PART I

Everything You Ever Really Wanted to Know About Networking

CHAPTER 2

Defining Networking

The Network+ Certification exam expects you to know how to

- Install, Configure, and Troubleshoot networks.
- Pass the Exam!

To achieve these goals, you must be able to

- Describe the birth of networking
- Explain the goal of networking
- Explain the difference between a server and a client system
- Define a network resource

If you ask the average person, "What's a network?" you'll usually get the same basic answer: "A *network* is a bunch of computers connected together so they can share information." This answer is absolutely correct—but how will it help you *fix* networks? Instead of concentrating on "What is a network?" a network tech might find thinking in terms of "What goal does a network achieve?" far more useful when installing, configuring, and repairing networks at any level.

If I'm doing any type of work, from frying an egg to building a 500-computer network from scratch, I find it helpful to remind myself of what I want as an end result. I need a goal. When I'm frying an egg, my goal isn't simply to get the egg cooked—that's just a step in my process. My goal is to make a *delicious* cooked egg. By concentrating on the goal instead of the process, I'm not limiting myself to just getting that egg cooked. I'm thinking about how I'd like to spice the egg, how to fry it, even the color of the plate I'll use to serve it. (I really do think this way—and I make great eggs!)

Goals don't just give you an overview of the end result; they also force you to think about the entire process of achieving that goal. Imagine you discover that your car makes a funny sound every time you hit the brakes, so you take it in to the garage for repair. The mechanic then says, "You need new brake pads" and immediately starts replacing them. Now, perhaps the brake pads *are* the cause of the noise problem, but the main goal of the brakes is to stop the car. If there's a problem with your brakes, wouldn't you prefer a mechanic who keeps that goal in mind, and makes sure to verify the entire braking system? Checking the entire process, as opposed to simply reacting to single problems, makes it more likely that the less obvious problems—which are just as likely to lead to disaster—will be caught and fixed.

I do the same thing when I'm working on a network. When someone pays me to install or fix a network, I'm not thinking about sharing information; I'm thinking how this network will serve the needs of the users. Do they want the network to share information? Sure, but that's not the network's goal. The goal might be for me to install a big laser printer and configure the network so that everyone can print. Perhaps a bunch of users suddenly can't get their e-mail, and they need me to get it going again. Maybe they need me to add a big computer that they use to save important files. Getting the users to access the printer so they can print, enabling them to send and receive e-mail messages, and saving files to a central computer are all goals.

Working on a network is a lot more complicated than frying an egg or fixing the brakes on a car. This complexity makes it too easy to concentrate on only one part of the process and forget other parts. By keeping a goal for your network job in mind, you'll remember all the steps you need to get every job done; you'll get your work done faster, and you'll do it with your users in mind.

After working on networks for more years than I care to admit, I've discovered that no matter what I'm doing on a network, the goal is always the same. Everything on a network involves getting some specific thing on another computer—a printer, some e-mail, a file—to work from the comfort of the computer where the user sits.

Having thought about this for so long, I've managed to refine the goal of networking into a single sentence. Are you ready? Here it is:

> *The goal of networking is to make a resource shared by a remote system function like a resource on a local system.*

Whoa! What is a resource? What is local? What is remote? The rest of this chapter has only one job: to clarify the goal of networking in such a way that you can use it in the Network+ exam—and more important, in the real networking world. Memorize this goal. No, you won't see a "What is the goal of networking?" question on the Network+ exam, but you'll see plenty of questions on the exam that will find you thinking about this goal to help you answer them correctly. And while this goal of networking won't exactly set the room on fire if you bring it up at a party, you'd do well to recite it to yourself every time you run into a networking problem—it works!

With that goal in mind, we have a big job ahead. To help you digest the goal of networking, I need to take you through some of the most fundamental aspects of how networks function. This means we'll begin with a bit of a history lesson. This is not because I'm a networking history fan—I am, by the way—but because, if you know the history, many aspects of what's happening on your current Windows machine will make a lot more sense.

Historical/Conceptual

The Birth of Networks

At the beginning of real computing, back in the late 1960s, the world used individual *mainframe* computers. Early mainframes were physically large (initially, the size of whole buildings!), expensive computers designed to handle massive number-crunching jobs and to support multiple users. Although the word "mainframe" in this context may sound impressive, the computers sitting in our offices and homes today have far more computing power than the archaic systems of that era. Nevertheless, those early mainframes were the cutting edge of technology at the time and capable enough to put men on the moon! This cutting-edge cachet, combined with the fact that mainframes always lived behind locked doors in faraway rooms tended by geeky people who didn't talk much, gave them an aura of exclusivity and mystery that still exists today.

One aspect of mainframes did make them special: there just weren't that many of them. So, how were the geeks of yesteryear able to share such a system, ensuring that as many people as possible could use it? The earliest answer was simple—they stood in line. Early mainframes didn't have a monitor and keyboard the way PCs do today. (If you're under 35, you probably think I'm making this up, but stay with me.) If a keyboard did exist, it was on a single, large typewriter-like console in the computer room, and the operators used it to give the system detailed operating commands, like telling it to look in a specific memory address for a piece of data or program code.

Most systems loaded programs using punch cards or magnetic tape. You, as the powerless user, stood in line with your tape or stack of cards and took a number. You submitted your "job" to a person behind a counter and came back an hour (or a day) later to receive a stack of readout paper, along with your cards or tape. If you were lucky, you got a meaningful and relevant result, and shouted "Eureka!" or some other dandy phrase, much to the annoyance of the other programmers. Just as often, however, you got a pile of gibberish or an error code, at which point you went back to the keypuncher and tried to figure out what you'd done wrong. In short, early mainframe computing wasn't pretty—but it beat the heck out of doing the calculations by hand!

Fairly quickly, mainframes began to use CRT terminals and keyboards. But let's get one thing straight: these were not networks in any way! The terminals were simply data entry devices, designed to enable you to compose your programs. (Forget about ready-to-go applications—if you wanted to run a computer program for some purpose, you usually had to write it first!) The terminals themselves had no CPUs or other computer chips; they were strictly *input/output (I/O) devices*—pieces of hardware through which data flows into or out of the computer, like the keyboard and monitor on a modern PC. That's why we use the term *dumb terminal* when referring to these ancient devices. (See Figure 2-1.)

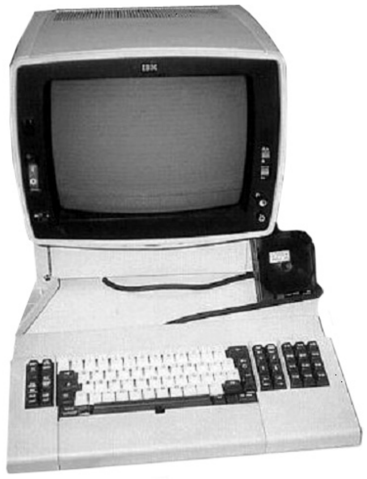


Figure 2-1 Typical dumb terminal from the 1970s

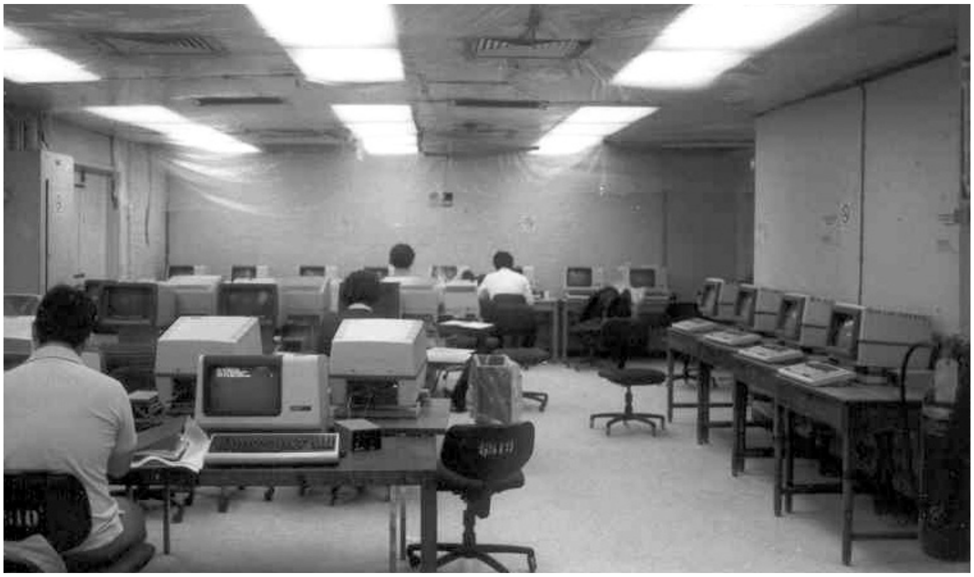
In time, a single mainframe could support dozens of dumb terminals (Figure 2-2). From a distance this resembles networking, but in this case, looks deceive. You need more than one computer to make a network. Having multiple dumb terminals attached to a single mainframe computer is roughly analogous to having a single PC with multiple monitors and keyboards. Today's PCs aren't designed to do this, but the mainframes back then had the firepower and capability to separate each user's view of the system in such a way that it worked. You could add as many dumb terminals as you wanted, but all the work still took place back at the single mainframe computer.

NOTE PCs have replaced dumb terminals in today's mainframe environments. They use special terminal emulation software that looks and acts like a dumb terminal.

Mainframe computers grew more sophisticated during the late 1960s and 1970s, incorporating features such as mass storage (hard drives) and more sophisticated operating systems to enable multiple mainframe users, each sitting at his or her dumb terminal, to access common data on mass storage devices. Users enjoyed having the capability to access common data on one mainframe, but that still wasn't networking because the data was only on one computer. By the late 1960s, however, scientists and researchers saw the benefits of enabling users on one mainframe computer to share data with users on other mainframes.

Pre-Networking Issues

The great issue that motivated the development of networking was based on the academic world's desire to share information among scholars. As mainframes began to spread into almost every school of academic thought (okay, maybe not philosophy, but I promise you that philosophy profs liked to talk about computers, even though they



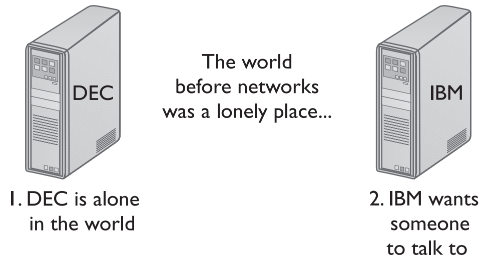
(Columbia University Academic Information Systems [1986], used by permission)

Figure 2-2 Multiple dumb terminals, from //www.columbia.edu/acis/history/

didn't use them much back in the mainframe days), the different universities wanted to enable other scholars at other locations to connect to their mainframes. Initially, the idea of networking simply didn't exist—the first idea was to come up with methods to provide dumb terminals wherever they were needed. This concept of "a terminal in every office" sounded great, but it presented two challenges that needed to be addressed before it could become a reality. First, how could they connect mainframes that were often hundreds, if not thousands, of miles apart? Second, by this time, many locations had acquired several mainframes, often from different manufacturers who used totally different operating systems, data formats, and interfaces. How could they get totally different machines—as depicted in Figure 2-3—to communicate? A lot of smart people had to work hard to come up with a way to hook computers together in this structure we eventually came to call a network.

The first great challenge was getting access to a computer physically far away from you. The answer came from an unlikely source: telephones. It took a little bit of magic,

Figure 2-3 What will it take for us to be able to talk?


DEC
The world before networks was a lonely place...
IBM
1. DEC is alone in the world
2. IBM wants someone to talk to

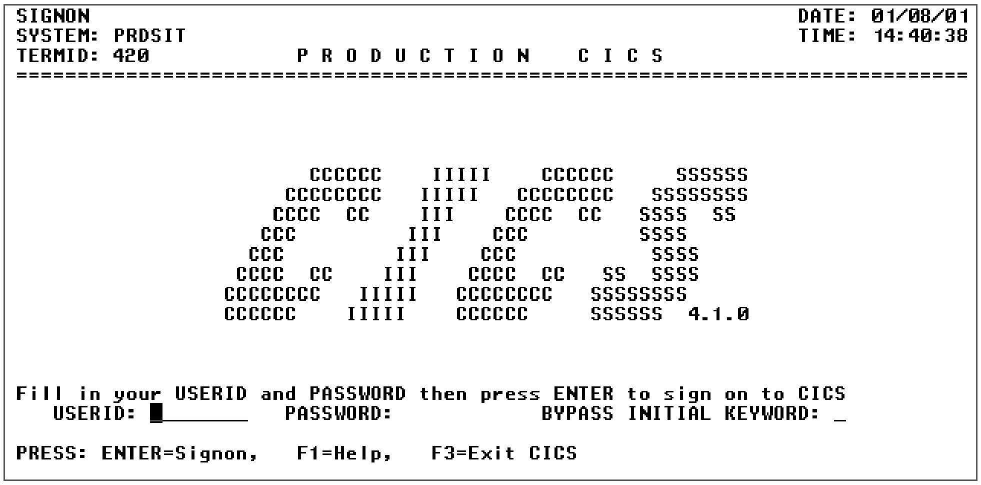
but smart people developed special devices called *modems* that enabled users to connect a dumb terminal to a far-off computer via a regular phone line. These early modems couldn't send or receive quickly—at best only around 150 characters per second. Fortunately, dumb terminals only sent and received basic I/O data (such as what key was pressed on the keyboard, or what letters appeared on the screen), so these early modems did the job.

NOTE Even though dumb terminals are virtually extinct, modems are still alive and well—in PCs!

Figure 2-4 gives you an idea of the typical computer interface on a dumb terminal. Note the lack of any graphics; the only items shown on the screen are characters—letters, numbers, and symbols. Many more years would pass before the graphical user interfaces like those in Windows were available to users!

The mainframe's primitive, character-based interface pales in comparison to a modern Windows PC's graphical desktop, but it worked well enough to get the type of work we needed done in a reasonable amount of time. While a remote terminal didn't provide networking, it did provide the idea that you could be far away from a computer and do work as though you were at a dumb terminal right next to the mainframe. This concept of long-distance connection would remain important in the minds of those who eventually began to create networks.

Remote terminals worked well for the times, but as more terminals began to appear in offices and computer rooms, another problem surfaced: different mainframe makers often required different terminals. In many situations, a school might require five or six different types of terminals just to connect to the mainframes of other schools. This be-



```
SIGNON                                                          DATE: 01/08/01
SYSTEM: PRDSIT                                                  TIME: 14:40:38
TERMID: 420            P R O D U C T I O N   C I C S
==============================================================================




                        CCCCCC    IIIII    CCCCCC     SSSSSS
                      CCCCCCCC   IIIII   CCCCCCCC   SSSSSSSS
                     CCCC  CC    III    CCCC  CC   SSSS  SS
                    CCC         III    CCC         SSSS
                   CCC         III    CCC           SSSS
                  CCCC  CC    III    CCCC  CC   SS  SSSS
                 CCCCCCCC   IIIII   CCCCCCCC   SSSSSSSS
                 CCCCCC    IIIII    CCCCCC     SSSSSS  4.1.0



Fill in your USERID and PASSWORD then press ENTER to sign on to CICS
   USERID: █_______   PASSWORD:            BYPASS INITIAL KEYWORD: _

PRESS: ENTER=Signon,   F1=Help,   F3=Exit CICS
```

Figure 2-4 Typical mainframe user interface

gan another important pre-networking step: *cross-platform support*. Terminal manufacturers began to develop standards that enabled different companies' terminals to interact with different mainframes. In this way, a professor or researcher at some remote location could use one terminal to connect to many different mainframes. This was an important idea that would later play a big part in the propagation of networks.

NOTE Dumb terminals never reached the high level of integration we see in the PC world. Sometimes you had to buy the right dumb terminal for your type of mainframe.

The proliferation of dumb terminals enabled people to connect to individual mainframe computers, but after a while, some unknown person—no doubt sitting in front of his terminal—realized the inefficiency of having to access each mainframe as a separate entity. Instead of making users jump (via their terminals) from one system to the next, why not connect the mainframes in such a way that accessing the local mainframe would provide access to the others?

Just imagine the things those users could do with such an arrangement! A professor in Cambridge, Massachusetts, could send a message to another professor in San Jose, California, electronically instead of mailing a letter! It would be electronic mail! We could nickname it *e-mail*! A defense contractor could create a series of technical documents and then send them to a military procurement team, enabling the generals to make changes to the actual document without having to print the thing! By interconnecting the mainframes, no one would ever need more than one terminal! Heck, manufacturers had already started to think about cross-platform standards, so they were ready to work together to come up with a method to interconnect different mainframes. Throw in remote terminals, and you could sit in your house and do everything in your pajamas! A Brave New World! Paperless office! Information at your fingertips! Woo hoo!

There was only one little problem. No one had ever done this before. It landed on the plate of a United States government agency to create the first practical network, the now famous *ARPANET*.

ARPANET

Computer historians trace the beginnings of networking to a number of now-famous research papers that discussed the myriad issues involved in making a workable network. Long before any real network ever existed, researchers spent years theorizing about networking. Most people agree that the first practical network ever created was ARPANET. ARPANET was conceived by an organization called the Advanced Research Projects Agency (ARPA).

ARPA was created in 1958 by President Eisenhower, the same president who created another important network, the Interstate Highway System. ARPA is more commonly referred to as *DARPA* (Defense Advanced Research Projects Agency), and it still exists today (www.darpa.mil). Its name changed from ARPA to DARPA in 1972, back to ARPA in 1993, and back to DARPA again in 1996; so I'll refer to it as DARPA in this book. DARPA is a consortium of federal organizations and researchers who work on a number of highly technical projects for the U.S. government. DARPA was the organization that first funded a small project to pull together the existing mass of theoretical research and try to create a practical, working network.

NOTE DARPA is still alive and well, developing advanced computer technologies for the U.S. government.

The earliest version of ARPANET successfully interconnected four mainframes in late 1969. Initially, ARPANET only provided two types of data transfer. First was the *File Transfer Protocol* (FTP). FTP, still popular today, enabled users to transfer files from one mainframe to another. The second was called *Telnet*. Telnet was a cool way to control another mainframe from the comfort of your own local mainframe session. You could log into your local mainframe, then Telnet into another mainframe and enter commands, just as if you were seated at a terminal connected directly to that mainframe. This was a tremendous advantage over the older idea of connecting to one mainframe, and then needing to connect to another mainframe over a separate line. One connection gave you access to all of the other computers. E-mail came soon after. Networking was born!

NOTE From its early, four-computer start, ARPANET slowly evolved into what we now call the Internet. For a detailed history of the Internet, check out the excellent articles at the Smithsonian online, http://smithsonian.yahoo.com/arpanet.html. Another good source is Michael Hauben's "History of ARPANET: Behind the Net—The untold history of the ARPANET" here: http://www.dei.isep.ipp.pt/docs/arpa.html.

The idea of networking was old hat by the time PCs first appeared in the early 1980s. A level of expectation existed that PCs could network with other PCs, but unlike mainframes, the idea of having far-flung PCs interconnect to each other wasn't obvious when they first came out. Instead, early PC networks took on a far less ambitious niche. Groups of PCs, physically close to one another, were interconnected to form what we now call *local area networks (LANs)*. Yet, even though the first PC networks were humble compared to ARPANET, the legacy of ARPANET lives on in every PC network in existence. ARPANET defined almost the entire scope of networking concepts we use in today's networks. Concepts like "server" and "client" first came into existence with this now-ancient network. If you're unfamiliar with the terms "server" and "client" as they apply to networks, fear not—we cover them in the section "Servers and Clients."

NOTE Many people tend to separate LANs from the Internet. In the most basic sense, though, the major difference between the Internet and a LAN is nothing more than size!

Test Specific

The Goal of Networking

Folks often make two big mistakes when they initially attempt to understand networking. First, they fail to appreciate the phenomenal complexity of even the simplest networks. Second, they fail to understand the goal of networking. I'll deal with the complexity issue in a moment. Right now, let's think about the goal of networking. The magic word here is "sharing."

A single mainframe computer with a zillion terminals can't really share. Granted, all those terminals provide multiple access points to its data, but remember—all the data is on a single computer. For something to be a network, there must be more than one computer. This is a critical issue, and one that comes up even in today's post-mainframe world.

NOTE A network must consist of more than one computer.

Let's get back to the concept of sharing. Assuming we have more than one system, what is there on the other system that we want to access? To put it another way, what do we want to share?

What to Share?

The designers of PC networks used ARPANET as a guide for how PC networks should work. By the mid-1980s, ARPANET had evolved into the Internet and there must have been a strong temptation to look to ARPANET for a model of what services to share in a PC network. But that's not what happened. The problem was this: ARPANET offered all sorts of services, such as FTP and Telnet, that went way beyond the conceivable scope of a little PC LAN. Why would you Telnet into a computer, for example, that was just around the corner in another office? It would make more sense to walk to the second machine, sit down, and start typing!

Rather than trying to re-create ARPANET on PC LANs, the companies that wrote the early versions of PC networking software—IBM, Microsoft, and Novell—concentrated on only two items to share: folders and printers, the two most obvious needs of a small LAN. It was not until the mid-1990s that we saw PC network makers develop the software to enable a PC to connect to the Internet. Even though early PC networks were separate from the Internet, Novell and Microsoft were smart enough to appreciate that they couldn't even begin to guess what else they might want to share in the far-off future. More than anything else, they wanted to create some type of standardized networking structure, hardware, and software that would enable a network to grow and adapt as new uses came to light. Almost no one could have imagined something as amazing as the World Wide Web way back then—but they still managed to create a networking methodology that enabled existing networks to integrate the technology of the World Wide Web easily when it later came along. So, even though PC networks are based in the idea of folder and printer sharing, they had the basic software underpinnings to enable a PC to access the Internet when consumers demanded that capability in the 1990s. Today, PCs share more than just folders and printers; they also share web pages, e-mail, FTP, and even each other's desktops.

How Do We Share?

Even though a PC network shares many types of data, all the sharing *processes* work basically the same way. Let's consider two seemingly different types of networking—surfing the World Wide Web and printing to a shared printer—and see how they share a number of important similarities. When you're checking the weather or a sports score on the Web, you are asking a computer at some other location to send data in the form of a web page to your computer. That web page may be just some text and graphics, or it might be something more complex like a sound file or a bit of java script that plays on your system. The important point is this: the information, be it a text file or a picture or something more complex, is not on your computer—it's far away on another computer, and you want it on your computer so you can experience it with your monitor and speakers. With that same thought in mind, consider the act of printing to a printer on the network. That printer might be across the room or across the country, but you want that printer to act as though it is connected to the back of your system.

What does that faraway (*remote*) web page have in common with that printer on the other side of the room? Well, they both have something that you want to access on your *local* computer; by local, I mean the computer you are physically using at this moment. Both the web page and the printer certainly require data transfers, but the data from a web page is completely different from the data you send to a printer. What we need is a better term than "data," one that's generic enough to cover anything we might want to send from one machine to another. This term must also stress the idea that another system has something we need to access and use on our system. The term I like to use for this is *resources*. Networks enable computers to share resources. A resource is anything that a particular device on a particular network wants to share with other systems on the same network. Typical resources include folders, web pages, and printers, but there are also other types of resources, ones that aren't nearly as simple to visualize. For example, e-mail is a resource for transferring messages: a system somewhere has your e-mail, and you need to go get that e-mail and bring it down to your machine to read it, send responses, and so on.

Here's an interesting little tidbit. Even though all operating systems provide ways for one system to access another system's folders, no OS lets you specify which individual files to share! All operating systems enable you to share folders, but not individual files. If you want to share a file, you must share the entire folder in which the file resides. Once a folder is shared, you can then make specific rules on how a file in that folder is shared. For example, you can set a file to be "read-only," so that no one may make changes to the file.

Servers and Clients

Okay, so a network centers around the concept of shared resources—so far, so good. Now we need to determine who shares and who simply accesses the shared resource. That's where the terms *server* and *client* come into play. A *server* is a system on a network that shares resources, while a *client* is a system that accesses a shared resource. To share a resource, you must have at least one serving computer and one client computer.

I can hear you right now: "But Mike, I thought a server was one of those big super PCs that hides in a closet!" Well, yes, we do call those servers, but that is a special use of the word. Any system that shares resources on a network will work best if it has extra power to handle all the incoming requests for its shared resources. In response to this need, the PC industry makes higher-powered systems specifically designed to meet the extra demands of serving up resources. These systems have powerful, redundant hard drives, incredibly fast network connections, and other super-powerful hardware that would be complete overkill on a regular PC. And everybody calls them . . . you guessed it: servers! (See Figure 2-5.)

The key addition you must make to create a network server is not special hardware, but rather software. Any system that wants to share its resources must run a serving program. By the same token, a client system must run a client program to access shared resources on a network. (See Figure 2-6.) Thus, even though servers tend to be the musclemen of the PC world, any system able to run a serving program can be a server. Each serving program is separate. If you want a system to share folders, it must have some type of folder-sharing software. If you want a system to share web pages, it must have some form of web page-sharing software. If you want a system to share a printer, it needs some type of printer-sharing software.

If any system running a server program is a server, then can there be more than one server on a network? For that matter, can one system run multiple serving programs?

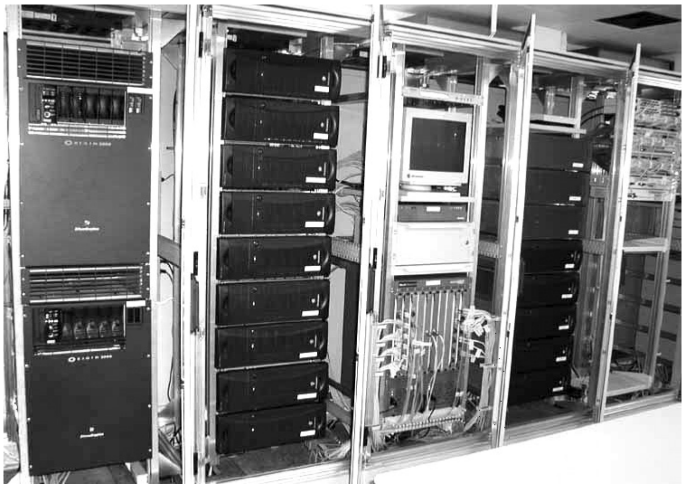


Figure 2-5 Typical servers

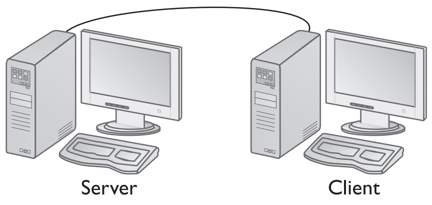


Figure 2-6 Traditional roles of networked computers

Heck, yes! It's done all the time. In my office, for example, I have one computer that runs at least seven different serving programs, sharing everything from files to e-mail to a web site. Depending on the time of day, my office also contains roughly a dozen additional systems, each of which can serve up something.

Where do these sharing programs come from? Do you have to buy them? Well, many are built into the OS. Microsoft's many versions of Windows all have serving programs either built in or easily added from the installation CD. In other cases, you might have to buy serving programs separately. Microsoft has special server versions of Windows that include many serving programs not included in the more basic versions. These Windows server operating systems, such as Windows Server 2003, cost much more than the "regular" Windows we use on our personal systems.

The final thing to appreciate is that a system can be both a server and a client at the same time. With the exception of Novell NetWare, every OS that can do networking (Windows, UNIX/Linux, and Macintosh) enables systems to act as servers and as clients at the same time. (See Figure 2-7.)

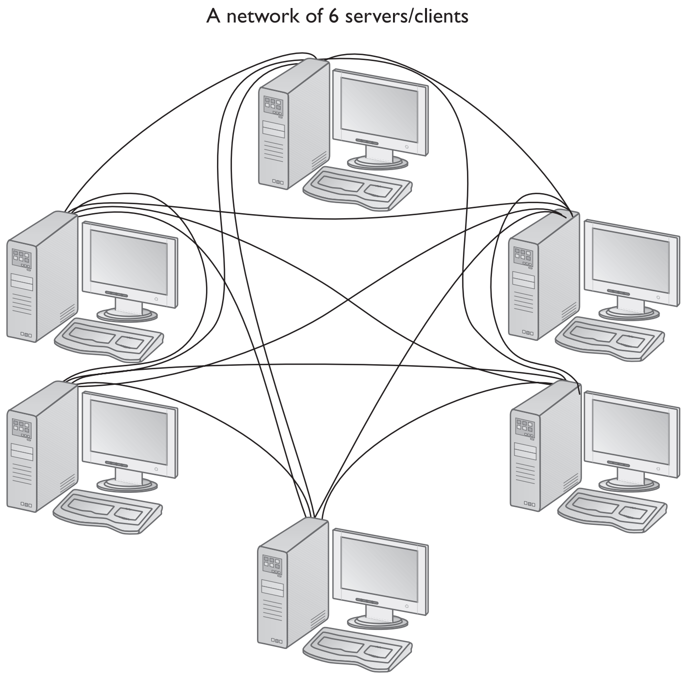


Figure 2-7 Modern roles of networked PCs

The PCs in my office are a combination of Windows 2000, Windows XP, and a few Linux computers. (I have some Windows 9*x* systems, but they're only used when people call and ask me questions about Windows 9*x*.) I have a few Microsoft Windows 2000 Server and Windows Server 2003 systems that I use to save important files. Each PC on the network, including these servers, can act as both a server and a client. This setup is common in office environments because it facilitates sharing both work files and common peripherals, like printers.

EXAM TIP The Network+ exam tests your understanding of clients and servers. Make sure you're comfortable with the fact that to share and access any type of resource, you need both a client and a server!

Making Shared Resources Usable

Okay, so servers share resources and clients access those shared resources. The last big conceptual question is this: how do we make sure that a client system—and the human using it—can use a shared resource? The better question would be this: how does a shared resource look and act as though it's local to the client system? The answer involves a two-part process. First we share the resource on the serving system, and then we access the shared resource on a client system.

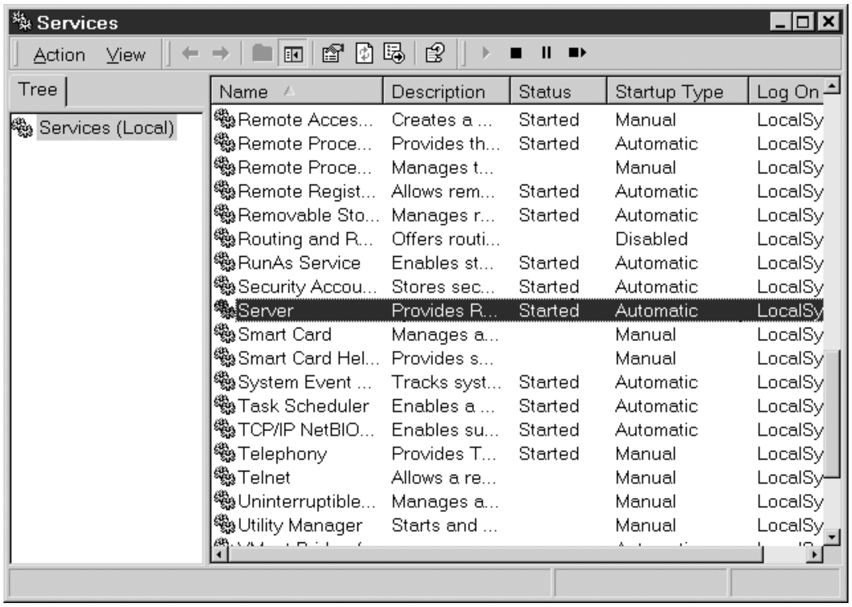
Sharing a Resource

First, the serving system must ensure that its serving software is started. How this is done varies tremendously, but it must be done. In many cases, this is an automatic process. Figure 2-8, for example, shows the Services applet in a Windows 2000 Professional system. Note the highlighted Server service. This service, which starts automatically in all versions of Windows 2000, XP, and 2003, is the main service that must run to enable you to share folders and printers.

Once a serving program has started, you must then go through some process of defining what you want to share. In Windows, you alternate-click a folder and select the Sharing menu option. Every sharing program has this step, and one of the great challenges of a network support person is to determine how to start sharing a resource once the sharing software is turned on!

Part of sharing a resource is giving it some form of network name or address so that client systems can access the shared resource. These names manifest in many ways. A web site will need a name like www.slammo.net. A shared printer will need a name like LASER1 (see Figure 2-9). No matter what type of resource you share, at some point the shared resource must have a name.

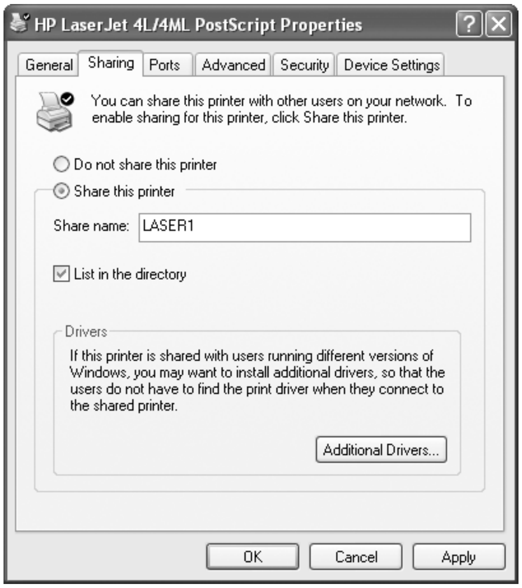
The last part of sharing a resource is defining what those who access a resource may do with it. In most cases, this means creating account names and passwords with a defined set of what I generically call *permissions*. If I share a folder, I might define the permissions so that only a certain user account can change the files, while other accounts



Name	Description	Status	Startup Type	Log On
Remote Acces...	Creates a ...	Started	Manual	LocalSy
Remote Proce...	Provides th...	Started	Automatic	LocalSy
Remote Proce...	Manages t...		Manual	LocalSy
Remote Regist...	Allows rem...	Started	Automatic	LocalSy
Removable Sto...	Manages r...	Started	Automatic	LocalSy
Routing and R...	Offers routi...		Disabled	LocalSy
RunAs Service	Enables st...	Started	Automatic	LocalSy
Security Accou...	Stores sec...	Started	Automatic	LocalSy
Server	Provides R...	Started	Automatic	LocalSy
Smart Card	Manages a...		Manual	LocalSy
Smart Card Hel...	Provides s...		Manual	LocalSy
System Event ...	Tracks syst...	Started	Automatic	LocalSy
Task Scheduler	Enables a ...	Started	Automatic	LocalSy
TCP/IP NetBIO...	Enables su...	Started	Automatic	LocalSy
Telephony	Provides T...	Started	Manual	LocalSy
Telnet	Allows a re...		Manual	LocalSy
Uninterruptible...	Manages a...		Manual	LocalSy
Utility Manager	Starts and ...		Manual	LocalSy

Figure 2-8 Windows Server Services

Figure 2-9 Giving a name to a shared printer in Windows



You can share this printer with other users on your network. To enable sharing for this printer, click Share this printer.

( ) Do not share this printer
(●) Share this printer

Share name: LASER1

☑ List in the directory

Drivers

If this printer is shared with users running different versions of Windows, you may want to install additional drivers, so that the users do not have to find the print driver when they connect to the shared printer.

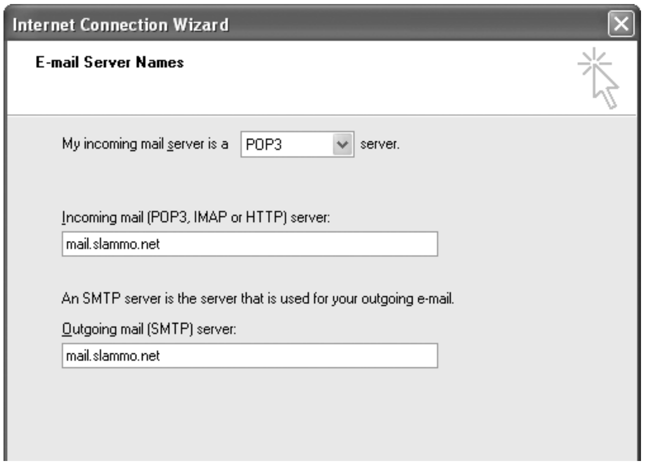
can only open and read them. If I have a web page, I might define that certain pages are only available to certain accounts. If I have an e-mail server, I might limit the amount or size of e-mails for certain accounts. Again, this permissions process varies tremendously based on the type of resource shared.

Accessing a Shared Resource

Once a serving system shares a resource, it's up to the client to go out and access that shared resource. As you might imagine, how this is done depends upon the shared resource. Remember the goal of networking: to make a shared resource look and act as though the resource were local to the client system. Let's use e-mail as an example. To access your e-mail, you need some form of e-mail client. I'll use the venerable Outlook Express as an example here. To get to your e-mail, you must know the name of the shared resource. For e-mail, this is usually a name like mail.slammo.net. In Figure 2-10, I've entered the name of the shared resource into the configuration panel. This enables me to access the shared resource.

After defining the name of the shared resource, you may have to set a username and password. Not all serving programs require this step, but more often than not, you will need to do this. Also, many serving programs use the Windows logon. It might seem you're not doing a logon, but Windows is doing it for you.

How do you know the name of the shared resource? Well, it depends on what's being shared. In a Windows network, you'll find shared printers and folders in your My Network Places—although you might have to dig a bit. You find the name of a web site by using a search engine—or simply by being told the name, reading it on a billboard, or seeing in on television. You can find out your e-mail server's name by asking the person



My incoming mail server is a POP3 server.

Incoming mail (POP3, IMAP or HTTP) server:
mail.slammo.net

An SMTP server is the server that is used for your outgoing e-mail.

Outgoing mail (SMTP) server:
mail.slammo.net

Figure 2-10 Setting the name of the e-mail server in Outlook Express

who set up the server. Regardless of the specifics, the bottom line is that you must know the name to access the shared resource.

After you've accessed the resource, you're finally at a point to enjoy the fruits of your labor and begin using that resource as though it were local to your PC. In some cases, this is easy—you just type in the name of a web site and it appears or you click Send/Receive in your e-mail client and your e-mail just starts to show up.

In other cases—especially with shared folders and printers in Windows—a remote resource might not look precisely like a local resource, but it should *act* like a local resource when you access it. It may be okay and even useful, for example, if the icon of a shared folder looks slightly different on the screen—blue instead of yellow, say, or with an added symbol—but the shared folder should be similar enough to a local folder that you can tell it's a folder. Even though the shared folder may not look exactly like a local resource, it should interact with the client system's file manager or word processing software just as if it were local to that system. Figure 2-11 shows an example of how a shared folder appears in Windows 2000.

So, in some cases, the shared resource may not look exactly like a local resource—that's okay as long as that shared resource is comprehensible and usable to the local machine.

The Goal of Networking Redux

You are now armed with three critical pieces of information about what a network must do: it must have shared resources; there must always be a client and a server; and shared resources must look, or at least act, like local resources. With these three features in mind, let's once again see the goal of networking:

> *The goal of networking is to make a resource shared by a remote system act as a resource on a local system.*

No matter what takes place on a network, no matter whether you are trying to access a web page in China or a Word document on the machine in the next office, the goal of networking stays the same. The rest of this book is nothing more than learning processes involved with enabling a network to achieve its goal. Thinking about a network in terms of its goal as opposed to simply what it is, we become better network techs—and we do better on the Network+ exam!

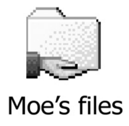


Figure 2-11
Shared folder

Chapter Review

Questions

1. Which of the following statements are true of all servers? (Select two.)
   - **A.** Servers access resources on client computers.
   - **B.** Servers make resources available for client computers to access.
   - **C.** Servers have special server hardware installed.
   - **D.** Servers have special server software installed.
2. Which of the following is not a resource that can be shared by a server?
   - **A.** Web page
   - **B.** Printer
   - **C.** Folder
   - **D.** Individual files
3. Which of the following network operating systems cannot act as both a client and a server simultaneously?
   - **A.** Windows 2000
   - **B.** Linux
   - **C.** Macintosh
   - **D.** Novell NetWare
4. Which of the following is necessary to have a network?
   - **A.** A modem
   - **B.** At least one server and two clients
   - **C.** More than one computer
   - **D.** A remote terminal
5. The goal of networking is
   - **A.** To enable remote systems to connect to each other efficiently and to access each other's files.
   - **B.** To make a resource shared by a remote system function like a resource on a local system.
   - **C.** To enable servers to access resources on one or more client systems in such a way that the shared resources are comprehensible to the accessing systems.
   - **D.** To enable users on remote terminals to access mainframe systems as if they were directly connected to those systems.

Answers

1. **B, D.** Servers must have server software installed. This software is what enables them to serve up resources to client computers. Hardware upgrades can help a server handle its workload, but are not required.
2. **D.** Interestingly, no operating system shares individual files, so we share folders instead.
3. **D.** Novell NetWare cannot act as both a client and a server simultaneously. All of the other operating systems can.
4. **C.** To have a network, you must have more than one discrete system. A mainframe connected to dumb terminals, whether remote or local, is still just a single system.
5. **B.** The goal of networking is to make a resource shared by a remote system function like a resource on a local system.