FTP is very robust. In Sec. 2, there was an in-depth discussion on port and sockets: how they are established and used. The FTP protocol actually uses two port assignments (and therefore two connections): 20 and 21. Remember that most connections between two network stations are made via one source port and one destination port. A network station wanting a connection to a remote network station must connection to two ports on the destination station in order for FTP to work.

Port 20 is used for the initial setup of the connection and is used as the control connection. No data passes over this circuit except for control information. Port 21 is used for user data (the file to be transferred) to pass over the connection.

Once the connection is established, the server process awaits a command from the client. To transfer a file from the server to the client, the user would type in: get <a name of a file>, which will be transmitted over to the remote network station. With this, a second connection is established between the server and client FTP process. It is known as the *data connection*. Now we have two connections, but only during the file transfer process. Once the file is transferred, the data connection port is closed.

All data connections are made with port number 20. This is the well-known (or static) FTP data port. From a user's standpoint, to establish a connection between itself and a remote station, the command is similar to TELNET; FTP <domain name or IP address>. A user could also type in FTP and wait for the FTP prompt. At the prompt, the user would use the open command to establish the connection. Table 6.11 lists the available commands in FTP.

There are a lot of commands listed but, in reality, only a few are used. They are:

TABLE 6.11 FTP Commands

!	cr	macdef	proxy	send
$	delete	mdelete	sendport	status
account	debug	mdir	put	struct
append	dir	mget	pwd	sunique
ascii	disconnect	mkdir	quit	tenex
bell	form	mls	quote	trace
binary	get	mode	recv	type
bye	glob	mput	remotehelp	user
case	hash	nmap	rename	verbose
cd	help	ntrans	reset	?
cdup	lcd	open	rmdir	
close	ls	prompt	runique	

open	open a connection to a remote resource
close	close a connection to remote resource
bye	end this FTP session
binary	indicate that the file transfer will be a file of binary type (i.e., executable file, lotus file, etc)
get	get a file from the remote resource; get <filename>; mget <multiple files, wildcards included>
put	put a file to the remote resource; put <filename>; mput <multiple files, wildcards included>
cd	change directory on the remote device; to change the directory on the local end use: lcd
dir	get a directory listing on the remote device; to get a directory listing on the local end use: ldir
hash	Display hash marks on the screen to indicate a file in being transferred

Data transfer. Refer to Fig. 6.33. If a user wanted to establish an FTP connection between 148.1.1.2 and an FTP server process on 148.1.1.19, the following sequence of events would take place on a DOS PC (for other operating systems, the prompt would change, but the commands are all the same in every FTP implementation):

```
C> FTP
```

The user would establish the connection via one of three possible ways:

1. FTP (entered with no arguments)
   FTP> OPEN <IP address or domain name>
2. FTP <IP address>
3. FTP <domain name>

The FTP client process would return information to the screen, showing a connection had been made, and then ask for a user name and password. Once the authentication has been approved, the FTP prompt would return. From this, the user would enter one of the listed commands.

Since FTP is about file transfer, we shall get a file:

```
FTP> get <remote_file_name> <local_file_name>
```

This means *get* the file specified (remote_file_name) and copy it to the local file system under the name of <local_file_name>. If <local_file_name> is not specified, the remote file would be copied to the local file system with the same name.

Figure 6.33 FTP proces

The FTP process being set up and point, the user cou program by typing:

```
FTP> close
FTP> quit

C:>
```

If multiple files we or MPUT, which sta

Table 6.11 listed the file we wanted are examples of bin word "binary" at th

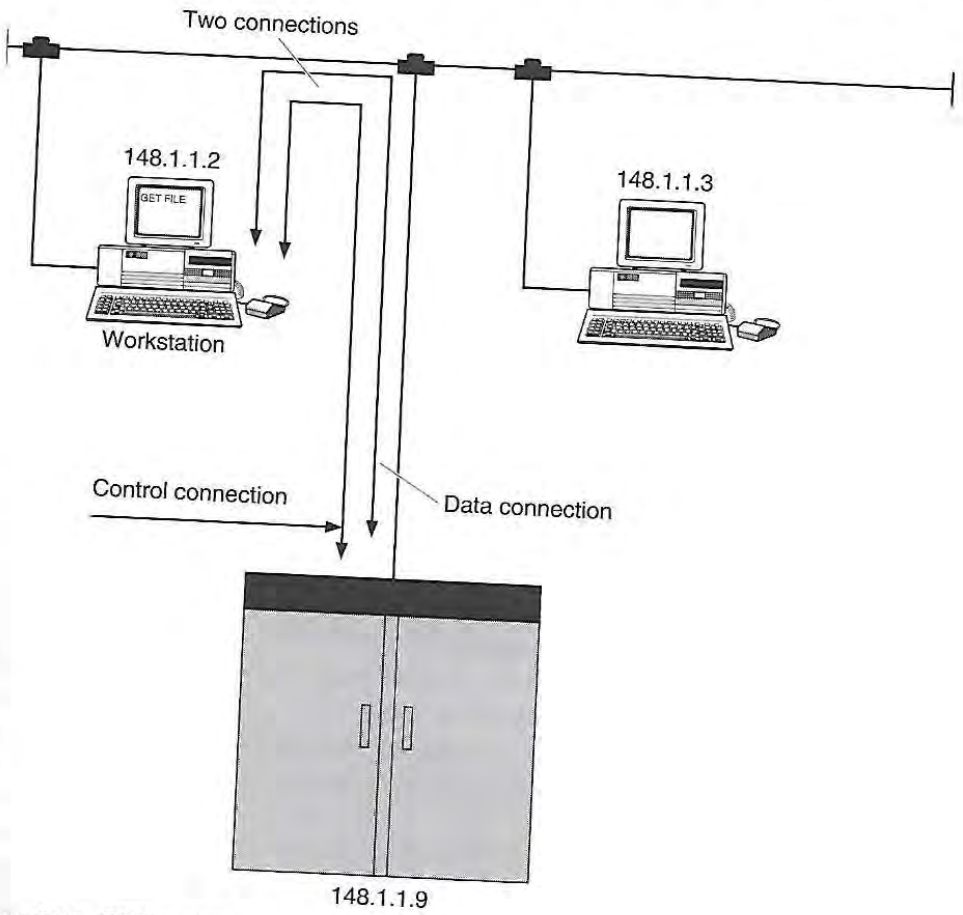

Figure 6.33 FTP process.

The FTP process would then determine that a data connection was being set up and whether or not the transfer was complete. At this point, the user could get more files or put more files, and just quit the program by typing:

```
FTP> close
FTP> quit

C:>
```

If multiple files were needed, the user could use the command MGET or MPUT, which stands for multiple GET and multiple PUT.

Table 6.11 listed the available commands under the FTP prompt. If the file we wanted was a binary file (a spreadsheet and an application are examples of binary files), the user would have to type in the keyword "binary" at the FTP prompt. This would indicate to the FTP pro-

gram that file to be transferred is a binary file. Any of the commands may be entered at the FTP prompt.

The user could also enter "disconnect," which would disconnect the current connection but leave the FTP program open so that the user could enter another connection.

Trivial file transfer program (TFTP)

An alternative to the FTP program is the TFTP program. This is a simplex file transfer program and is primarily used to bootstrap diskless network workstations (the program is small enough to fit in a ROM chip on the diskless workstation to initiate a boot across the network) or even network components (network bridges and routers). The FTP program is an extremely robust and complex program, and situations exist that require file transfer capabilities without complexity. Hence, FTP is also a larger file. TFTP is a small file and provides a more restrictive file transfer (for example, no user authentication); it is also a smaller executable software program.

There are differences between FTP and TFTP. TFTP does not provide a reliable service; therefore, it uses the transport services of UDP instead of TCP. It also restricts its datagram size to 512 bytes, and every datagram must be acknowledged (no multiple packet windowing). There are no windows for packets to be acknowledged. It could be said that it has a window of 1.

The protocol is very simple. The first packet transmitted from the client process to the server process is a control packet, which specifies the file name and whether it is to be read or written (get or put command). Subsequent packets are data packets and file transfer is accomplished with 512 bytes transferred at one time. The initial data packet is specially marked with a number of 1. Each subsequent data is incremented by 1. This is the sequence numbering system for TFTP. The receiving station will acknowledge this packet immediately upon receipt, using this mark number. Any packet of less than 512 bytes in length signifies the end of the transfer. Error messages may be transmitted in place of the data in the data field, but any error message will terminate the transmission. Also notice that only one connection is made to the remote resource. FTP has one for data and one for control information. The commands of get and put are used the same as the FTP program.

The sequencing of the data is accomplished through TFTP, not the transport-layer service of UDP. UDP provides only unreliable, connectionless service. TFTP is keeping track of the sequencing of the blocks of data and the acknowledgments that should be received. For those readers familiar with NetWare, this is the same type of transaction accomplished between the NetWare Control Protocol (NCP) and its underlying delivery system, called IPX. This is fully discussed in Chap. 5.

Domain name servic

The IP protocol ma
this address to con
remember the add
an impossible task
they are to remem
also noticed this an

The first entry to
known as HOST.TX
ford Research Insti
networks, gateways
responding address

This type of nam
download (using FT
cation like TELNET
so with a name inst
the host name in the
days of the Internet
becoming extremely
administrators deci

DNS is hierarchic
structure is exactly
pany. When you loo
person at the head
from this "root."

The advantage of
users can assign the
names when they m

The domain name
local part is usually
assigned by the loc
example, when app
assign a company a

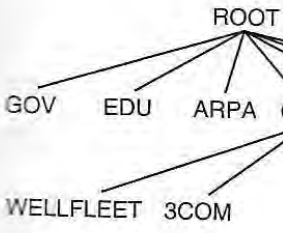

Figure 6.34 The domain h

Domain name service (DNS)

The IP protocol mandates the use of IP addresses. Any user may use this address to connect to any service on the network. But for a user to remember the addresses of all the network servers on the network is an impossible task. Users are more likely to remember names than they are to remember numbers. The architects of the TCP/IP protocol also noticed this and started work on the naming service for TCP/IP.

The first entry to an IP naming scheme was a public domain text file known as HOST.TXT. The Network Information Center (NIC) at Stanford Research Institute maintained this file, which listed the names of networks, gateways (routers), and hosts on the Internet and their corresponding addresses.

This type of name service was a simple text file that anyone could download (using FTP) to their hosts. If a user wanted to use an application like TELNET to connect to a remote station, he or she could do so with a name instead of an IP address. The application would look for the host name in the host-text file. This was simple enough for the early days of the Internet. With the expansion of the Internet, this file was becoming extremely hard to maintain. Therefore, in 1983, the Internet administrators decided to create the Domain Name Service (DNS).

DNS is hierarchical in structure, as shown in Fig. 6.34. This type of structure is exactly the same as an organization chart for any company. When you look at this type of chart, you will notice there is one person at the head of the chart with the subordinates branching out from this "root."

The advantage of this structure is that at the bottom of the chart, users can assign their own names. These users will use the upper-level names when they must access nodes outside of their region.

The domain name takes the form of local-part@domain name. The local part is usually a name of a host, a name of a user, etc. It is assigned by the local network administrator and not the NIC. For example, when applying for a network number, the NIC will also assign a company a Domain Name. As shown in Fig. 6.34, the Domain

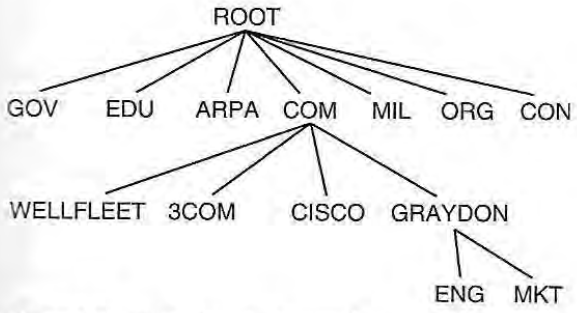

Figure 6.34 The domain hierarchy.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.