The names can also be resolved using static name definition entries contained in a file called LMHOSTS (for LAN Manager HOSTS).

Because some networking applications still use NetBIOS Names, either WINS or LMHOSTS allows such applications to continue to function in a TCP/IP-only network. As far as the application is concerned, it is still working with NetBIOS, while TCP/IP performs the actual work in the background.

AppleTalk

AppleTalk has been extended into AppleTalk Phase 2, which now allows routing of AppleTalk packets (assuming an AppleTalk Phase 2-capable router). The Phase 2 variant can run over Ethernet, Token Ring, or Apple's LocalTalk media. Under Ethernet, AppleTalk uses a variant of the 802.2 frame type called Ethernet Subnetwork Access Point (SNAP).

AppleTalk has an important history for Apple Macintosh networking, but Apple now fully supports and recommends TCP/IP for its computers.

Chapter Summary

This chapter is built on the knowledge you gained in earlier chapters, delving into various important protocols involved in virtually all networks, including the Internet. You learned primarily about the TCP/IP protocol, which has essentially displaced older protocols such as IPX/SPX and NetBIOS/NetBEUI (although these older protocols are still used). You also learned about some specific application-layer Internet protocols, such as SMTP, DHCP, and HTTP. These are all vital protocols to understand for any networking professional.

It would be nice if the protocols discussed in this chapter were all you had to contend with, but, unfortunately, many more protocols exist. Some are specific to certain functions, such as remote access to a network, and are discussed in appropriate chapters within this book. Others are still being developed and are not a factor now, but may be in the near future. You will certainly want to stay up-to-date with emerging protocols that may become important to networking.

The next chapter is about directory services, which make complex networks easier to use and administer.

This page intentionally left blank