Chapter

6

Introduction to Internet Protocol (IP)

THE FOLLOWING COMPTIA NETWORK+ EXAM OBJECTIVES ARE COVERED IN THIS CHAPTER:

✓ **1.1 Explain the function of common networking protocols**

- TCP
- FTP
- UDP
- TCP/IP suite
- DHCP
- TFTP
- DNS
- HTTP(S)
- ARP
- SIP (VoIP)
- RTP (VoIP)
- SSH
- POP3
- NTP
- IMAP4
- TELNET
- SMTP
- SMNP2/3
- ICMP
- IGMP
- TLS

✓ **1.2 Identify commonly used TCP and UDP default ports**

- TCP ports
  - FTP – 20, 21
  - SSH – 22
  - TELNET – 23
  - SMTP – 25
  - DNS – 53
  - HTTP – 80
  - POP3 – 110
  - NTP – 123
  - IMAP4 – 143
  - HTTPS – 443
- UDP ports
  - TFTP – 69
  - DNS – 53
  - BOOTPS/DHCP – 67
  - SNMP – 161

✓ **1.4 Given a scenario, evaluate the proper use of the following addressing technologies and addressing schemes**

- DHCP (static, dynamic APIPA)

The *Transmission Control Protocol/Internet Protocol (TCP/IP)* suite was created by the Department of Defense (DoD) to ensure and preserve data integrity, as well as to maintain communications in the event of catastrophic war. So it follows that if designed and implemented correctly, a TCP/IP network can truly be a solid, dependable, and resilient network solution. In this chapter, I'll cover the protocols of TCP/IP.

I'll begin by covering the DoD's version of TCP/IP and then compare this version and its protocols with the OSI reference model discussed in Chapter 2, "The Open Systems Interconnection Specifications."

After going over the various protocols found at each layer of the DoD model, I'll finish the chapter by providing a more detailed explanation of data encapsulation that I started in Chapter 2.

NOTE

For up-to-the-minute updates for this chapter, please see `www.lammle.com` or `www.sybex.com/go/comptianetwork+studyguide`.

Introducing TCP/IP

Because TCP/IP is so central to working with the Internet and intranets, it's essential for you to understand it in detail. I'll begin by giving you some background on TCP/IP and how it came about, and then move on to describing the important technical goals defined by the original designers. After that, you'll find out how TCP/IP compares to a theoretical model—the Open Systems Interconnection (OSI) model.

A Brief History of TCP/IP

The very first Request for Comments (RFC) was published in April 1969, which paved the way for today's Internet and its protocols. Each of these protocols is specified in the multitude of RFCs, which are observed, maintained, sanctioned, filed, and stored by the Internet Engineering Task Force (IETF).

TCP/IP first came on the scene in 1973. Later, in 1978, it was divided into two distinct protocols: TCP and IP. Then, back in 1983, TCP/IP replaced the Network Control Protocol

(NCP) and was authorized as the official means of data transport for anything connecting to ARPAnet, the Internet's ancestor that was created by ARPA, the DoD's Advanced Research Projects Agency way back in 1957 in reaction to the Soviet's launching of Sputnik. ARPA was soon re-dubbed DARPA, and it was divided into ARPAnet and MILNET (also in 1983); both were finally dissolved in 1990.

But contrary to what you might think, most of the development work on TCP/IP happened at UC Berkeley in Northern California, where a group of scientists were simultaneously working on the Berkeley version of UNIX, which soon became known as the BSD, or Berkeley Software Distribution series of UNIX versions. Of course, because TCP/IP worked so well, it was packaged into subsequent releases of BSD UNIX and offered to other universities and institutions if they bought the distribution tape. So basically, BSD UNIX bundled with TCP/IP began as shareware in the world of academia, and as a result, became the basis of the huge success and exponential growth of today's Internet as well as smaller, private and corporate intranets.

As usual, what may have started as a small group of TCP/IP aficionados evolved, and as it did, the U.S. government created a program to test any new published standards and make sure they passed certain criteria. This was to protect TCP/IP's integrity and to ensure that no developer changed anything too dramatically or added any proprietary features. It's this very quality—this open-systems approach to the TCP/IP family of protocols—that pretty much sealed its popularity, because it guarantees a solid connection between myriad hardware and software platforms with no strings attached.

TCP/IP and the DoD Model

The DoD model is basically a condensed version of the OSI model—it's composed of four, instead of seven, layers:

- Process/Application layer
- Host-to-Host layer
- Internet layer
- Network Access layer

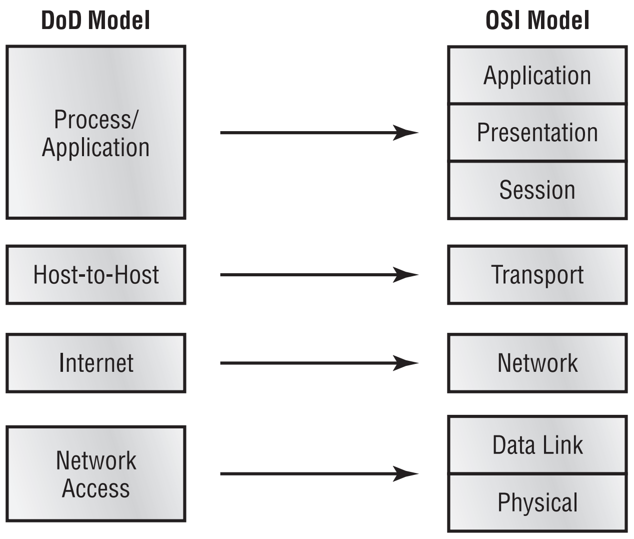
Figure 6.1 shows a comparison of the DoD model and the OSI reference model. As you can see, the two are similar in concept, but each has a different number of layers with different names.

NOTE

When the different protocols in the IP stack are discussed, the layers of the OSI and DoD models are interchangeable. In other words, the Internet layer and the Network layer describe the same thing, as do the Host-to-Host layer and the Transport layer. The other two layers of the DoD model are composed of multiple layers of the OSI model.

A vast array of protocols combine at the DoD model's *Process/Application layer* to integrate the various activities and duties spanning the focus of the OSI's corresponding top three layers (Application, Presentation, and Session). We'll be looking closely at those protocols in the next part of this chapter. The Process/Application layer defines protocols for node-to-node application communication and also controls user-interface specifications.

FIGURE 6.1 The DoD and OSI models



The *Host-to-Host layer* parallels the functions of the OSI's Transport layer, defining protocols for setting up the level of transmission service for applications. It tackles issues such as creating reliable end-to-end communication and ensuring the error-free delivery of data. It handles packet sequencing and maintains data integrity.

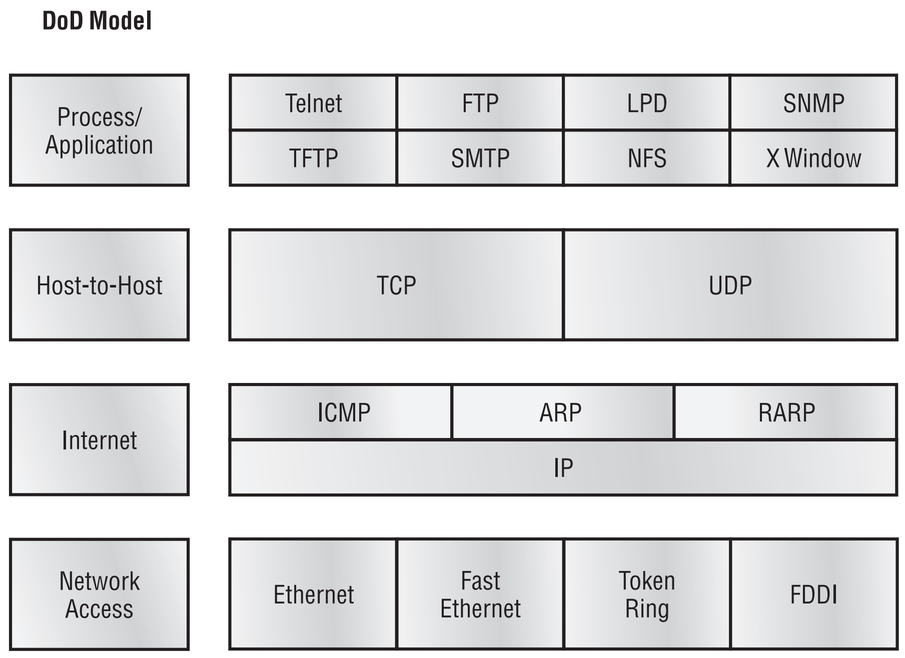
The *Internet layer* corresponds to the OSI's Network layer, designating the protocols relating to the logical transmission of packets over the entire network. It takes care of the addressing of hosts by giving them an IP address, and it handles the routing of packets among multiple networks.

At the bottom of the DoD model, the *Network Access layer* monitors the data exchange between the host and the network. The equivalent of the Data Link and Physical layers of the OSI model, the Network Access layer oversees hardware addressing and defines protocols for the physical transmission of data.

The DoD and OSI models are alike in design and concept and have similar functions in similar layers. Figure 6.2 shows the TCP/IP protocol suite and how its protocols relate to the DoD model layers.

In the following sections, we'll look at the different protocols in more detail, starting with the Process/Application layer protocols.

FIGURE 6.2 The TCP/IP protocol suite



The Process/Application Layer Protocols

In this section, I'll describe the different applications and services typically used in IP networks.

Telnet

Telnet is the chameleon of protocols—its specialty is terminal emulation. It allows a user on a remote client machine, called the Telnet client, to access the resources of another machine, the Telnet server. Telnet achieves this by pulling a fast one on the Telnet server and making the client machine appear as though it were a terminal directly attached to the local network. This projection is actually a software shell—a virtual terminal that can interact with the chosen remote host.

These emulated terminals are of the text-mode type and can execute refined procedures such as displaying menus that give users the opportunity to choose options and access the applications on the duped server. Users begin a Telnet session by running the Telnet client software and then logging into the Telnet server.

Telnet offers no security or encryption and is being replaced by Secure Shell (SSH) when security across the remote-configuration session is needed or desired.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the protocol that actually lets you transfer files across an IP network, and it can accomplish this between any two machines using it. But FTP isn't just a protocol; it's also a program. Operating as a protocol, FTP is used by applications. As a program, it's employed by users to perform file tasks by hand. FTP also allows for access to both directories and files and can accomplish certain types of directory operations, such as relocating files into different directories. FTP can team up with Telnet to transparently log you into the FTP server and then provides for the transfer of files.

Accessing a host through FTP is only the first step, though. Users must then be subjected to an authentication login that's probably secured with passwords and usernames implemented by system administrators to restrict access. You can get around this somewhat by adopting the username *anonymous*—although what you'll gain access to will be limited.

Even when employed by users manually as a program, FTP's functions are limited to listing and manipulating directories, typing file contents, and copying files between hosts. It can't execute remote files as programs.

Secure File Transfer Protocol (SFTP)

Secure File Transfer Protocol (SFTP) is used when you need to transfer files over an encrypted connection. It uses an SSH session (more on this later), which encrypts the connection. Apart from the secure part, it's used just as FTP is—for transferring files between computers on an IP network, such as the Internet.

Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol (TFTP) is the stripped-down, stock version of FTP, but it's the protocol of choice if you know exactly what you want and where to find it—plus it's easy to use, and it's fast too! It doesn't give you the abundance of functions that FTP does, though. TFTP has no directory-browsing abilities; it can do nothing but send and receive files. This compact little protocol also skimps in the data department, sending much smaller blocks of data than FTP; and there's no authentication as with FTP, so it's insecure. Few sites support it because of the inherent security risks.

Real World Scenario

When Should You Use FTP?

The folks at your San Francisco office need a 50MB file emailed to them right away. What do you do? Most email servers would reject the email because they have size limits. Even if there's no size limit on the server, it would still take a while to send this big file. FTP to the rescue!

If you need to give someone a large file or you need to get a large file from someone, FTP is a nice choice. Smaller files (less than 5MB) can be sent via email if you have the bandwidth of DSL or a cable modem. However, most ISPs don't allow files larger then 5MB to be emailed; so FTP is an option you should consider if you need to send and receive large files, even if they're compressed. (Who doesn't, these days?) To use FTP, you'll need to set up an FTP server on the Internet so that the files can be shared.

Besides, FTP is faster than email, which is another reason to use FTP for sending or receiving large files. In addition, because it uses TCP and is connection-oriented, if the session dies, FTP can sometimes start up where it left off. Try that with your email client!

Network File System (NFS)

Network File System (NFS) is a jewel of a protocol specializing in file sharing. It allows two different types of file systems to interoperate. It works like this: Suppose the NFS server software is running on an NT server and the NFS client software is running on a UNIX host. NFS allows for a portion of the RAM on the NT server to transparently store UNIX files, which can, in turn, be used by UNIX users. Even though the NT file system and UNIX file system are unalike—they have different case sensitivity, filename lengths, security, and so on—both UNIX users and NT users can access that same file with their normal file systems, in their normal way.

Simple Mail Transfer Protocol (SMTP)

Simple Mail Transfer Protocol (SMTP), answering our ubiquitous call to email, uses a spooled, or queued, method of mail delivery. Once a message has been sent to a destination, the message is spooled to a device—usually a disk. The server software at the destination posts a vigil, regularly checking the queue for messages. When it detects them, it proceeds to deliver them to their destination. SMTP is used to send mail; POP3 is used to receive mail.

Post Office Protocol (POP)

Post Office Protocol (POP) gives us a storage facility for incoming mail, and the latest version is called POP3 (sound familiar?). Basically, how this protocol works is when a client device connects to a POP3 server, messages addressed to that client are released for downloading. It doesn't allow messages to be downloaded selectively; but once they are, the client/server interaction ends and you can delete and tweak your messages locally at will. Lately we're seeing a newer standard, IMAP, being used more and more in place of POP3. Why?

Internet Message Access Protocol, Version 4 (IMAP4)

Because Internet Message Access Protocol (IMAP) makes it so you get control over how you download your mail, with it, you also gain some much-needed security. It lets you peek at the message header or download just a part of a message—you can now just nibble at the bait instead of swallowing it whole and then choking on the hook hidden inside!

With it, you can choose to store messages on the email server hierarchically, and link to documents and user groups too. IMAP even gives you search commands to use to hunt for messages based on their subject, header, or content. As you can imagine, it has some serious authentication features—it actually supports the Kerberos authentication scheme that MIT developed. And yes, IMAP4 is the current version.

Transport Layer Security (TLS)

Both Transport Layer Security (TLS) and its forerunner, Secure Sockets Layer (SSL), are cryptographic protocols that come in really handy for enabling secure online data-transfer activities like browsing the Web, instant messaging, internet faxing, and so on. They're so similar it's not within the scope of this book to detail the differences between them.

SIP (VoIP)

Session Initiation Protocol (SIP) is a hugely popular signaling protocol used to construct and deconstruct multimedia communication sessions for many things like voice and video calls, video conferencing, streaming multimedia distribution, instant messaging, presence information, and online games over the Internet.

RTP (VoIP)

Real-time Transport Protocol (RTP) describes a packet-formatting standard for delivering audio and video over the Internet. Although initially designed as a multicast protocol, it's now used for unicast applications too. It's commonly employed for streaming media, video-conferencing, and push-to-talk systems—all things that make it a de-facto standard in Voice over IP industries.

Line Printer Daemon (LPD)

The Line Printer Daemon (LPD) protocol is designed for printer sharing. The LPD, along with the Line Printer (LPR) program, allows print jobs to be spooled and sent to the network's printers using TCP/IP.

X Window

Designed for client/server operations, *X Window* defines a protocol for writing client/server applications based on a graphical user interface (GUI). The idea is to allow a program, called a client, to run on one computer and have it display things through a window server on another computer.

Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP) collects and manipulates valuable network information. It gathers data by polling the devices on the network from a management station at fixed or random intervals, requiring them to disclose certain information. When all is well, SNMP receives something called a *baseline*—a report delimiting the operational traits of a healthy network. This protocol can also stand as a watchdog over the network, quickly notifying managers of any sudden turn of events. These network watchdogs are called *agents*, and when aberrations occur, agents send an alert called a *trap* to the management station. In addition, SNMP can help simplify the process of setting up a network as well as the administration of your entire internetwork.

Secure Shell (SSH)

Secure Shell (SSH) protocol sets up a secure Telnet session over a standard TCP/IP connection and is employed for doing things like logging into other systems, running programs on remote systems, and moving files from one system to another. And it does all of this while maintaining a nice, strong, encrypted connection. You can think of it as the new-generation protocol that's now used in place of `rsh` and `rlogin`—even Telnet.

SNMP versions 1, 2, and 3

SNMP versions 1 and 2 are pretty much obsolete. This doesn't mean you won't see them in a network at some time, but v1 is super old and, well, obsolete. SNMPv2 provided improvements, especially in security and performance. But one of the best additions was what was called "GETBULK", which allowed a host to retrieve a large amount of data at once. However, v2 never really caught on in the networking world. SNMPv3 is now the standard and uses both TCP and UDP, unlike v1, which used only UDP. V3 added even more security and message integrity, authentication, and encryption. So, be careful when running SNMPv1 and v2 as they are susceptible to a packet sniffer reading the data.

Hypertext Transfer Protocol (HTTP)

All those snappy websites comprising a mélange of graphics, text, links, and so on—the Hypertext Transfer Protocol (HTTP) is making it all possible. It's used to manage communications between web browsers and web servers and opens the right resource when you click a link, wherever that resource may actually reside.

Hypertext Transfer Protocol Secure (HTTPS)

The Hypertext Transfer Protocol Secure (HTTPS) is also known as Secure Hypertext Transfer Protocol. Sometimes you'll see it referred to as SHTTP or S-HTTP, but no matter—as indicated, it's a secure version of HTTP that arms you with a whole bunch of security tools for keeping transactions between a web browser and a server secure. It's what your browser needs to fill out forms, sign in, authenticate, and encrypt an HTTP message when you make a reservation or buy something online.

NOTE

Both SSH (port 22) and HTTPS (port 443) is used to encrypt packets over your intranet and the internet.

Network Time Protocol (NTP)

Kudos to Professor David Mills of the University of Delaware for coming up with this handy protocol that's used to synchronize the clocks on our computers to one standard time source (typically, an atomic clock). Network Time Protocol (NTP) works in conjunction with other synchronization utilities to ensure all computers on a given network agree on the time. This may sound pretty simple, but it's very important because so many of the transactions done today are time- and date-stamped. Think about your precious databases, for one. It can mess up a server pretty badly if it's out of sync with the machines connected to it, even by mere seconds (think crash!). You can't have a transaction entered by a machine at, say, 1:50 a.m., when the server records that transaction as having occurred at 1:45 a.m. So basically, NTP works to prevent "back to the future sans DeLorean" from bringing down the network—very important indeed!

NOTE The Requests for Comments (RFCs) form a series of notes, started in 1969, about the Internet (originally the ARPAnet). The notes discuss many aspects of computer communication; they focus on networking protocols, procedures, programs, and concepts but also include meeting notes, opinion, and sometimes humor. You can find the RFCs by visiting `www.iana.org`.

Network News Transfer Protocol (NNTP)

Network News Transfer Protocol (NNTP) is how you access the Usenet news servers that hold the legion of specific message boards called *newsgroups*. As you likely know, these groups represent pretty much any special interest humans have under the sun. For instance, if you happen to be a classic car buff or a WWII aircraft enthusiast, odds are good there're lots of newsgroups available to join based upon those interests. NNTP is specified in RFC 977. And because it's complicated to configure a news reader program, lots of websites—even search engines—are the entities we usually depend upon to access these many and varied resources.

Secure Copy Protocol (SCP)

FTP is great. It's a super easy, user-friendly way to transfer files—if you don't need to transfer those files securely. That's because when you use FTP for transferring data, usernames and passwords get sent right along with the file request in the clear for all to see with no encryption whatsoever! Kind of like Hail Mary passes, you basically just throw them out there and hope your information doesn't fall into the wrong hands and get intercepted.

That's where Secure Copy Protocol (SCP) comes to your rescue—its whole purpose is to protect your precious files. Through SSH, it first establishes and then sustains a secure, encrypted connection between the sending and receiving hosts until file transfer is complete. When armed with SCP, your Hail Mary pass can be caught only by your intended receiver—snap! In today's networks, however, the more robust SFTP is used more commonly than SCP.

Lightweight Directory Access Protocol (LDAP)

If you're the system administrator of any decent-sized network, odds are you've got a type of directory in place that keeps track of all your network resources, such as devices and users. But how do you access those directories? Through the Lightweight Directory Access Protocol (LDAP), that's how. This protocol standardizes how you access directories, and its first and second inceptions are described in RFCs 1487 and 1777, respectively. There were a few glitches in those two earlier versions, so a third version—the one most commonly used today—was created to address those issues, and is described in RFC 3377.

Internet Group Management Protocol (IGMP)

Internet Group Management Protocol (IGMP) is the TCP/IP protocol used for managing IP multicast sessions. It accomplishes this by sending out unique IGMP messages over the network to reveal the multicast-group landscape and to find out which hosts belong to which multicast group. The host machines in an IP network also use IGMP messages to become members of a group and to quit the group, too. IGMP messages come in seriously handy for tracking group memberships as well as active multicast streams.

Line Printer Remote (LPR)

When printing in an unblended, genuine TCP/IP environment, a combination of Line Printer (LPR) and the Line Printer Daemon (LPD) is typically what's used to get the job done. LPD, installed on all printing devices, handles both printers and print jobs. LPR acts on the client, or sending machine, and is used to send the data from a host machine to the network's print resource so you end up with actual printed output.

Domain Name Service (DNS)

Domain Name Service (DNS) resolves hostnames—specifically, Internet names, such as `www.lammle.com`, to their corresponding IP addresses.

You don't have to use DNS; you can just type in the IP address of any device you want to communicate with. An IP address identifies hosts on a network and the Internet as well. However, DNS was designed to make our lives easier. Think about this: What would happen if you wanted to move your web page to a different service provider? The IP address would change, and no one would know what the new one was. DNS allows you to use a domain name to specify an IP address. You can change the IP address as often as you want, and no one will know the difference.

DNS is used to resolve a *fully qualified domain name (FQDN)*—for example, `www.lammle.com` or `todd.lammle.com`. An FQDN is a hierarchy that can logically locate a system based on its domain identifier.

If you want to resolve the name *todd*, you must either type in the FQDN of `todd.lammle.com` or have a device, such as a PC or router, add the suffix for you. For example, on a Cisco router, you can use the command `ip domain-name lammle.com` to append each request with the `lammle.com` domain. If you don't do that, you'll have to type in the FQDN to get DNS to resolve the name.

TIP An important thing to remember about DNS is that if you can ping a device with an IP address but can't use its FQDN, then you might have some type of DNS configuration failure.

Dynamic Host Configuration Protocol (DHCP)/Bootstrap Protocol (BootP)

Dynamic Host Configuration Protocol (DHCP) assigns IP addresses to hosts with information provided by a server. It allows easier administration and works well in small to even very large network environments. Many types of hardware can be used as a DHCP server, including routers.

DHCP differs from Bootstrap Protocol (BootP) in that BootP assigns an IP address to a host but the host's hardware address must be entered manually in a BootP table. You can think of DHCP as a dynamic BootP. But remember that BootP is also used to send an operating system that a host can boot from. DHCP can't do that.

But there is a lot of information a DHCP server can provide to a host when the host is requesting an IP address from the DHCP server. Here's a partial list of the information a DHCP server can provide:

- IP address
- Subnet mask
- Domain name
- Default gateway (routers)
- DNS
- Windows Internet Naming Service (WINS) information

A DHCP server can give even more information than this, but the items in the list are the most common.

A client that sends out a DHCP DISCOVER message in order to receive an IP address sends out a broadcast at both Layer 2 and Layer 3. The Layer 2 broadcast is all *F*s in hex, which looks like this: FF:FF:FF:FF:FF:FF. The Layer 3 broadcast is 255.255.255.255, which means all networks and all hosts. DHCP is connectionless, which means it uses User Datagram Protocol (UDP) at the Transport layer, also known as the Host-to-Host layer, which we'll talk about next.

In case you don't believe me, here's an example of output from my trusty analyzer:

```
Ethernet II, Src: 192.168.0.3 (00:0b:db:99:d3:5e), Dst: Broadcast➥
 (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255➥
 (255.255.255.255)
```

The Data Link and Network layers are both sending out "all hands" broadcasts saying, "Help—I don't know my IP address!"

Figure 6.3 shows the process of a client/server relationship using a DHCP connection.

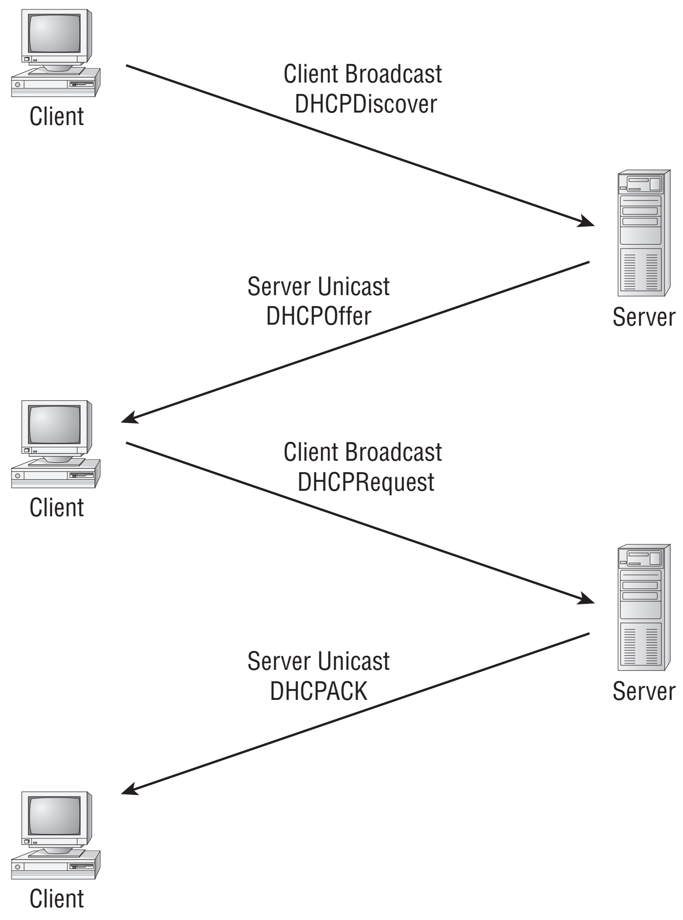
The following is the four-step process a client takes to receive an IP address from a DHCP server:

1. The DHCP client broadcasts a DHCP Discover message looking for a DHCP server (Port 67).
2. The DHCP server that received the DHCP Discover message sends a unicast DHCP Offer message back to the host
3. The client then broadcasts to the server a DHCP Request message asking for the offered IP address and possibly other information.
4. The server finalizes the exchange with a unicast DHCP Acknowledgment message.

What happens if you have a few hosts connected together with a switch or hub and you don't have a DHCP server? You can add IP information by hand (this is called *static IP addressing*; or, Windows provides what is called Automatic Private IP Addressing [APIPA], a feature of later Windows operating systems). With APIPA, clients can automatically self-configure an IP address and subnet mask (basic IP information that hosts use to communicate, which is covered in detail in Chapter 7, "IP Addressing," and Chapter 8, "IP Subnetting, Troubleshooting IP, and Introduction to NAT") when a DHCP server isn't available. The IP address range for APIPA is 169.254.0.1 through 169.254.255.254. The client also configures itself with a default class B subnet mask of 255.255.0.0. If you have a DHCP server and your host is using this IP address, this means your DHCP client on your host is not working, or the server is down or can't be reached because of a network issue.

Now, let's take a look at the Transport layer, or what the DoD calls the Host-to-Host layer.

FIGURE 6.3 DHCP client four-step process



The Host-to-Host Layer Protocols

The main purpose of the Host-to-Host layer is to shield the upper-layer applications from the complexities of the network. This layer says to the upper layer, "Just give me your data stream, with any instructions, and I'll begin the process of getting your information ready to send."

The following sections describe the two protocols at this layer:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

In addition, we'll look at some of the key host-to-host protocol concepts, as well as the port numbers.

Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) takes large blocks of information from an application and breaks them into segments. It numbers and sequences each segment so that the destination's TCP process can put the segments back into the order the application intended. After these segments are sent, TCP (on the transmitting host) waits for an acknowledgment from the receiving end's TCP process, retransmitting those segments that aren't acknowledged.

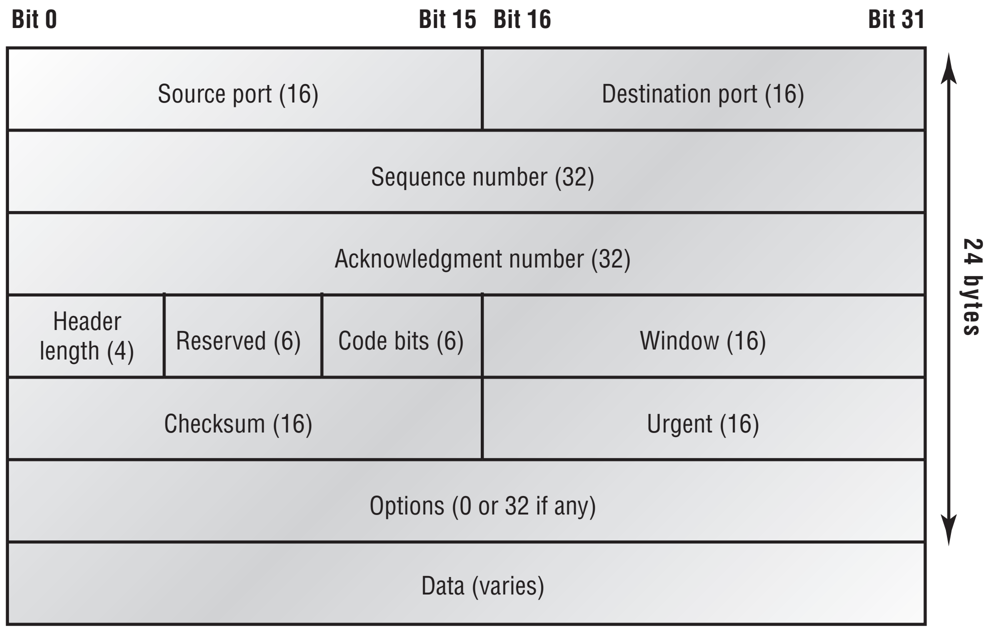
Remember that in reliable transport operation, a device that wants to transmit sets up a connection-oriented communication with a remote device by creating a session. The transmitting device first establishes a connection-oriented session with its peer system, which is called a *call setup* or a *three-way handshake*. Data is then transferred; and when the transfer is complete, a call termination takes place to tear down the virtual circuit.

TCP is a full-duplex, connection-oriented, reliable, and accurate protocol, but establishing all these terms and conditions, in addition to error checking, is no small task. TCP is very complicated and, not surprisingly, costly in terms of network overhead. And because today's networks are much more reliable than those of yore, this added reliability is often unnecessary.

Because the upper layers just send a data stream to the protocols in the Transport layers, I'll demonstrate how TCP segments a data stream and prepares it for the Internet layer. When the Internet layer receives the data stream, it routes the segments as packets through an internetwork. The segments are handed to the receiving host's Host-to-Host layer protocol, which rebuilds the data stream to hand to the upper-layer protocols.

Figure 6.4 shows the TCP segment format. The figure shows the different fields within the TCP header.

FIGURE 6.4 TCP segment format



The TCP header is 20 bytes long, or up to 24 bytes with options.

NOTE For more detailed information regarding the TCP header, which is beyond the scope of the CompTIA Network+ exam objectives, please see my *CCNA: Cisco Certified Network Associate Study Guide, 6th Edition* (Sybex, 2007).

User Datagram Protocol (UDP)

If you were to compare *User Datagram Protocol (UDP)* with TCP, the former is basically the scaled-down economy model that's sometimes referred to as a *thin protocol*. Like a thin person on a park bench, a thin protocol doesn't take up a lot of room—or in this case, much bandwidth on a network.

UDP doesn't offer all the bells and whistles of TCP either, but it does do a fabulous job of transporting information that doesn't require reliable delivery—and it does so using far fewer network resources.

There are some situations in which it would definitely be wise for developers to opt for UDP rather than TCP. Remember the watchdog SNMP up there at the Process/Application layer? SNMP monitors the network, sending intermittent messages and a fairly steady flow of status updates and alerts, especially when running on a large network. The cost in overhead to establish, maintain, and close a TCP connection for each one of those little messages would reduce what would be an otherwise healthy, efficient network to a dammed-up bog in no time!

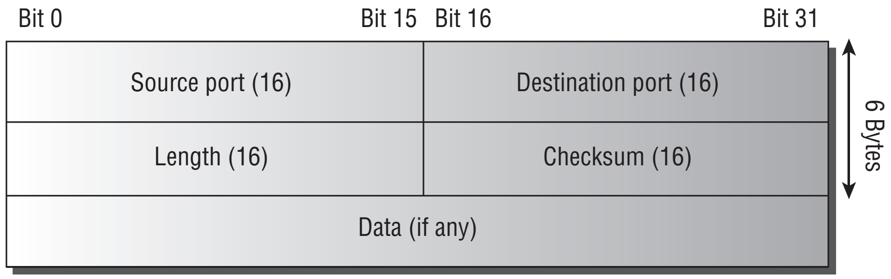
Another circumstance calling for UDP over TCP is when reliability is already handled at the Process/Application layer. NFS handles its own reliability issues, making the use of TCP both impractical and redundant. But ultimately, it's up to the application developer to decide whether to use UDP or TCP, not the user who wants to transfer data faster.

UDP does *not* sequence the segments and doesn't care in which order the segments arrive at the destination. But after that, UDP sends the segments off and forgets about them. It doesn't follow through, check up on them, or even allow for an acknowledgment of safe arrival—complete abandonment. Because of this, it's referred to as an *unreliable* protocol. This doesn't mean that UDP is ineffective, only that it doesn't handle issues of reliability. Because UDP assumes that the application will use its own reliability method, it doesn't use any. This gives an application developer a choice when running the IP stack: TCP for reliability or UDP for faster transfers.

Further, UDP doesn't create a virtual circuit, nor does it contact the destination before delivering information to it. Because of this, it's also considered a *connectionless* protocol.

Figure 6.5 clearly illustrates UDP's markedly low overhead as compared to TCP's hungry usage. Look at the figure carefully—can you see that UDP doesn't use windowing or provide for acknowledgments in the UDP header?

FIGURE 6.5 UDP segment



NOTE For more detailed information regarding the UDP header, which is beyond the scope of the CompTIA Network+ exam objectives, please see my *CCNA: Cisco Certified Network Associate Study Guide, 6th Edition* (Sybex, 2007).

Key Concepts of Host-to-Host Protocols

Now that you've seen both a connection-oriented (TCP) and connectionless (UDP) protocol in action, it would be good to summarize the two here. Table 2.1 highlights some of the key concepts that you should keep in mind regarding these two protocols. You should memorize this table.

TABLE 6.1 Key Features of TCP and UDP

TCP	UDP
Sequenced	Unsequenced
Reliable	Unreliable
Connection-oriented	Connectionless
Virtual circuit	No virtual circuit
High overhead	Low overhead
Acknowledgments	No acknowledgment
Windowing flow control	No windowing or flow control

A telephone analogy could really help you understand how TCP works. Most of us know that before you speak to someone on a phone, you must first establish a connection with that other person—wherever they are. This is like a virtual circuit with the TCP protocol. If you were giving someone important information during your conversation, you might say, "You know?" or ask, "Did you get that?" Saying something like this is a lot like a TCP acknowledgment—it's designed to get your verification. From time to time (especially on cell phones), people also ask, "Are you still there?" They end their conversations with a "Goodbye" of some kind, putting closure on the phone call. TCP also performs these types of functions.

Alternatively, using UDP is like sending a postcard. To do that, you don't need to contact the other party first. You simply write your message, address the postcard, and mail it. This is analogous to UDP's connectionless orientation. Because the message on the postcard is probably not a matter of life or death, you don't need an acknowledgment of its receipt. Similarly, UDP doesn't involve acknowledgments.

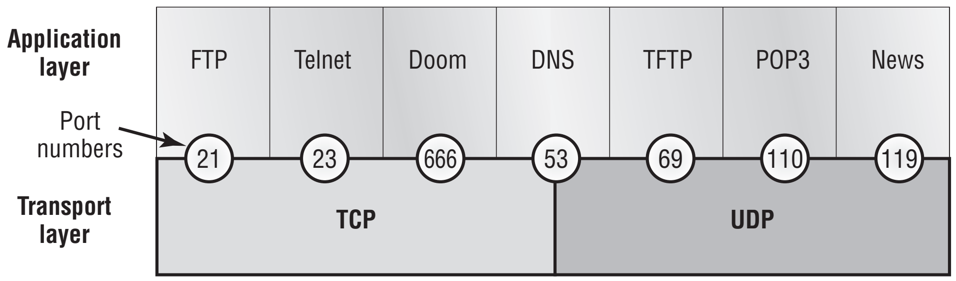
Port Numbers

TCP and UDP must use *port numbers* to communicate with the upper layers because they're what keep track of different simultaneous conversations originated by or accepted by the local host. Originating-source port numbers are dynamically assigned by the source host and will usually have a value of 1024 or higher. Ports 1023 and below are defined in RFC 3232, which discusses what are called *well-known port numbers*.

Virtual circuits that don't use an application with a well-known port number are assigned port numbers randomly from a specific range instead. These port numbers identify the source and destination application or process in the TCP segment.

Figure 6.6 illustrates how both TCP and UDP use port numbers.

FIGURE 6.6 Port numbers for TCP and UDP



Numbers below 1024 are considered well-known port numbers and are defined in RFC 3232. Numbers 1024 and above are used by the upper layers to set up sessions with other hosts and by TCP to use as source and destination identifiers in the TCP segment.

Table 6.2 gives you a list of the typical applications used in the TCP/IP suite, their well-known port numbers, and the Transport layer protocols used by each application or process. It's important that you study and memorize this table for the CompTIA Network+ objectives.

TABLE 6.2 Key Protocols That Use TCP and UDP

TCP	UDP
Telnet 23	SNMP 161
SMTP 25	TFTP 69
HTTP 80	DNS 53
FTP 20, 21	BOOTPS/DHCP 67
DNS 53	
HTTPS 443	
SSH 22	

TABLE 6.2 Key Protocols That Use TCP and UDP *(continued)*

TCP	UDP
POP3 110	
NTP 123	
IMAP4 143	

Notice that DNS uses both TCP and UDP. Whether it opts for one or the other depends on what it's trying to do. Even though it's not the only application that can use both protocols, it's certainly one that you should remember in your studies.

The Internet Layer Protocols

In the DoD model, there are two main reasons for the Internet layer's existence: routing and providing a single network interface to the upper layers.

None of the other upper- or lower-layer protocols have any functions relating to routing—that complex and important task belongs entirely to the Internet layer. The Internet layer's second duty is to provide a single network interface to the upper-layer protocols. Without this layer, application programmers would need to write what are called *hooks* into every one of their applications for each different Network Access protocol. This would not only be a pain in the neck, but it would also lead to different versions of each application—one for Ethernet, another one for Token Ring, and so on. To prevent this, IP provides one single network interface for the upper-layer protocols. That accomplished, it's then the job of IP and the various Network Access protocols to get along and work together.

All network roads don't lead to Rome—they lead to IP. And all the other protocols at this layer, as well as all those at the upper layers, use it. Never forget that. All paths through the DoD model go through IP. The following sections describe the protocols at the Internet layer:

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Reverse Address Resolution Protocol (RARP)
- Proxy ARP

Internet Protocol (IP)

Internet Protocol (IP) essentially is the Internet layer. The other protocols found here merely exist to support it. IP holds the big picture and could be said to "see all," in that it's aware of all the interconnected networks. It can do this because all the machines on the network have a software, or logical, address called an IP address, which I'll cover more thoroughly later in this chapter.

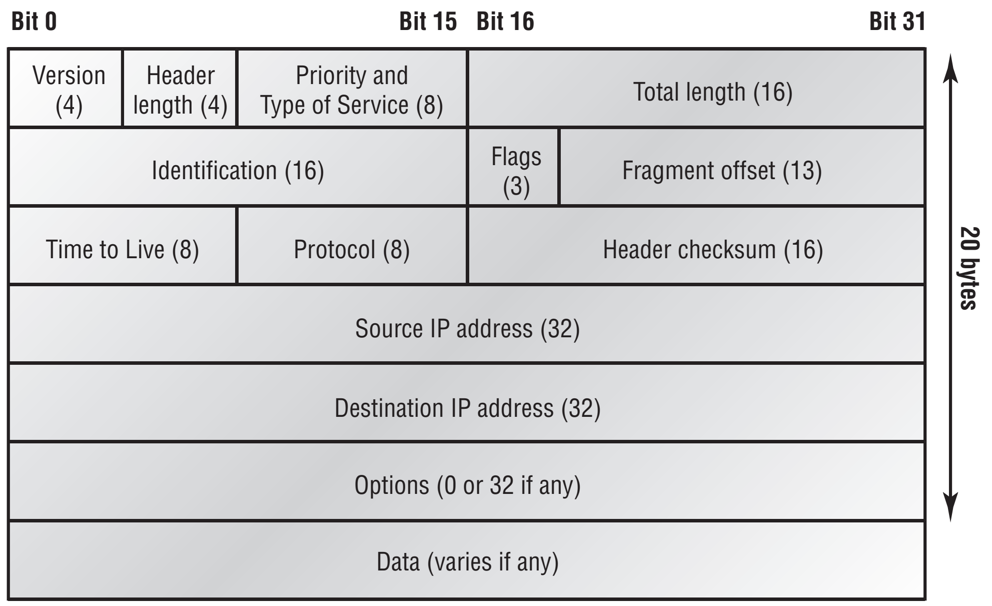
IP looks at each packet's destination address. Then, using a routing table, it decides where a packet is to be sent next, choosing the best path. The protocols of the Network Access layer at the bottom of the DoD model don't possess IP's enlightened scope of the entire network; they deal only with physical links (local networks).

Identifying devices on networks requires answering these two questions: Which network is it on? And what is its ID on that network? The first answer is the *software address*, or *logical address* (the correct street). The second answer is the *hardware address* (the correct mailbox). All hosts on a network have a logical ID called an IP address. This is the software, or logical, address and contains valuable encoded information, greatly simplifying the complex task of routing. (IP is discussed in RFC 791.)

IP receives segments from the Host-to-Host layer and fragments them into packets if necessary. IP then reassembles packets back into segments on the receiving side. Each packet is assigned the IP address of the sender and of the recipient. Each router (Layer 3 device) that receives a packet makes routing decisions based on the packet's destination IP address.

Figure 6.7 shows an IP header. This will give you an idea of what the IP protocol has to go through every time user data is sent from the upper layers and is to be sent to a remote network.

FIGURE 6.7 IP header



NOTE

For more detailed information regarding the IP header, which is beyond the CompTIA Network+ exam objectives, please see my *CCNA: Cisco Certified Network Associate Study Guide, 6th Edition* (Sybex, 2007).

Internet Control Message Protocol (ICMP)

Internet Control Message Protocol (ICMP) works at the Network layer and is used by IP for many different services. ICMP is a management protocol and messaging service provider for IP. Its messages are carried as IP packets.

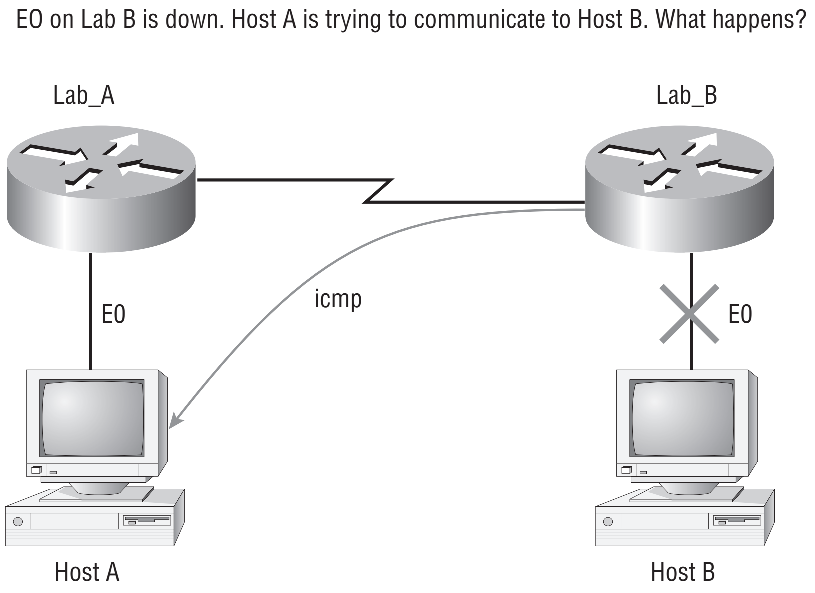
ICMP packets have the following characteristics:

- They can provide hosts with information about network problems.
- They are encapsulated within IP datagrams.

The following are some common events and messages that ICMP relates to:

Destination Unreachable If a router can't send an IP datagram any further, it uses ICMP to send a message back to the sender, advising it of the situation. For example, take a look at Figure 6.8, which shows that interface E0 of the Lab_B router is down.

FIGURE 6.8 ICMP error message is sent to the sending host from the remote router.


E0 on Lab B is down. Host A is trying to communicate to Host B. What happens?
Lab_A
Lab_B
E0
icmp
E0
Host A
Host B

When Host A sends a packet destined for Host B, the Lab_B router will send an ICMP Destination Unreachable message back to the sending device (Host A, in this example).

Buffer Full If a router's memory buffer for receiving incoming datagrams is full, it will use ICMP to send out this message until the congestion abates.

Hops Each IP datagram is allotted a certain number of routers, called *hops*, to pass through. If it reaches its limit of hops before arriving at its destination, the last router to receive that datagram deletes it. The executioner router then uses ICMP to send an obituary message, informing the sending machine of the demise of its datagram.

Ping Ping uses ICMP echo request and reply messages to check the physical and logical connectivity of machines on an internetwork.

Traceroute Traceroute uses IP packet Time-to-Live time-outs to discover the path a packet takes as it traverses an internetwork.

NOTE Both Ping and Traceroute (also just called Trace; Microsoft Windows uses tracert) allow you to verify address configurations in your internetwork.

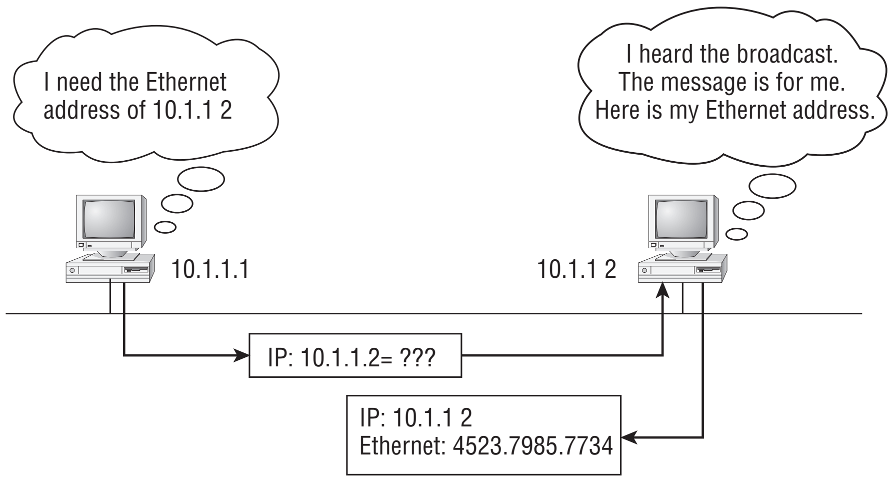
Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) finds the hardware address of a host from a known IP address. Here's how it works: When IP has a datagram to send, it must inform a Network Access protocol, such as Ethernet or Token Ring, of the destination's hardware address on the local network. (It has already been informed by upper-layer protocols of the destination's IP address.) If IP doesn't find the destination host's hardware address in the ARP cache, it uses ARP to find this information.

As IP's detective, ARP interrogates the local network by sending out a broadcast asking the machine with the specified IP address to reply with its hardware address. So basically, ARP translates the software (IP) address into a hardware address—for example, the destination machine's Ethernet address. Figure 6.9 shows how an ARP broadcast looks to a local network.

NOTE ARP resolves IP addresses to Ethernet (MAC) addresses.

FIGURE 6.9 Local ARP broadcast


I need the Ethernet address of 10.1.1 2
I heard the broadcast. The message is for me. Here is my Ethernet address.
10.1.1.1
10.1.1 2
IP: 10.1.1.2= ???
IP: 10.1.1 2
Ethernet: 4523.7985.7734

The following trace shows an ARP broadcast—notice that the destination hardware address is unknown and is all 0s in the ARP header. In the Ethernet header, a destination of all *F*s in hex (all 1s in binary), a hardware-address broadcast, is used to make sure all devices on the local link receive the ARP request.

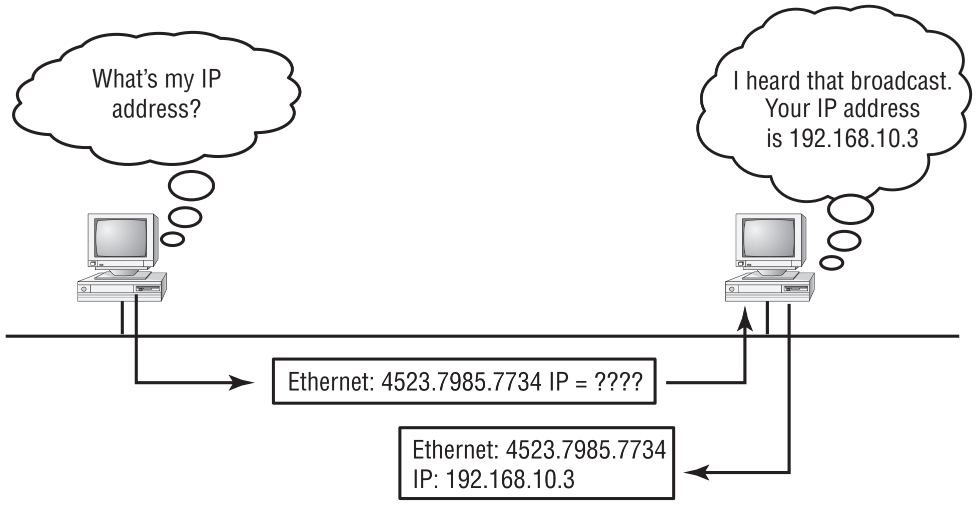
```
 Flags:         0x00
 Status:        0x00
 Packet Length: 64
 Timestamp:     09:17:29.574000 12/06/03
Ethernet Header
 Destination:   FF:FF:FF:FF:FF:FF Ethernet Broadcast
 Source:        00:A0:24:48:60:A5
 Protocol Type: 0x0806 IP ARP
ARP - Address Resolution Protocol
 Hardware:                1 Ethernet (10Mb)
 Protocol:                0x0800 IP
 Hardware Address Length: 6
 Protocol Address Length: 4
 Operation:               1 ARP Request
 Sender Hardware Address: 00:A0:24:48:60:A5
 Sender Internet Address: 172.16.10.3
 Target Hardware Address: 00:00:00:00:00:00 (ignored)
 Target Internet Address: 172.16.10.10
Extra bytes (Padding):
 ................ 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
  0A 0A 0A 0A 0A
Frame Check Sequence: 0x00000000
```

Reverse Address Resolution Protocol (RARP)

When an IP machine happens to be a diskless machine, it has no way of initially knowing its IP address. But it does know its MAC address. *Reverse Address Resolution Protocol (RARP)* discovers the identity of the IP address for diskless machines by sending out a packet that includes its MAC address and a request for the IP address assigned to that MAC address. A designated machine, called a *RARP server*, responds with the answer, and the identity crisis is over. RARP uses the information it does know about the machine's MAC address to learn its IP address and complete the machine's ID portrait.

Figure 6.10 shows a diskless workstation asking for its IP address with a RARP broadcast.

FIGURE 6.10 RARP broadcast example


What's my IP address?
I heard that broadcast. Your IP address is 192.168.10.3
Ethernet: 4523.7985.7734 IP = ????
Ethernet: 4523.7985.7734
IP: 192.168.10.3

Proxy Address Resolution Protocol (Proxy ARP)

On a network, your hosts can't have more than one default gateway configured. Think about this: What if the default gateway (router) happens to go down? The host won't just start sending to another router automatically—you've got to reconfigure that host. But Proxy ARP can actually help machines on a subnet reach remote subnets without configuring routing or even a default gateway.

One advantage of using Proxy ARP is that it can be added to a single router on a network without disturbing the routing tables of all the other routers that live there too. But there's a serious downside to using Proxy ARP. Using Proxy ARP will definitely increase the amount of traffic on your network segment, and hosts will have a larger ARP table than usual in order to handle all the IP-to-MAC-address mappings. And Proxy ARP is configured on all Cisco routers by default—you should disable it if you don't think you're going to use it.

One last thought on Proxy ARP: Proxy ARP isn't really a separate protocol. It's a service run by routers on behalf of other devices (usually PCs) that are separated from their query to another device by a router, although they think they share the subnet with the remote device.

Data Encapsulation

I started to discuss data encapsulation in Chapter 2, but I could only provide an overview at that point in the book because you needed to have a firm understanding of how ports work in a virtual circuit. With the last five chapters of foundational material under your belt, you're ready to get more into the details of encapsulations.

When a host transmits data across a network to another device, the data goes through *encapsulation*: It's wrapped with protocol information at each layer of the OSI model. Each layer communicates only with its peer layer on the receiving device.

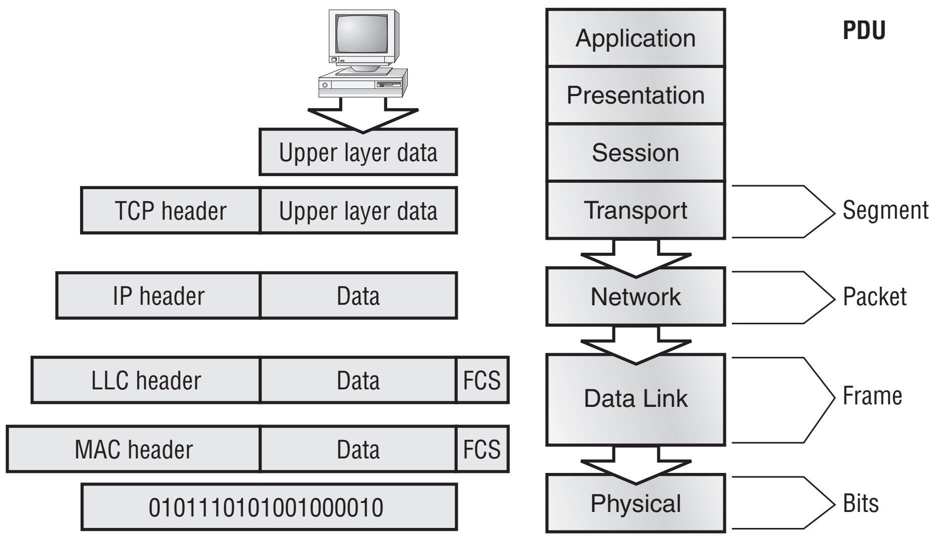
To communicate and exchange information, each layer uses *Protocol Data Units (PDUs)*. These hold the control information attached to the data at each layer of the model. They're usually attached to the header in front of the data field but can also be in the trailer, or end, of it.

Each PDU attaches to the data by encapsulating it at each layer of the OSI model, and each has a specific name depending on the information provided in each header. This PDU information is read only by the peer layer on the receiving device. After it's read, it's stripped off, and the data is then handed to the next layer up.

Figure 6.11 shows the PDUs and how they attach control information to each layer. This figure demonstrates how the upper-layer user data is converted for transmission on the network. The data stream is then handed down to the Transport layer, which sets up a virtual circuit to the receiving device by sending over a synch packet. Next, the data stream is broken up into smaller pieces, and a Transport layer header (a PDU) is created and attached to the header of the data field; now the piece of data is called a *segment*. Each segment is sequenced so the data stream can be put back together on the receiving side exactly as it was transmitted.

Each segment is then handed to the Network layer for network addressing and routing through the internetwork. Logical addressing (for example, IP) is used to get each segment to the correct network. The Network layer protocol adds a control header to the segment handed down from the Transport layer, and what we have now is called a *packet* or *datagram*. Remember that the Transport and Network layers work together to rebuild a data stream on a receiving host, but it's not part of their work to place their PDUs on a local network segment—which is the only way to get the information to a router or host.

FIGURE 6.11 Data encapsulation



It's the Data Link layer that's responsible for taking packets from the Network layer and placing them on the network medium (cable or wireless). The Data Link layer encapsulates each packet in a *frame*, and the frame's header carries the hardware address of the source and destination hosts. If the destination device is on a remote network, then the frame is sent to a router to be routed through an internetwork. Once it gets to the destination network, a new frame is used to get the packet to the destination host.

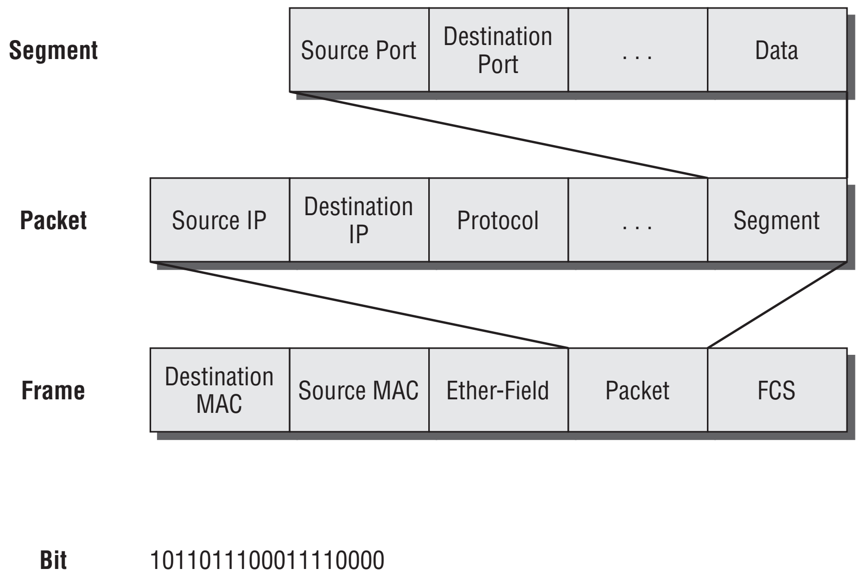
To put this frame on the network, it must first be put into a digital signal. Because a frame is really a logical group of 1s and 0s, the Physical layer is responsible for encoding these digits into a digital signal, which is read by devices on the same local network. The receiving devices will synchronize on the digital signal and extract (decode) the 1s and 0s from the digital signal. At this point, the devices build the frames, run a cyclic redundancy check (CRC), and then check their answer against the answer in the frame's Frame Check Sequence (FCS) field. If it matches, the packet is pulled from the frame and what's left of the frame is discarded. This process is called *de-encapsulation*. The packet is handed to the Network layer, where the address is checked. If the address matches, the segment is pulled from the packet and what's left of the packet is discarded. The segment is processed at the Transport layer, which rebuilds the data stream and acknowledges to the transmitting station that it received each piece. It then happily hands the data stream to the upper-layer application.

At a transmitting device, the data-encapsulation method works like this:

1. User information is converted to data for transmission on the network.
2. Data is converted to segments, and a reliable connection is set up between the transmitting and receiving hosts.
3. Segments are converted to packets or datagrams, and a logical address is placed in the header so each packet can be routed through an internetwork.
4. Packets or datagrams are converted to frames for transmission on the local network. Hardware (Ethernet) addresses are used to uniquely identify hosts on a local network segment.
5. Frames are converted to bits, and a digital encoding and clocking scheme is used.

To explain this in more detail using the layer addressing, I'll use Figure 6.12.

FIGURE 6.12 PDU and layer addressing



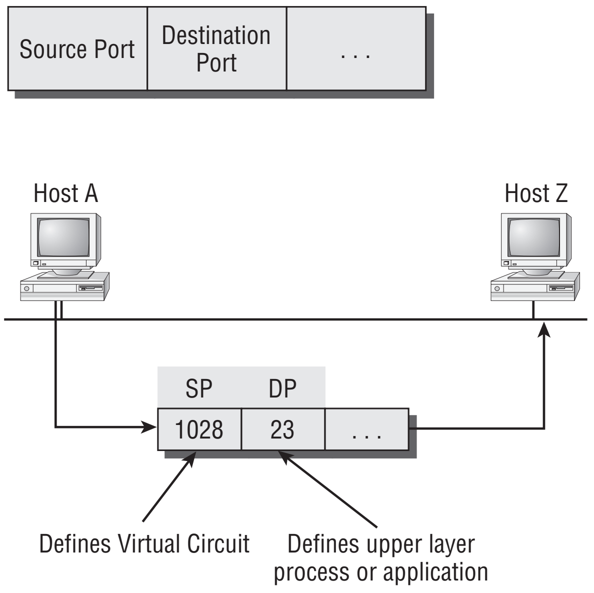
Remember that a data stream is handed down from the upper layer to the Transport layer. As technicians, we really don't care who the data stream comes from because that's a programmer's problem. Our job is to rebuild the data stream reliably and hand it to the upper layers on the receiving device.

Before we go further in our discussion of Figure 6.12, let's review port numbers and make sure you understand them. The Transport layer uses port numbers to define both the virtual circuit and the upper-layer process, as you can see from Figure 6.13.

The Transport layer takes the data stream, makes segments out of it, and establishes a reliable session by creating a virtual circuit. It then sequences (numbers) each segment and uses acknowledgments and flow control. If you're using TCP, the virtual circuit is defined by the source port number. Remember, the host just makes this up starting at port number 1024 (0 through 1023 are reserved for well-known port numbers). The destination port number defines the upper-layer process (application) that the data stream is handed to when the data stream is reliably rebuilt on the receiving host.

Now that you understand port numbers and how they're used at the Transport layer, let's go back to Figure 6.12. Once the Transport layer header information is added to the piece of data, it becomes a segment and is handed down to the Network layer along with the destination IP address. (The destination IP address was handed down from the upper layers to the Transport layer with the data stream, and it was discovered through a name resolution method at the upper layers—probably DNS.)

FIGURE 6.13 Port numbers at the Transport layer



The Network layer adds a header, and adds the logical addressing (IP addresses), to the front of each segment. Once the header is added to the segment, the PDU is called a *packet*. The packet has a protocol field that describes where the segment came from (either UDP or TCP) so it can hand the segment to the correct protocol at the Transport layer when it reaches the receiving host.

The Network layer is responsible for finding the destination hardware address that dictates where the packet should be sent on the local network. It does this by using ARP. IP at the Network layer looks at the destination IP address and compares that address to its own source IP address and subnet mask. If it turns out to be a local network request, the hardware address of the local host is requested via an ARP request. If the packet is destined for a remote host, IP will look for the IP address of the default gateway (router) instead.

The packet, along with the destination hardware address of either the local host or default gateway, is then handed down to the Data Link layer. The Data Link layer will add a header to the front of the packet, and the piece of data then becomes a *frame*. (We call it a frame because both a header and a trailer are added to the packet, which makes the data resemble bookends or a frame, if you will.) This is shown in Figure 6.12. The frame uses an Ether-Type field to describe which protocol the packet came from at the Network layer. Now a CRC is run on the frame, and the answer to the CRC is placed in the FCS field found in the trailer of the frame.

The frame is now ready to be handed down, one bit at a time, to the Physical layer, which will use bit-timing rules to encode the data in a digital signal. Every device on the network segment will synchronize with the clock, extract the 1s and 0s from the digital signal, and build a frame. After the frame is rebuilt, a CRC is run to make sure the frame is okay. If everything turns out to be good, the hosts will check the destination address to see if the frame is for them.

If all this is making your eyes cross and your brain freeze, don't freak—things will become much clearer as we go through the book—really! Soon, I'll be going over exactly how data is encapsulated and routed through an internetwork in even more detail, in an easy to understand, step-by-step manner, in Chapter 9, "Introduction to Routing."

Summary

Protocols, protocols everywhere—so many different reasons for them, and so many jobs they do for us! And sometimes they even work in conjunction with each other. This can seem like way too much information, but no worries—as you become familiar with the various layers and their functions, I promise it will soon become clear that this hierarchical structure is a seriously tight, robust networking foundation.

Similarly, as you understand The TCP/IP big picture, the reason why all those protocols exist and are necessary will also become much easier to understand. They're really like a team that works jointly, from layer to layer, to make our TCP/IP networks the wonderful, great tools they are.

Exam Essentials

Remember the Process/Application layer protocols. Telnet is a terminal-emulation program that allows you to log into a remote host and run programs. File Transfer Protocol (FTP) is a connection-oriented service that allows you to transfer files. Trivial FTP (TFTP) is a connectionless file transfer program. Simple Mail Transfer Protocol (SMTP) is a sendmail program.

Remember the Host-to-Host layer protocols. Transmission Control Protocol (TCP) is a connection-oriented protocol that provides reliable network service by using acknowledgments and flow control. User Datagram Protocol (UDP) is a connectionless protocol that provides low overhead and is considered unreliable.

Remember the Internet layer protocols. Internet Protocol (IP) is a connectionless protocol that provides network address and routing through an internetwork. Address Resolution Protocol (ARP) finds a hardware address from a known IP address. Reverse ARP (RARP) finds an IP address from a known hardware address. Internet Control Message Protocol (ICMP) provides diagnostics and destination-unreachable messages.

Remember the difference between connection-oriented and connectionless network services. Connection-oriented services use acknowledgments and flow control to create a reliable session. More overhead is used than in a connectionless network service. Connectionless services are used to send data with no acknowledgments or flow control. This is considered unreliable.

Written Lab

1. What might be the problem if a DHCP client suddenly finds itself in a different IP subnet from the one it should be in?
2. Name the protocol that uses both TCP ports 20 and 21.
3. What two well-known port numbers does a DNS server use?
4. Which protocol dynamically reports errors to source hosts by using IP directly to build packets?
5. What could cause a server that you can ping not to provide the particular TCP/IP service, such as FTP, HTTP, and so on, that you expect it to offer?
6. What might cause your email to stop functioning properly when you change Internet service providers?
7. Which UNIX command is used for terminal emulation in the same way Telnet is used?
8. What protocol is at the heart of the `ping` and `tracert` commands in a Windows operating system?
9. Which destination Transport-layer protocol and port number does a TFTP client use to transfer files over the network?
10. What well-known port numbers do SMTP, POP3, and IMAP4 servers use?

(The answers to the Written Lab can be found following the answers to the Review Questions for this chapter.)

Review Questions

1. The OSI has seven layers, which layer does SMTP work at?
   - **A.** Network
   - **B.** Transport
   - **C.** Session
   - **D.** Application
2. You need to have secure communications using HTTPS. What port number is used by default?
   - **A.** 69
   - **B.** 23
   - **C.** 21
   - **D.** 443
3. You want to implement a mechanism that automates the IP configuration, including IP address, subnet mask, default gateway, and DNS information. Which protocol will you use to accomplish this?
   - **A.** SMTP
   - **B.** SNMP
   - **C.** DHCP
   - **D.** ARP
4. What protocol is used to find the hardware address of a local device?
   - **A.** RARP
   - **B.** ARP
   - **C.** IP
   - **D.** ICMP
   - **E.** BootP
5. You need to login to a Unix server across a network that is not secure. Which of the following protocols will allow you to remotely administrator this server securely?
   - **A.** Telnet
   - **B.** SSH
   - **C.** SFTP
   - **D.** HTTP

6. If you can ping by IP address but not by hostname, or FQDN, which of the following port numbers is related to the server process that is involved?

A. 21
B. 23
C. 53
D. 69
E. 80

7. Which of the following describe the DHCP Discover message? (Choose two.)

A. It uses FF:FF:FF:FF:FF:FF as a Layer 2 broadcast.
B. It uses UDP as the Transport layer protocol.
C. It uses TCP as the Transport layer protocol.
D. It does not use a Layer 2 destination address.

8. What layer 4 protocol is used for a Telnet connection, and what is the default port number?

A. IP, 6
B. TCP, 21
C. UDP, 23
D. ICMP, 21
E. TCP, 23

9. Which statements are true regarding ICMP packets? (Choose two.)

A. They acknowledge receipt of a TCP segment.
B. They guarantee datagram delivery.
C. They can provide hosts with information about network problems.
D. They are encapsulated within IP datagrams.
E. They are encapsulated within UDP datagrams.

10. Which of the following services use TCP? (Choose three.)

A. DHCP
B. SMTP
C. SNMP
D. FTP
E. HTTP
F. TFTP

11. Which of the following services use UDP? (Choose three.)

A. DHCP

B. SMTP

C. SNMP

D. FTP

E. HTTP

F. TFTP

12. Which of the following are TCP/IP protocols used at the Application layer of the OSI model? (Choose three.)

A. IP

B. TCP

C. Telnet

D. FTP

E. TFTP

13. Which of the following protocols is used by e-mail servers to exchange messages with one another?

A. POP3

B. IMAP

C. SMTP

D. HTTP

14. If you use either Telnet or FTP, which is the highest layer you are using to transmit data?

A. Application

B. Presentation

C. Session

D. Transport

15. Which of the following protocols can use TCP and UDP, permits authentication and secure polling of network devices, and allows for automated alerts and reports on network devices?

A. DNS

B. SNMP

C. SMTP

D. TCP

16. You need to transfer files between two hosts. Which two protocol can you use?

A. SNMP

B. SCP

C. RIP

D. NTP

E. FTP

17. What layer in the IP stack is equivalent to the Transport layer of the OSI model?

A. Application

B. Host-to-Host

C. Internet

D. Network Access

18. You need to make sure that your network devices have a consistent time across all devices. What protocol do you need to run on your network?

A. FTP

B. SCP

C. NTP

E. RTP

19. Which of the following allows a server to distinguish among different simultaneous requests from the same host?

A. They have different port numbers.

B. A NAT server changes the IP address for subsequent requests.

C. A server is unable to accept multiple simultaneous sessions from the same host. One session must end before another can begin.

D. The MAC address for each one is unique.

20. Which of the following protocols uses both TCP and UDP?

A. FTP

B. SMTP

C. Telnet

D. DNS

Answers to Review Questions

1. D. SMTP resides at the Application layer of the OSI/DoD model.
2. D. HTTPS, or Secure HTTP uses port 443 by default.
3. C. Dynamic Host Configuration Protocol (DHCP) is used to provide IP information to hosts on your network. DHCP can provide a lot of information, but the most common is IP address, subnet mask, default gateway, and DNS information.
4. B. Address Resolution Protocol (ARP) is used to find the hardware address from a known IP address.
5. B. Secure Shell (SSH) allows you to remotely administer router, switches and even servers securely.
6. C. The problem is with DNS, which uses both TCP and UDP port 53.
7. A, B. A client that sends out a DHCP Discover message in order to receive an IP address sends out a broadcast at both Layer 2 and Layer 3. The Layer 2 broadcast is all *F*s in hex, or FF:FF:FF:FF:FF:FF. The Layer 3 broadcast is 255.255.255.255, which means all networks and all hosts. DHCP is connectionless, which means it uses User Datagram Protocol (UDP) at the Transport layer, also called the Host-to-Host layer.
8. E. Telnet uses TCP at the Transport layer with a default port number of 23.
9. C, D. Internet Control Message Protocol (ICMP) is used to send error messages through the network, but ICMP does not work alone. Every segment or ICMP payload must be encapsulated within an IP datagram (or packet).
10. B, D, E. SMTP, FTP, and HTTP use TCP.
11. A, C, F. DHCP, SNMP, and TFTP use UDP. SMTP, FTP, and HTTP use TCP.
12. C, D, E. Telnet, File Transfer Protocol (FTP), and Trivial FTP (TFTP) are all Application layer protocols. IP is a Network layer protocol. Transmission Control Protocol (TCP) is a Transport layer protocol.
13. C. SMTP is used by a client to send mail to its server and by that server to send mail to another server. POP3 and IMAP are used by clients to retrieve their mail from the server that stores it until it is retrieved. HTTP is only used with web-based mail services.
14. A. Both FTP and Telnet use TCP at the Transport layer; however, they both are Application layer protocols, so the Application layer is the best answer for this question.
15. B. Simple Network Management Protocol, is typically implemented using version 3, which allows for a connection oriented service, authentication and secure polling of network devices, and allows for alerts and reports on network devices.

16. B, E. Secure Copy Protocol (SCP), and File Transfer Protocol (FTP), can be used to transfer files between two systems.

17. B. The four layers of the IP stack (also called the DoD model) are Application/Process, Host-to-Host, Internet, and Network Access. The Host-to-Host layer is equivalent to the Transport layer of the OSI model.

18. C. Network Time Protocol will ensure a consistent time across network devices on the network.

19. A. Through the use of port numbers, TCP and UDP can establish multiple sessions between the same two hosts without creating any confusion. The sessions can be between the same or different applications, such as multiple web-browsing sessions or a web-browsing session and an FTP session.

20. D. DNS uses TCP for zone exchanges between servers and UDP when a client is trying to resolve a hostname to an IP address.

Answers to Written Lab

1. The most likely problem is that a rogue DHCP server has been introduced into the network and is handing this device an incorrect lease.
2. FTP uses both TCP ports 20 and 21 for the data channel and the control channel, respectively.
3. A DNS server uses TCP port 53 for zone transfers and UDP port 53 for name resolutions.
4. ICMP uses IP directly to build error-reporting packets that are transmitted back to the originating source host when issues arise during the delivery of data packets. ICMP is also used during ping and some Traceroute operations.
5. Quite simply, the service might not be running currently on that server. Another possibility might be that a firewall between the client and the server has blocked the protocol in question from passing.
6. Most ISPs have their own mail servers. When you switch service, you might need to point your mail application to the servers provided by the new service provider.
7. The UNIX command `rlogin` functions similarly to Telnet.
8. ICMP is the protocol that the `ping` and `tracert` commands rely on. If you're having trouble getting pings and Traceroutes through a router, you might need to check if ICMP is being allowed thorough.
9. TFTP servers respond to UDP messages sent to port 69.
10. SMTP uses TCP port 25; POP3 uses TCP port 110; IMAP4 uses TCP port 143.