CHAPTER 3

Building a Network with OSI

The Network+ Certification exam expects you to know how to

- 2.1 Identify a MAC (Media Access Control) address and its parts
- 2.2 Identify the seven layers of the OSI (Open Systems Interconnect) model and their functions
- 2.3 Identify the OSI (Open Systems Interconnect) layers at which the following network components operate: Hubs, Routers, NICs (Network Interface Card)

To achieve these goals, you must be able to

- Explain the major functions of network hardware
- Describe the functions of network software
- Define each of these functions as part of the OSI seven-layer model

A functional PC network needs to do a number of jobs, using both hardware and software, to make a remote resource look like a local resource. Even though networking is fairly simple from a user's standpoint—most folks find getting a document to print to a networked printer trivially easy as long as everything is working properly—the design of this hardware and software to make them work together is a huge undertaking. You'll need to understand this huge undertaking if you're going to install, configure, or troubleshoot networks.

Before I begin any big job in my life, I find it beneficial to break that job down into discrete chunks or functions. Let's say I decide to move my family from Houston, Texas to Miami, Florida; this certainly counts as a big job in my book! I would start by breaking the moving job into discrete functions, such as choosing what to move, deciding how to go about packing, loading the van, choosing whether to move it myself or hire a driver, unloading the van, unpacking, and so forth—the usual tasks that anyone deals with when changing residences. Breaking a big job down into separate functions enables me both to grasp the overall task and to address each function separately.

Note that at this point, I'm not deciding any details within each function, such as how many boxes I need or what size moving van to use. Of course, these issues will eventually need addressing, but initially, I only want to understand and appreciate the scope of the necessary major functions to get my family moved! Each function may have many

optional methods—all of them perfectly acceptable—but right now, the idea is to grasp the big picture of this big job and nothing else.

Anyone who has ever moved his or her residence could probably do a pretty good job of breaking the entire home-moving process into a few major chunks—experience is a wonderful teacher! Unfortunately, few of us ever consider the big job of moving data from one computer to another. Sure, we use networks every day—but the movement of a web page to your PC is akin to walking out of your old house and driving to your new one, only to find all your furniture already neatly arranged, dinner on the stove, and your favorite football game on the television!

This chapter breaks down networking into a series of discrete steps called the OSI seven-layer model. The *OSI seven-layer model* is a guideline for what it takes to make a network. Learning about OSI in networking is about the same as learning your multiplication tables in arithmetic: it's important, but its importance isn't obvious until after you learn it. To make OSI a bit more interesting, let's not even consider it at first. Instead, let's first concentrate on the steps required to make a network function—then we can talk about this OSI thing-a-ma-bob.

Just as the process of moving your home is best learned by going through a move, the best way to learn the steps of moving data in a network is to observe the move as it takes place in a real network. For this reason, I'll begin this chapter by introducing you to a small network that needs to get a file copied from one computer to another. Using this example, we will go through each of the steps needed to get that file moved, taking time to explain each step and why it is necessary. Finally, you'll see that these steps are defined under the important OSI seven-layer model.

Historical/Conceptual

Welcome to MHTechEd!

Mike's High-Tech Educational Supply Store and Post Office, or MHTechEd for short, has a small network of PCs running Windows XP—a situation typical of many small businesses today. Windows XP runs just fine on a PC unconnected to a network, but it also comes with all the network software it needs to connect to a network, making Windows XP a *network operating system (NOS)*, as well as just an operating system. All the computers in the MHTechEd network are connected by special network cabling.

I can see some questions forming from those of you with some networking experience: "Hey Mike, what's the difference between an operating system and a network operating system?" "What type of cabling is the network using?" Well, just hang on, buckaroo! The specific type of operating system and cabling used by our example network doesn't matter for the purposes of this chapter. We could use any operating system and any type of cabling for this chapter's goal of a conceptual overview. Trust me, by the time you close this book, you'll have all the details you can imagine—but for now, think "big picture." We have to start with something! So sit back, gather your wits about you, and join me as we take a look inside the visible network.

NOTE This section is a conceptual overview of the hardware and software functions of a network. Your network may have different hardware or software, but it will share the same functions!

Without further ado, let's head over to MHTechEd and start by taking a look at the overall network. As in most offices, virtually everyone has his or her own PC. Figure 3-1 shows two workers, Janelle and Dana, who handle all the administrative functions at MHTechEd. Because of the kinds of work they do, these two often need to exchange data between their two PCs. At the moment, Janelle has just completed a new employee handbook in Microsoft Word, and she wants Dana to check it for accuracy. Janelle could transfer a copy of the file to Dana's computer by the tried-and-true *Sneakernet* method, saving the file on a floppy disk and walking it over to her, but thanks to the wonders of computer networking, she doesn't even have to turn around in her chair. Let's watch in detail each piece of the process that gives Dana direct access to Janelle's computer, so she can copy the Word document from Janelle's system to her own.

Long before Janelle ever saved the Word document on her system—when the systems were first installed—someone who knew what they were doing set up and configured all the systems at MHTechEd to be part of a common network. All this setup activity resulted in multiple layers of hardware and software that can now work together behind the scenes to get that Word document from Janelle's system to Dana's. Let's examine the different pieces of the network, and then return to the process of Dana grabbing that Word document.

Figure 3-1 Janelle and Dana—happy workers!

Test Specific

Let's Get Physical

Clearly the network needs a physical channel through which it can move bits of data between systems. Most networks use a cable like the one shown in Figure 3-2. This cable, known in the networking industry as *unshielded twisted pair (UTP)*, contains either four or eight wires that transmit data. The MHTechEd network's UTP cable uses only four: two for sending data and two for receiving.

Another key piece of hardware the network uses is a special box-like device called a *hub* (Figure 3-3), often tucked away in a closet somewhere. Each system on the network has its own cable that runs to the hub. Think of the hub as being like one of those old-time telephone switchboards, where operators created connections between persons who called in wanting to reach other telephone users. A hub departs from the switchboard/operator analogy in one way: the hub doesn't connect the "caller" to one specific "callee." Instead, the hub passes along the data received from one system to *all* the other systems, leaving each system with the job of determining whether a piece of data is meant for it. Remember this fact, because it becomes an important concept later.

Figure 3-2 UTP network cable showing wires

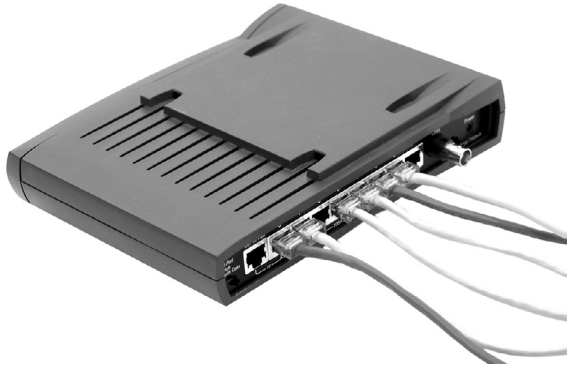
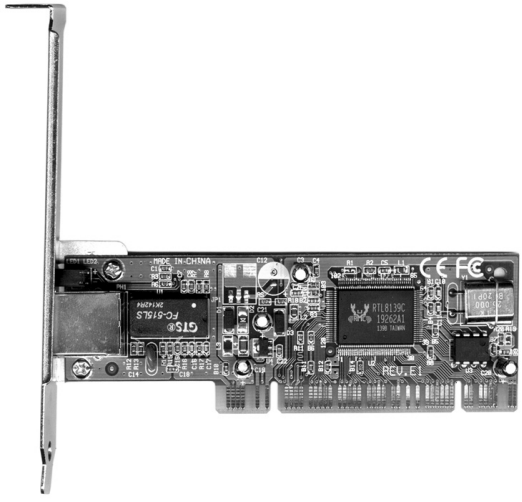


Figure 3-3 Typical hub

Figure 3-4
Typical NIC



Let's get back to the hardware, because there's another key piece you will be hearing a lot about: the *network interface card*, or *NIC* (pronounced "nick"). The real magic of a network starts with the NIC, which serves as the interface between the PC and the network. While NICs come in a wide array of shapes and sizes, the ones at MHTechEd look like Figure 3-4.

When installed in a PC, the NIC looks like Figure 3-5. Note the cable running from the back of the NIC into the wall; inside that wall is another cable running all the way back to the hub.

Now that you have a picture of all the pieces, Figure 3-6 shows a diagram of the network cabling system. I'll build on this diagram as I delve deeper into the network process.

Figure 3-5
NIC installed in a PC

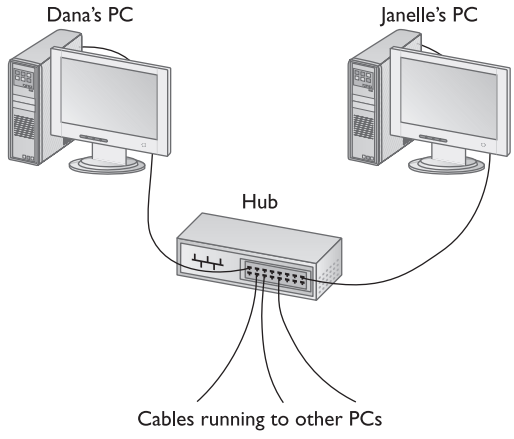


Figure 3-6 The MHTechEd network

The NIC

To understand networks, you must understand what takes place inside a NIC. If you look at the previous diagram, you'll notice that all the networked systems connect to the same hub. The network must provide a mechanism that gives each system a unique identifier—like a telephone number—so that data is delivered to the right system. That's one of the most important jobs of a NIC. Inside every NIC, burned onto some type of ROM chip, is special firmware containing a unique identifier with a 48-bit value called the *media access control address*, or *MAC address*. No two NICs ever share the same MAC address—ever. Any company that makes NICs must contact the Institute of Electrical and Electronics Engineers (IEEE) and request a block of MAC addresses, which they then burn into the ROMs on their NICs. Many NIC makers also print the MAC address on the surface of each NIC, as shown in Figure 3-7. Note that the NIC shown here displays the

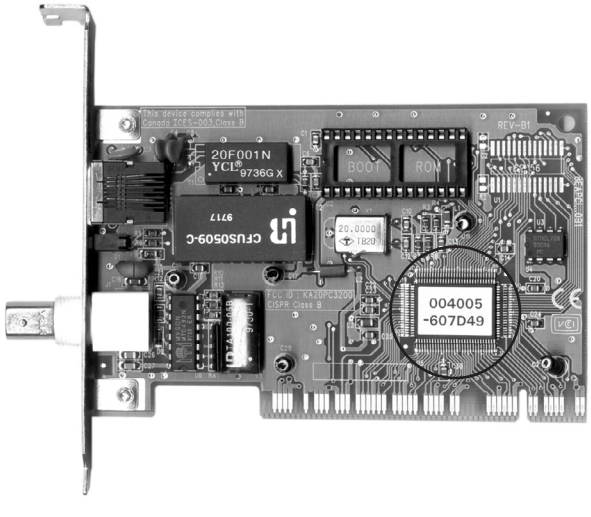


Figure 3-7 NIC with printed MAC address

MAC address in hexadecimal notation. Count the number of hex characters—because each hex character represents four bits, it takes 12 hex characters to represent 48 bits.

The MAC address in Figure 3-7 is 004005-607D49, although in print, we represent the MAC as 00–40–05–60–7D–49. The first six digits, in this example 00–40–05, represent the number of the manufacturer of the NIC. Once the IEEE issues a manufacturer those six hex digits—often referred to as the *organizationally unique identifier*, or *OUI*—no other manufacturer may use them. The last six digits, in this example 60–7D–49, are the manufacturer's unique serial number for that NIC; this portion of the MAC is often referred to as the *device ID*.

Would you like to see the MAC address for your NIC? If your system uses Windows 9*x* or Me, run the **winipcfg** command from Start | Run to see the MAC address (Figure 3-8).

If you have a Windows NT/2000/XP system, run **ipconfig /all** from a command prompt to display the MAC address (Figure 3-9).

Okay, so every NIC in the world has a unique thingy called a MAC address, but how is it used? Ah, that's where the fun begins! Recall that computer data is binary, which means it's made up of streams of ones and zeroes. NICs send and receive this binary data as pulses of electricity, light, or radio waves. The NICs that use electricity to send and receive data are the most common, so let's consider that type of NIC. The exact process by which a NIC uses electricity to send and receive data is exceedingly complicated, but lucky for you, not necessary to understand. Instead, just think of a *charge* on the wire as a *one*, and *no charge* as a *zero*. A chunk of data moving in pulses across a wire might look something like Figure 3-10.

If you put an oscilloscope on the wire measuring voltage, you'd see something like Figure 3-11.

Now, remembering that the pulses represent binary data, visualize instead a string of ones and zeroes moving across the wire (Figure 3-12).

Once you understand how data moves along the wire, the next question becomes this: how does the network get the right data to the right system? All networks transmit data by breaking whatever is moving across the network (files, print jobs, web pages, and so forth) into discrete chunks called *frames*. A frame is basically a container for a chunk of data moving across a network. The NIC creates and sends, as well as receives and reads, these frames. I like to visualize an imaginary table inside every NIC that acts as a frame creation and reading station. I see frames as those pneumatic canisters you see

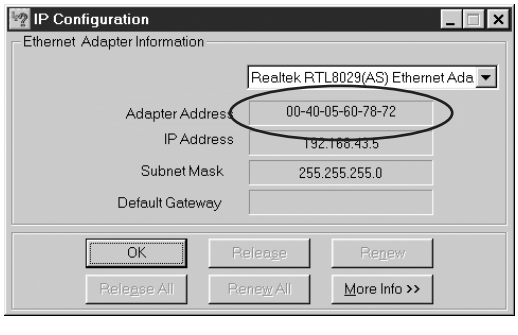


Figure 3-8 WINIPCFG (showing MAC address circled)

```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : scott64
        Primary Dns Suffix  . . . . . . . : totalhome
        Node Type . . . . . . . . . . . . : Unknown
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : totalhome

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : totalhome
        Description . . . . . . . . . . . : 3Com Gigabit LOM (3C940)
        Physical Address. . . . . . . . . : 00-0C-6E-B3-3C-68
        Dhcp Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        IP Address. . . . . . . . . . . . : 192.168.4.5
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.4.152
        DHCP Server . . . . . . . . . . . : 192.168.4.155
        DNS Servers . . . . . . . . . . . : 192.168.4.155
                                            192.168.4.156
                                            192.168.4.154
                                            192.168.4.157
        Lease Obtained. . . . . . . . . . : Wednesday, February 18, 2004 11:22:4
9 AM
        Lease Expires . . . . . . . . . . : Thursday, February 26, 2004 11:22:49
 AM

C:\>
```

Figure 3-9 IPCONFIG/ALL (MAC address is listed as Physical Address)

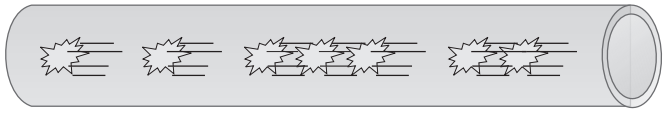


Figure 3-10 Data pulses on a wire

Figure 3-11 Oscilloscope readout of the voltages on the wire

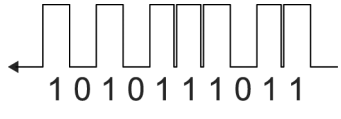


Figure 3-12 Ones and zeroes

when you go to a drive-in teller at a bank. A little guy inside the network card—named Nick, naturally!—builds these pneumatic canisters (the frames) on the table, and then shoots them out on the wire to the hub (Figure 3-13).

NOTE A number of different frame types are used in different networks. All NICs on the same network must use the same frame type or they will not be able to communicate with other NICs.

Here's where the MAC address becomes important. Figure 3-14 shows a representation of a generic frame. Even though a frame is a string of ones and zeroes, we often draw frames as a series of rectangles, each rectangle representing a part of the string of ones and zeroes. You will see this type of frame representation used quite often, so you should become comfortable with it (even though I still prefer to see frames as pneumatic canisters!). Note that the frame begins with the MAC address of the NIC to which the data is to be sent, followed by the MAC address of the sending NIC. Then comes the data, followed by a special bit of checking information called the *cyclic redundancy check (CRC)* that the receiving NIC uses to verify that the data arrived intact.

Most CRCs are only four bytes long, yet the average frame carries around 1,500 bytes of data. How can four bytes tell you if all 1,500 bytes in the data are correct? That's the

Figure 3-13
A frame-building table inside a NIC

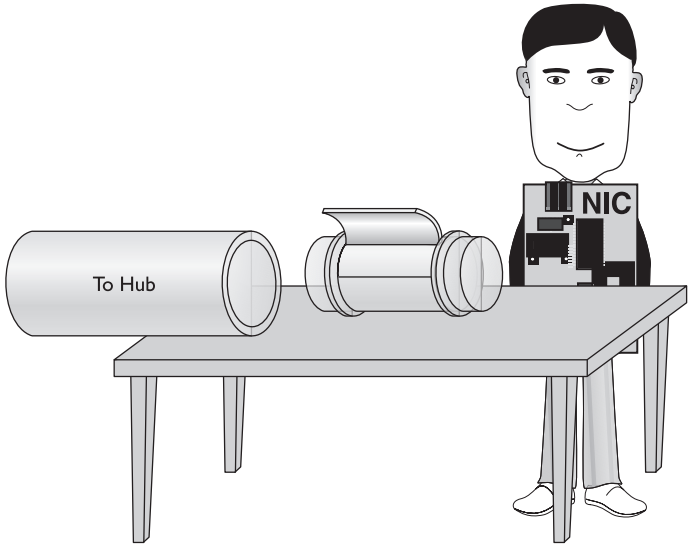
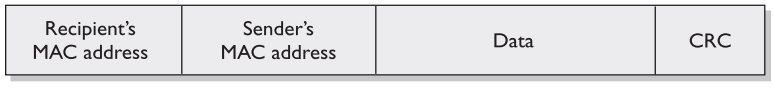


Figure 3-14
Generic frame



magic of CRCs. Without going into the grinding details, think of the CRC as just the remainder of a division problem. (Remember learning remainders from division back in elementary school?) The NIC sending the frame does a little math to make the CRC. Using binary arithmetic, it works a division problem on the data using a divisor called a *key*. This key is the same on all the NICs in your network—it's built in at the factory. The result of this division is the CRC. When the frame gets to the receiving NIC, it divides the data by the same key. If the receiving NIC's answer is the same as the CRC, it knows the data is good.

So, what's inside the data part of the frame? We neither know nor care. The data may be a part of a file, a piece of a print job, or part of a web page. NICs aren't concerned with content! The NIC simply takes whatever data is passed to it via its device driver and addresses it for the correct system. Special software will take care of *what* data gets sent and what happens to that data when it arrives. This is the beauty of imagining frames as little pneumatic canisters (Figure 3-15). A canister can carry anything from dirt to diamonds—the NIC doesn't care one bit (pardon the pun).

Like a canister, a frame can hold only a certain amount of data. Different networks use different sizes of frames, but generally, a single frame holds about 1,500 bytes of data. This raises a new question: what happens when the data to be sent is larger than the frame size? Well, the sending system's software must chop the data up into nice, frame-sized chunks, which are then handed to the NIC for sending. As the receiving system begins to accept the incoming frames, it's up to the receiving system's software to recombine the data chunks as they come in from the network. I'll show how this disassembling and reassembling is done in a moment—first, let's see how the frames get to the right system!

When a system sends a frame out on the network, the frame goes into the hub. The hub, in turn, makes an exact copy of that frame, sending a copy of the original frame to every other system on the network. The interesting part of this process is when the copy of the frame comes into all the other systems. I like to visualize a frame sliding onto the receiving NIC's "frame assembly table," where the electronics of the NIC inspect it. Here's where the magic takes place: only the NIC to which the frame is addressed will process that frame—the other NICs simply erase it when they see that it is not addressed to their MAC address. This is important to appreciate: *every* frame sent on a network is received by *every* NIC, but only the NIC with the matching MAC address will process that particular frame (Figure 3-16).

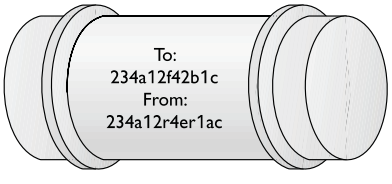


Figure 3-15 Frame as a canister

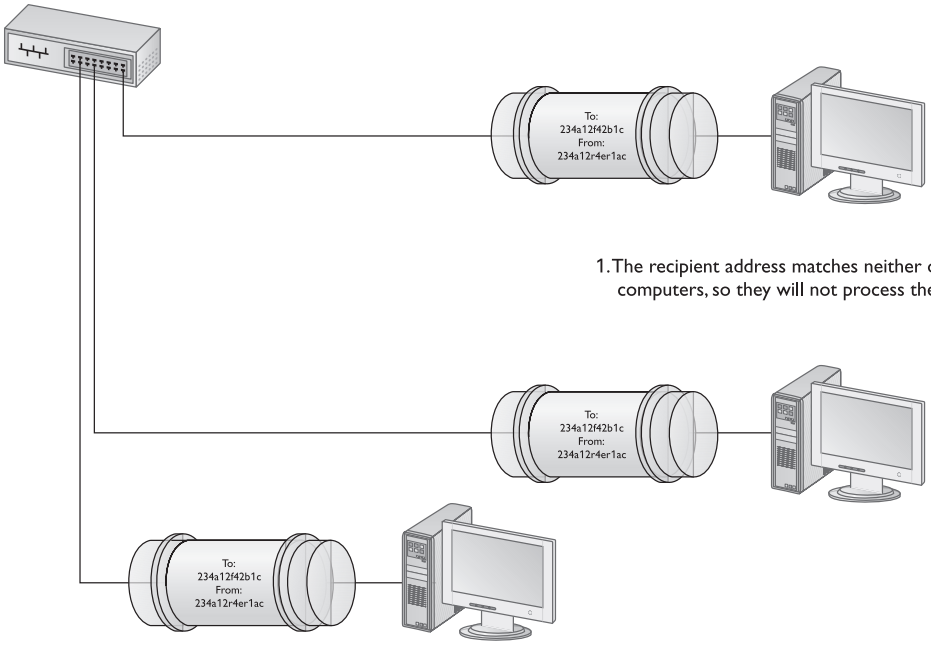


Figure 3-16 All NICs getting a frame, but only one processing it

Getting the Data on the Line

The process of getting data onto the wire, and then picking that data off the wire, is amazingly complicated. For instance, what happens to keep two NICs from speaking at the same time? Because all the data sent by one NIC is read by every other NIC on the network, only one system may speak at a time. Networks use frames to restrict the amount of data a NIC can send at once, giving all NICs a chance to send data over the network in a reasonable span of time. Dealing with this and many other issues requires sophisticated electronics, but lucky for us, the NICs handle these issues completely on their own without our help. So, thankfully, while the folks who design NICs worry about all these details, we don't have to!

Getting to Know You

Using the MAC address is a great way to move data around, but this process raises an important question. "How does a sending NIC know the MAC address of the NIC to which it's sending the data?" In most cases, the sending system already knows the destination MAC address, as the NICs had probably communicated earlier and each system stores that data. If it doesn't already know the MAC address, a NIC may send a *broadcast* onto the network to ask for it. The MAC address of FF-FF-FF-FF-FF-FF is the *broadcast address*—

if a NIC sends a frame using the broadcast address, every single NIC on the network will process that frame. That broadcast frame's data will contain a request for a system's MAC address. The system with the MAC address your system is seeking will then respond with its MAC address.

The Complete Frame Movement

Now that you've seen all the pieces used to send and receive frames, let's put these pieces together and see how a frame gets from one system to another. The basic send/receive process is as follows.

First, the sending system network software hands some data to its NIC. The NIC begins building a frame to transport that data to the receiving NIC (Figure 3-17).

After the NIC creates the frame, it adds the CRC, and then dumps it and the data into the frame (Figure 3-18).

Next, it puts both the destination MAC address and its own MAC address onto the frame. It then waits until no other NIC is using the cable, and then sends the frame through the cable to the network (Figure 3-19).

The frame propagates down the wire into the hub, which creates copies of the frame and sends it to every other system on the network. Every NIC receives the frame and checks the MAC address. If a NIC finds that a frame is addressed to it, it processes the frame (Figure 3-20); if the frame is not addressed to it, the NIC erases it.

So, what happens to the data when it gets to the *correct* NIC? First, the receiving NIC uses the CRC to verify that the data is valid. If it is, the receiving NIC strips off all the

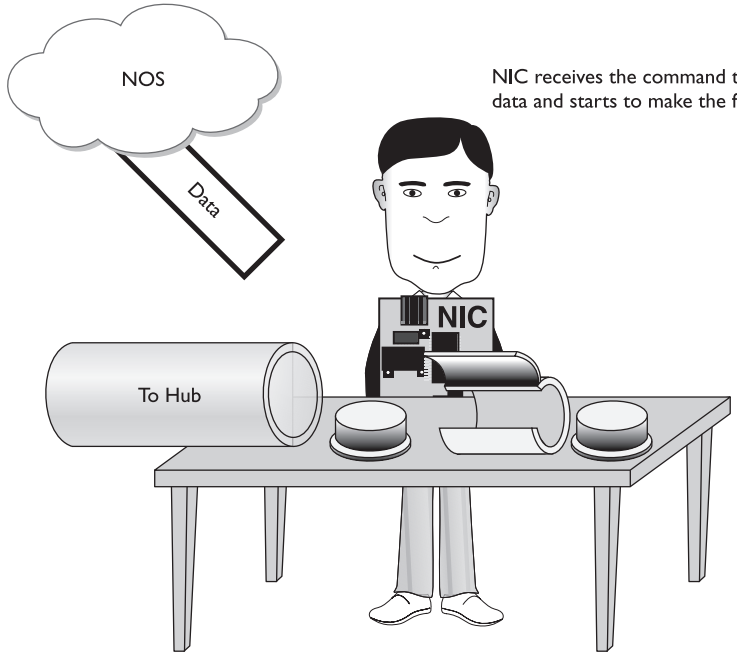


Figure 3-17 Building the frame

Figure 3-18
Adding the data and the CRC to the frame

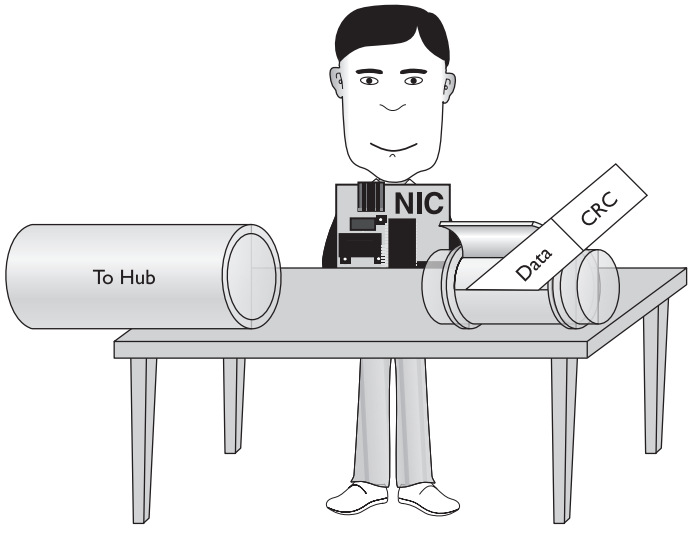
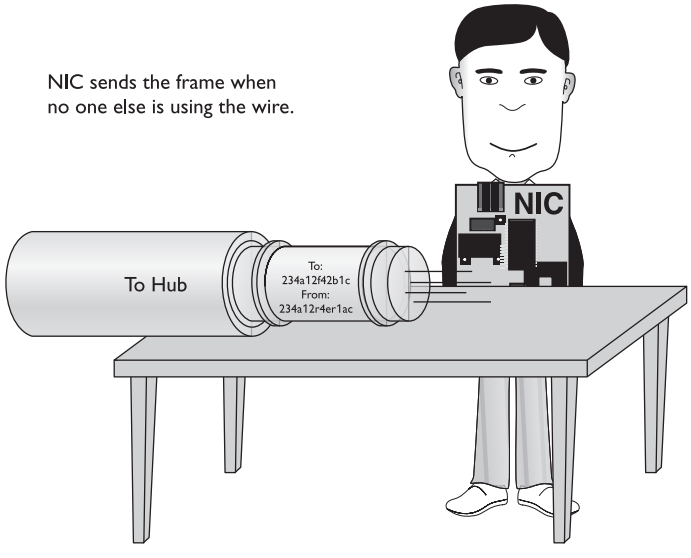


Figure 3-19
Sending the frame



framing information and sends the data to the software—the NOS—for processing. The receiving NIC doesn't care what the software does with the data; its job stops the moment it passes on the data to the NOS. *We,* however, are interested! For now, let's continue our conceptual overview—you'll learn what happens to that data when I talk about the NOS in more depth in Chapter 12, "Network Operating Systems."

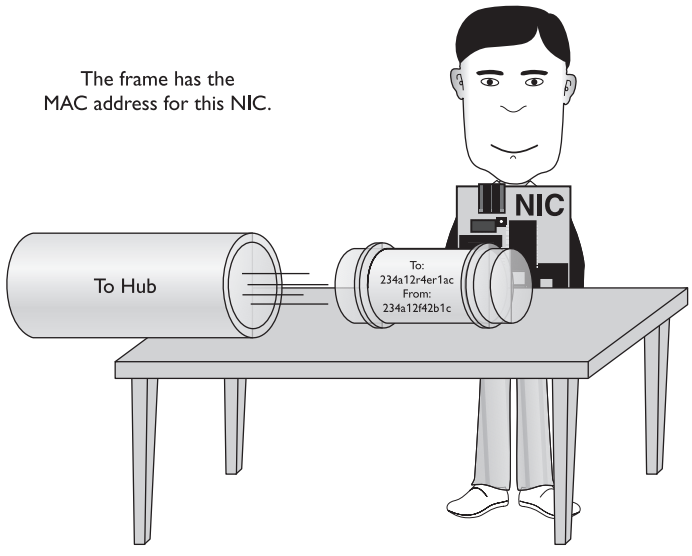


Figure 3-20
Receiving a frame

The Two Aspects of NICs

Consider how data moves in and out of a NIC. On one end, we have frames moving into and out of the NIC's network cable connection. On the other end, we have data moving back and forth between the NIC and the network operating system software. The many steps your NIC performs to keep this data moving—sending and receiving frames over the wire, creating outgoing frames and reading incoming frames, attaching MAC addresses—are classically broken down into two distinct jobs.

The first job is called the Logical Link Control (LLC). The LLC is the aspect of the NIC that talks to the operating system, places data coming from the software into frames, and creates the CRC on each frame. The LLC is also responsible for dealing with incoming frames: processing those that are addressed to this NIC and erasing frames addressed to other machines on the network.

The second job is called the Media Access Control, and I bet you can guess what it does! That's right—it remembers the NIC's own MAC address and handles the attachment of MAC addresses to frames. Remember that each frame that the LLC creates must include both the sender's and recipient's MAC addresses. The MAC also ensures that the frames, now complete with their MAC addresses, are then sent along the network cabling.

Beyond the Single Wire—Network Software

Getting data from one system to another in a "simple" network (defined as one in which all the computers connect to one hub) takes relatively little effort on the part of the NICs. But what happens when you start to use the network for more complex functions? What if someone wants to use a modem to dial into Janelle's system? Modems connect

to a network in a totally different way: they don't use MAC addresses and their frame types are completely different from the frames we use in networks. Or what if MHTechEd merges with a company that uses Macintosh computers and a different type of cabling? In these situations, you can't put these different systems on the same cable—the different frame types alone make them incompatible! (Figure 3-21.)

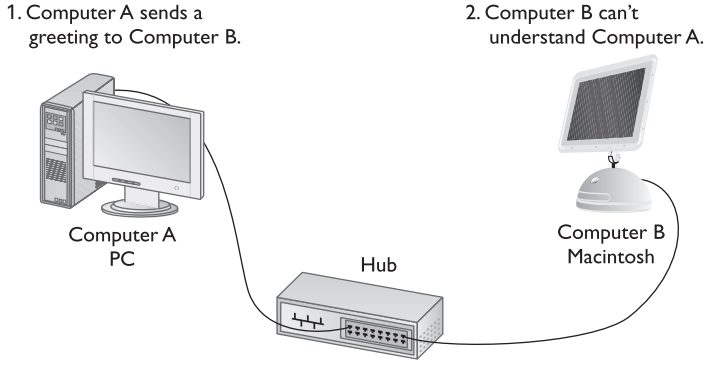


Figure 3-21 Computers using different frame types can't communicate on a network.

Even a network that uses the same frame and cabling runs into problems with MAC addresses. A single network, connecting to a single hub, cannot support more than a maximum of roughly 1,000 computers. As you add more computers to a single network, the amount of traffic becomes so great that you reach a point where the network runs unacceptably slow.

The answer to these problems comes in the form of magic little boxes called routers. *Routers* enable you to take one big network and chop it up into smaller networks. Routers also let you connect networks with different types of cabling or frames. Figure 3-22 shows a typical router. This router enables you to connect up to four computers to a cable network. Cable networks are popular for Internet connections. The connecter on the right side of the router is only for configuration.

The router in Figure 3-22 connects to a cable network and cable networks don't use MAC addresses—they have their own hardware-addressing scheme that is completely different from MAC addressing. So, when you start to connect multiple networks together with routers, each system on the network needs a more universal addressing method than MAC addresses. They need an addressing system to provide each system on the network with a unique identifier that works with any type of hardware.

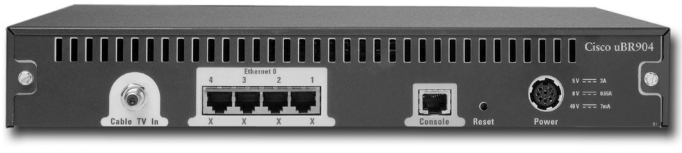


Figure 3-22 Typical cable router

This other addressing scheme shows up as special software loaded onto every computer on the network. This special software—usually called a *network protocol*—exists in almost every network-capable operating system. A network protocol not only has to create unique identifiers for each system, it must also create a set of communication rules for issues like how to handle data chopped up into multiple packets, and how to deal with routers. Let's take a moment to learn a little bit about one famous network protocol—TCP/IP and its unique universal addressing system. Then we'll return to the router to see how this all works together.

To be accurate, TCP/IP is really two sets of network protocols designed to work together—that's why there's a slash between TCP and IP. TCP stands for *Transmission Control Protocol*, and IP stands for *Internet Protocol*. IP is the network protocol I need to discuss first; rest assured, however, I'll cover TCP in plenty of detail later!

NOTE TCP/IP is the most famous network protocol, but there are plenty of others!

The IP protocol makes sure that a piece of data gets to where it needs to go on the network. It does this by giving each device on the network a unique numeric identifier. Every network protocol uses some type of naming convention, but no two protocols do it the same way. IP uses a rather unique *dotted-octet* numbering system based on four 8-bit numbers. Each 8-bit number ranges from 0 to 255, and the four numbers are separated by periods. (If you don't see how 8-bit numbers can range from 0 to 255, don't worry. By the end of this book, you'll understand these in more detail than you ever believed possible!) A typical IP address might look like this:

192.168.4.232

No two systems on the same network share the same IP address; if two machines accidentally receive the same address, a nasty error will occur. These IP addresses don't just magically appear—they must be configured. Sometimes, these numbers are typed into each system manually, but most IP networks take advantage of a groovy tool called Dynamic Host Configuration Protocol (DHCP) to configure these values automatically on each computer. We cover DHCP in Chapters 14 and 15.

What's important here is for you to appreciate that in a TCP/IP network, each system now has two unique identifiers: the MAC address and the IP address. The MAC address is literally burned into the chips on the NIC, while the IP address is simply stored in the software of the system. MAC addresses come with the NIC—we don't need to configure MAC addresses—while IP addresses must be configured through software. Here's the MHTechEd network diagram again, this time showing the IP and MAC addresses for each system (Figure 3-23).

This two-address system enables IP networks to do something really cool and powerful: using IP addresses, systems can send each other data without regard to the physical connection!

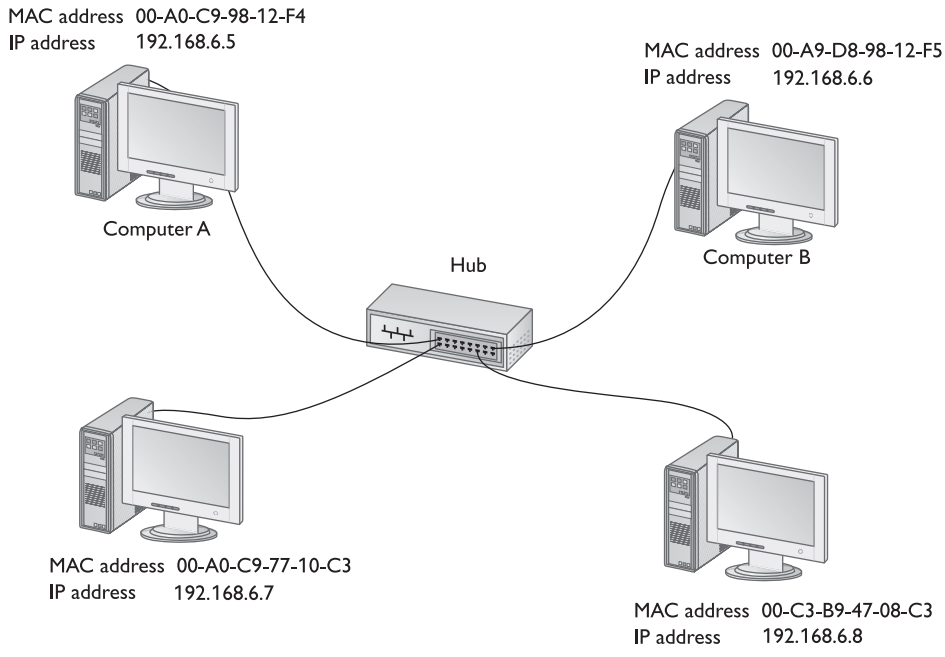


Figure 3-23 IP and MAC addresses for each system on the network

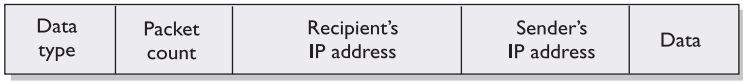
This capability requires more than the simple assignment of an IP address for each computer. The network protocol must also know where to send the frame, no matter what type of hardware the various computers are running. To do this, a network protocol also uses frames—actually, frames within frames!

There's Frames in Them Thar Frames!

Whoa! Frames within frames? What are you talking about, Mike? Never fear—I'll show you. Visualize the network protocol software as a layer between the system's software and the NIC. When the IP network protocol gets hold of data coming from your system's software, it places its own frame around that data. We call this inner frame an IP *packet*, so it won't be confused with the *frame* that the NIC will add later. Instead of adding MAC addresses to its packet, the network protocol adds sending and receiving IP addresses. Figure 3-24 shows a typical IP packet; notice the similarity to the frames you saw earlier.

NOTE This is a highly simplified IP packet—I am not including lots of little parts of the IP packet in this diagram because they are not important to what you need to understand right now—but don't worry, you'll see them later in the book!

Figure 3-24
IP packet



But IP packets don't leave their PC home naked. Each IP packet is handed to the NIC, which then encloses the IP packet in a regular frame, creating, in essence, a *packet within a frame*. I like to visualize the packet as an envelope, with the envelope in the pneumatic canister frame (Figure 3-25). A more conventional drawing would look like Figure 3-26.

Figure 3-25
An IP packet in a frame

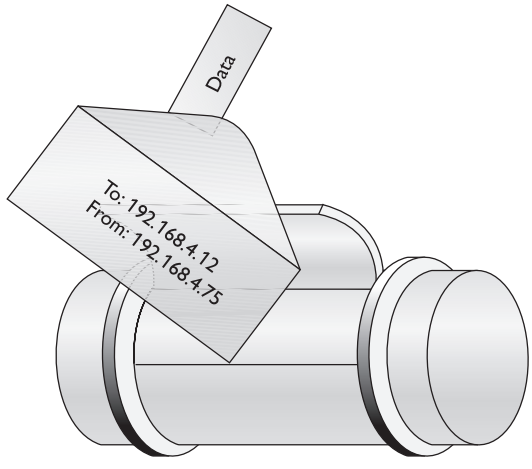


Figure 3-26
An IP packet with frame added

All very nice, you say, but why hassle with this *packet in a frame* business when you could just use MAC addresses? For that matter, why even bother with this IP thing in the first place? Good question! Let's get back to talking about routers!

Let's say that Janelle wants to access the Internet from her PC using her telephone line. We could just add a modem to her computer, but we'd rather create a way for everyone on the network to get on the Internet. To make this possible, we will connect the MHTechEd network to the Internet by adding a router (Figure 3-27).

The router that MHTechEd uses has two connections. One is just a built-in NIC that runs from the router to the hub. The other connection links the router to a telephone line. Therein lies our answer: telephone systems *don't* use MAC addresses. They use their own type of frame that has nothing to do with MAC addresses. If you tried to send a regular network frame on a phone line—well, I don't know exactly what would happen, but I assure you, it doesn't work! For this reason, when a router receives an IP packet inside a frame added by a NIC, it peels off that frame and replaces it with the type of frame the phone system needs (Figure 3-28).

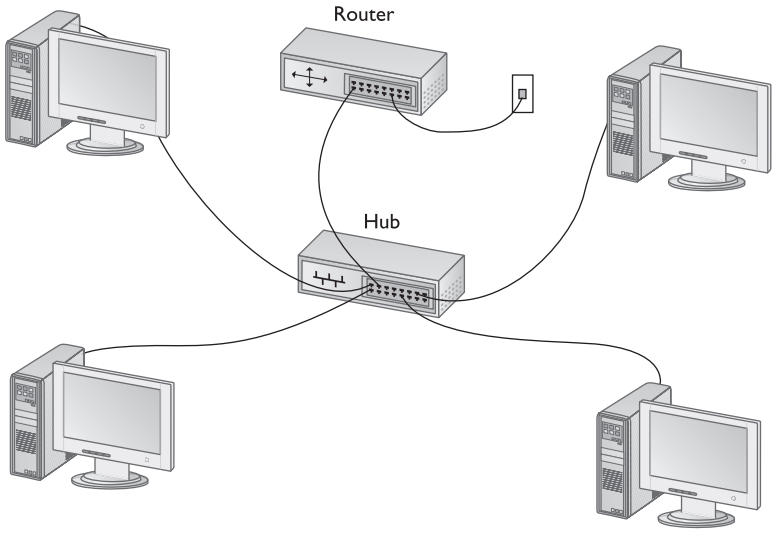


Figure 3-27
Adding a router to the network

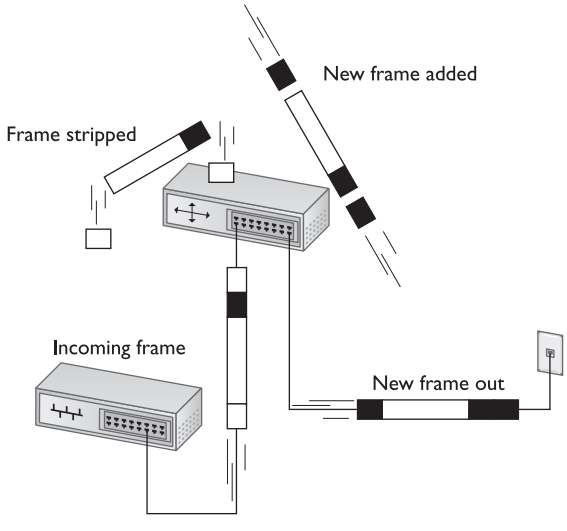


Figure 3-28
Router removing network frame and adding one for telephone line

Once the network frame is gone, so are the MAC addresses! Thus, you need some *other* naming system the router can use to get the data to the right computer—and that's why you use IP addresses on a network! After the router strips off the MAC addresses and puts on whatever type of addressing used by the telephone system, the frame flies through the telephone system, using the IP address to guide the frame to the router connected to the receiving system. At this point, the process reverses. The router rips off the telephone frame, adds the MAC address for the receiving system, and sends it on the network where the receiving system picks it up.

The receiving NIC strips away the MAC header information and passes the remaining packet off to the NOS. The networking software built into your operating system handles all the rest of the work. The NIC's driver software is the interconnection between the hardware and the software. The NIC driver knows how to communicate with the NIC to send and receive frames, but it can't do anything with the packet. Instead, the NIC driver hands the packet off to other programs that know how to deal with all the separate packets and turn them into web pages, e-mails, files, and so forth. Software handles the rest of the network function described from this point forward.

Assembly and Disassembly

Because most chunks of data are much larger than a single frame, they must be chopped up before they can be sent across a network. When a serving computer receives a request for some data, it must be able to chop the requested data into chunks that will fit into a packet (and eventually into the NIC's frame), organize the packets for the benefit of the receiving system, and hand them to the NIC for sending. The receiving system must be able to recognize a series of incoming packets as one data transmission, reassemble the packets correctly based on information included in the packets by the sending system, and verify all the packets for that piece of data arrived in good shape.

This part is relatively simple—the network protocol breaks up the data into packets and gives each packet some type of sequence number. I like to compare this process to the one that my favorite international shipping company uses. I receive boxes from UPS most every day; in fact, some days I receive many, many boxes from UPS! To make sure I get all the boxes for one shipment, UPS puts a numbering system, like the one shown in Figure 3-29, on the label of each box. A computer sending data on a network does the same thing. Embedded into the data of each packet is a sequencing number. By reading the sequencing numbers, the receiving system knows both the total number of packets and how to put them back together.

The MHTechEd network just keeps getting more and more complex, doesn't it? And you still haven't seen the Word document get copied, have you? Don't worry; you're almost there—just a few more pieces to go!

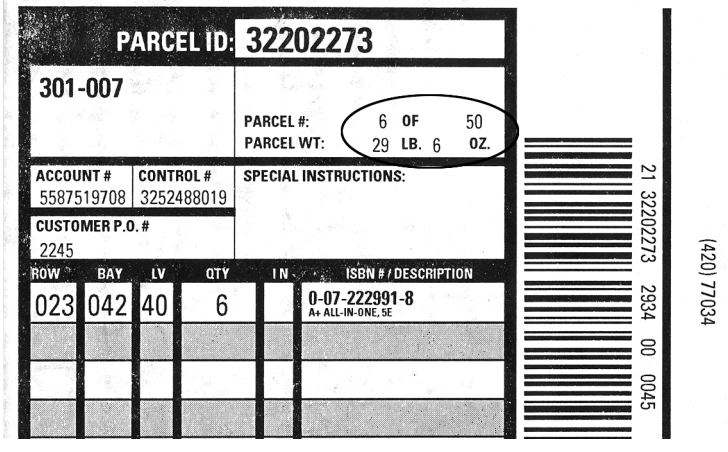


Figure 3-29 Sample UPS label showing sequencing number

Talking on a Network

Now that you understand that the system uses software to assemble and disassemble data packets, what's next? In a network, any one system may be talking to many other systems at any given moment. For example, Janelle's PC has a printer used by all the MHTechEd systems, so there's a better than average chance that as Dana tries to access the Word document, another system will be sending a print job to Janelle's PC (Figure 3-30). Janelle's system must know where to direct these incoming files, print jobs, web pages, and so on to the right programs (Figure 3-31). Additionally, the NOS must enable one system to make a connection to another system to verify that the other system can handle whatever operation the initiating system wants to perform. If Bill's system wants to send a print job to Janelle's printer, it first contacts Janelle's system to ensure that it is ready to handle the print job. We typically call the software that handles this part of networking the *session software*.

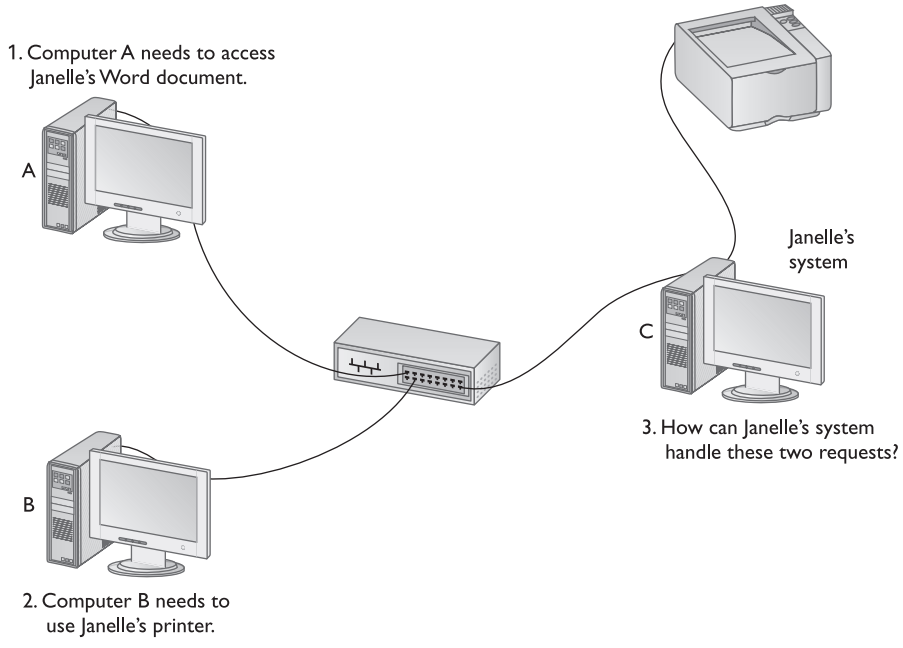


Figure 3-30 The system needs a way to handle multiple resource requests at the same time.

Standardized Formats

One of the most powerful aspects of a network lies in the fact that it works with (almost) any operating system. Today's networks easily connect, for example, a Macintosh system to a Windows 2000 PC, despite the fact that these different operating systems use different formats for many types of data. Different data formats used to drive us crazy back in

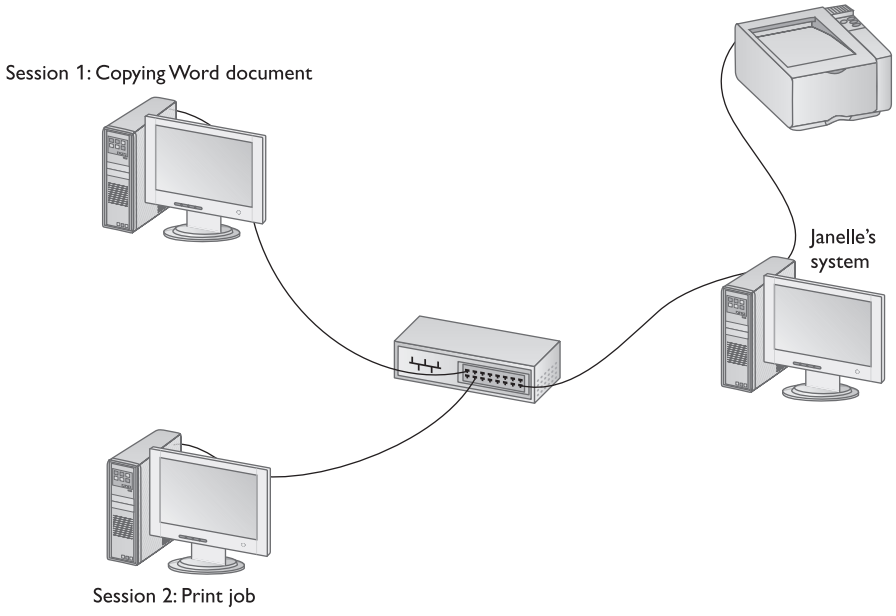


Figure 3-31 Each request becomes a session.

the days before word processors (like Microsoft Word) could import or export a thousand other word processor formats (Figure 3-32).

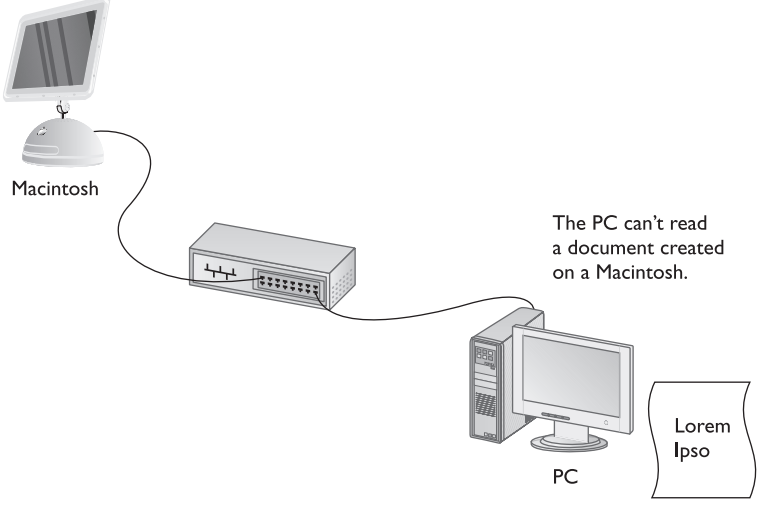


Figure 3-32 In the bad old days, differing formats could make file sharing difficult or impossible.

This created the motivation for standardized formats that anyone—at least with the right program—could read from any type of computer. Specialized file formats, such as Adobe's popular Portable Document Format (PDF) for documents and PostScript for printing, provide standard formats that any system, regardless of the operating system, can read, write, and edit (Figure 3-33).

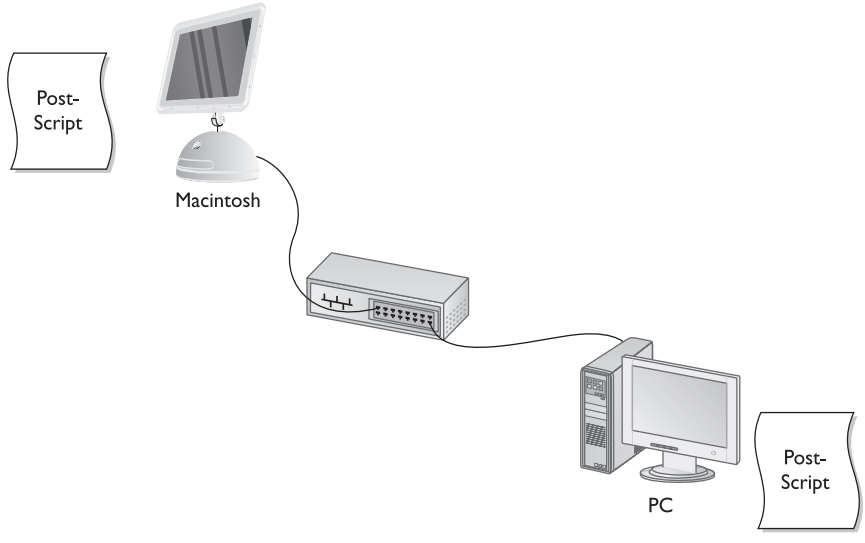


Figure 3-33 Macs and PCs both recognize Adobe PostScript.

Another function that comes into play at this point is encryption. Many networks encrypt data to prevent unauthorized access. One great example is a *Virtual Private Network (VPN)*. A VPN enables a system to access a private network via the Internet. Folks who live on the road love VPNs, because they eliminate the need to dial directly into the private network's server via a telephone line. Traveling employees can link securely into their company's private network using whatever Internet access is available to them locally. A common way VPNs manifest is through client software and server hardware (Figure 3-34).

The big problem with sending data over the Internet is security. Even a low-end hacker knows how to intercept data packets as they float by, and can look inside to see what those packets contain. Encryption stops these hackers cold—they may get the file, but they won't be able to read it if it is encrypted. For encryption to work, both the sending and the receiving system must know the encryption method, and they must be able to encrypt and decrypt on the fly (Figure 3-35).

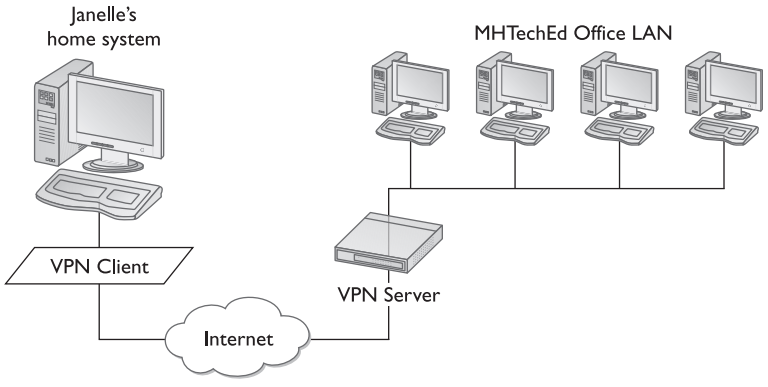


Figure 3-34 VPN diagram

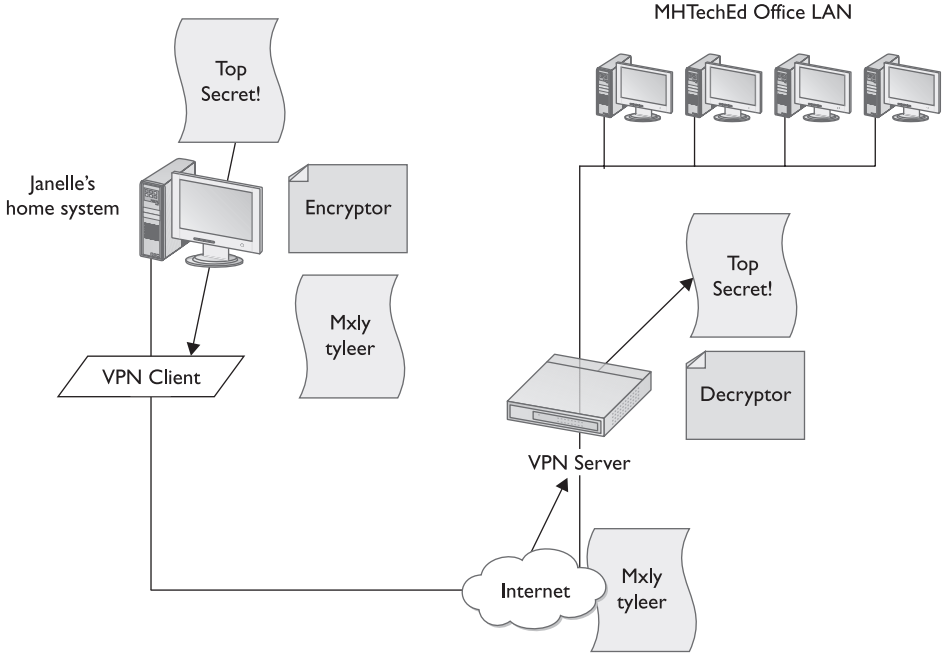


Figure 3-35 The same VPN diagram, showing encryption/decryption

Network Applications

The last, and most visible, part of any network is the software applications that use it. If you want to copy a file residing on another system in your network, you need an applica-

tion like My Network Places in Windows that lets you access files on remote systems. If you want to view web pages, you need a web browser like Internet Explorer or Netscape Navigator. The people who use a network experience it through an application. A user who knows nothing about all the other parts of a network may still know how to open an e-mail application to retrieve mail (Figure 3-36).

Figure 3-36 Network applications at work

Applications may include a number of additional functions, such as encryption, user authentication, and tools to control the look of the data. But these functions are specific to the given applications. In other words, if you want to put a password on your Word document, you must use the password functions of Word to do so.

How Dana Gets Her Document

Okay, you've now seen all the different parts of the network; keep in mind that not all networks contain all these pieces. Certain functions, such as encryption, may or may not be present, depending on the needs of the particular network. With that understanding, let's watch the network do its magic as Dana gets Janelle's Word document.

Dana has two choices for accessing Janelle's Word document. She can access the document by opening Word on her system, selecting File | Open and taking the file off Janelle's Desktop; or she can use My Network Places, My Computer, or Windows Explorer to copy the Word file from Janelle's Desktop to her computer, and then open her own copy of the file in Word. Dana wants to make changes to the document, so she chooses to copy it over to her system. This will leave an original copy on Janelle's system, so Janelle can still use it if she doesn't like Dana's changes.

Dana's goal is to copy the file from Janelle's shared Desktop folder to her system. Let's watch it happen. The process begins when Dana opens her My Network Places application. The My Network Places application shows her all the sharing computers on the MHTechEd network (Figure 3-37).

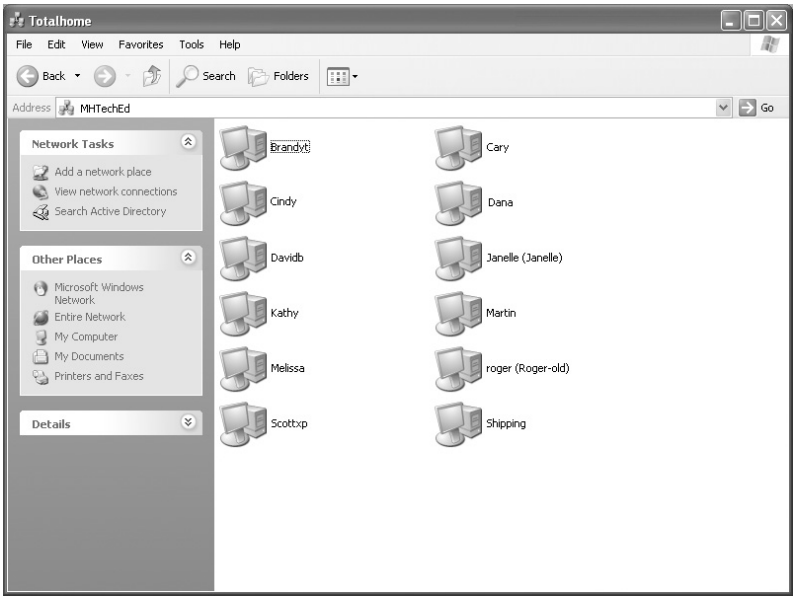


Figure 3-37 My Network Places in Windows XP

Both systems are PCs running Word, so Dana doesn't need to worry about incompatible data formats. This network does not use any encryption, but it does use authentication. As soon as Dana clicks the icon for Janelle's system in My Network Places, the two systems begin to communicate. Janelle's system checks a database of user names and privileges to see what Dana can and cannot do on Janelle's system. This checking process takes place a number of times during the process as Dana accesses various shared folders on Janelle's system. By this time, a session has been established between the two machines. Dana now opens the shared folder and locates the Word document. To copy the file, she drags and drops the Word document icon from her My Network Places onto her Desktop (Figure 3-38).

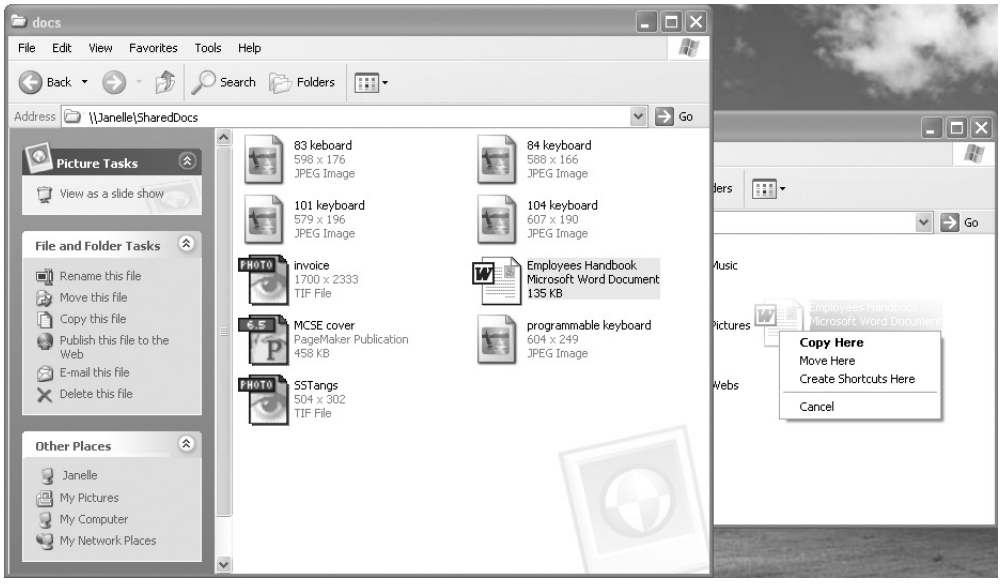


Figure 3-38 Copying the Word document

This simple act starts a series of actions. First, Janelle's system begins to chop the Word document into packets and assign each a sequence number, so that Dana's system will know how to reassemble them when they arrive on her system (Figure 3-39).

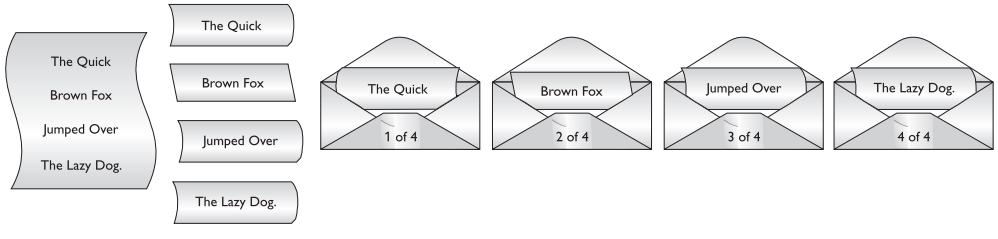


Figure 3-39 Janelle's system chopping packets

After Janelle's system chops the data into numbered packets, each packet gets the address of Dana's system, as well as Janelle's address (Figure 3-40).

The packets now get sent to the NIC for transfer. The NIC adds a frame around each packet that contains the MAC addresses for Dana's and Janelle's systems (Figure 3-41).

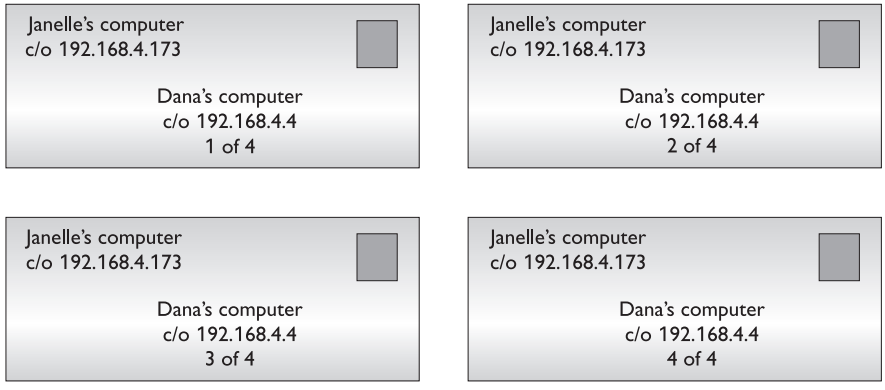


Figure 3-40 Adding packet headers

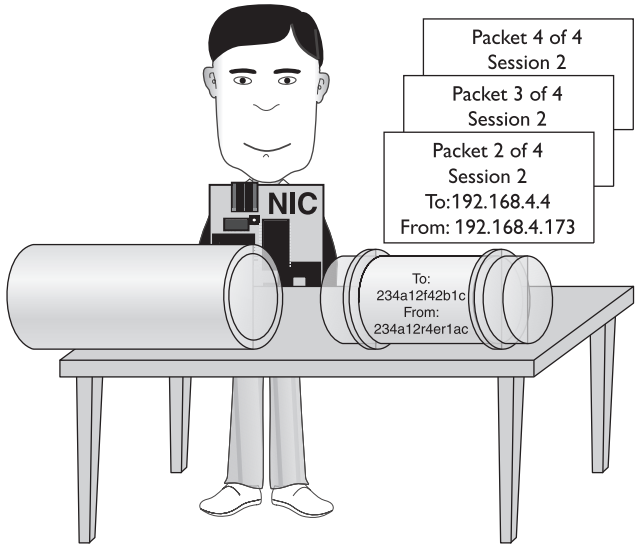
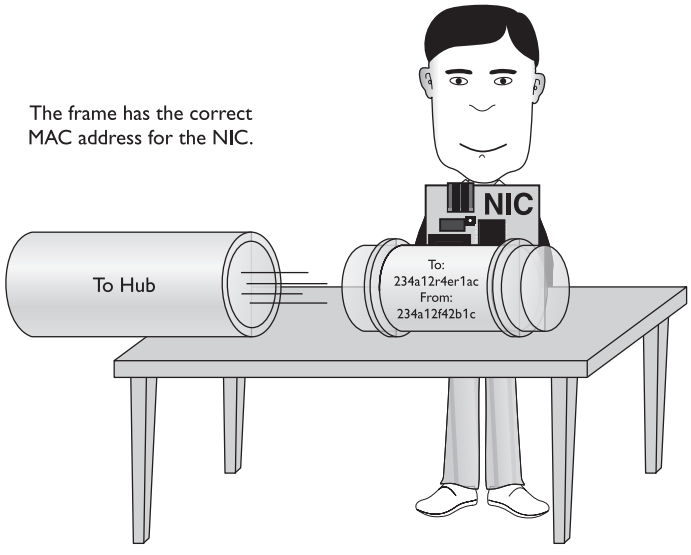


Figure 3-41 Frame being assembled

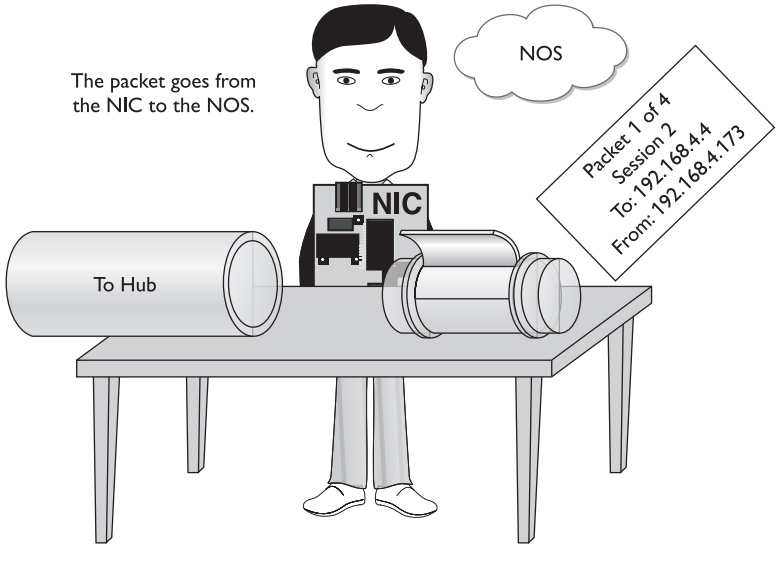
As the NIC assembles each frame, it checks the network cabling to see if the cable is busy. If not, it sends the frame down the wire. The frame goes through the hub and off to every other NIC in the network. Each NIC looks at the MAC address. All the other systems discard the frame, but Dana's system sees its MAC address and grabs it (Figure 3-42).

Figure 3-42
Dana's system grabbing a frame



As Dana's NIC begins to take in frames, it checks each one using the CRC to ensure the validity of the data in the frame. After verifying the data, the NIC strips off both the frame and the CRC and passes the packet up to the next layer (Figure 3-43).

Figure 3-43
Stripping off the frame and CRC



Dana's system then begins to reassemble the individual packets back into the complete Word document. If Dana's system fails to receive one of the packets, it simply requests that Janelle's computer resend it (Figure 3-44).

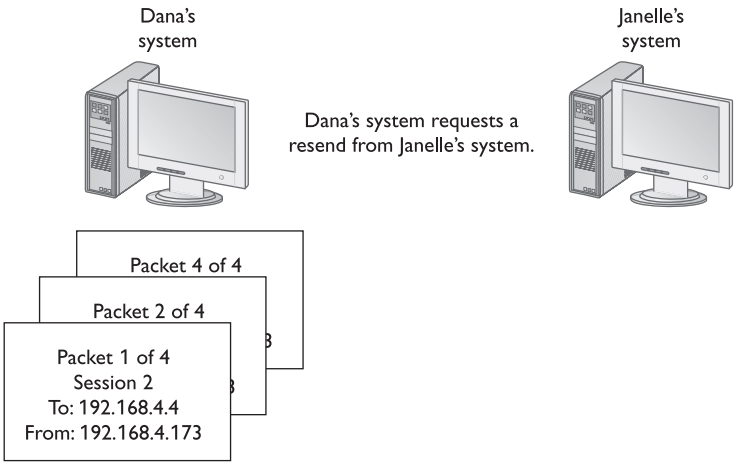


Figure 3-44 Handling a missing packet

Once Dana's system reassembles the completed Word document, it sends the document to the proper application—in this case, Windows Explorer, better known as the Desktop. Once the system copies the file to the Desktop, the network applications erase the session connection information from each system and prepare for what Dana and Janelle may want to do next.

The most amazing part of this process is that the users see virtually none of it. Dana simply opened her My Network Places, located Janelle's system, located the shared folder containing the Word document, and then just dragged and dropped the Word document onto her Desktop. This is the beauty and mystery of networks. The complexities of the different parts of software and hardware working together aren't noticed by users—nor should they be!

The OSI Seven-Layer Model

As much as I'd love to take credit for defining these steps, I admit that I'm simply using a special concept called the *OSI seven-layer model*. Folks who want to understand networks—and who want to pass the Network+ exam—must memorize and understand this handy method for conceptualizing computer networks.

Biography of a Model

In the early days of networking, lots of different folks made their own unique types of networks. For the most part, they worked well, but because each was created separately, these different networks were incapable of working together. Each one had its own hardware, drivers, naming conventions, and many other unique features that created a lot of headaches and heartaches for anyone who had to try to get them to work together. Additionally, the proprietary nature of these early networks made it difficult for other companies to create hardware or software that worked with them. It was common for one

company to supply cabling, NICs, hubs, and drivers, as well as the NOS for their brand of network, in one complete and expensive package! If the world of networking was going to grow, someone needed to create a guide, a model that described the functions of a network, so that people who made hardware and software could work together to make networks that worked together well.

The International Organization for Standardization, known as the ISO, proposed the *Open System Interconnection (OSI)* model. The OSI seven-layer model provides a precise terminology for discussing networks.

NOTE ISO may look like a misspelled acronym, but it's actually a word, derived from the Greek word *isos*, which means equal.

The Seven Layers

Most network documentation uses the OSI seven-layer model to define more precisely the role played by each protocol. The OSI model also provides a common jargon that network techs can use to describe the function of any network protocol. The model breaks up the task of networking computers into seven distinct layers, each of which addresses an essential networking task. The seven layers are

- **Layer 7** Application
- **Layer 6** Presentation
- **Layer 5** Session
- **Layer 4** Transport
- **Layer 3** Network
- **Layer 2** Data Link
- **Layer 1** Physical

EXAM TIP Be sure to memorize both the name and the number of each OSI layer. Network techs use terms such as "Layer 4" and "Transport layer" synonymously.

Each layer defines a challenge in computer networking, and the protocols that operate at that layer offer solutions to those challenges. The OSI model encourages modular design in networking, meaning that each protocol is designed to deal with a specific layer and to have as little to do with the operation of other layers as possible. Each protocol needs to understand the protocols handling the layers directly above and below it, but it can, and should, be oblivious to the protocols handling the other layers.

NOTE Keep in mind that these layers are not laws of physics—anybody who wants to design a network can do it any way they want. While many protocols fit neatly into one of the seven layers, others do not.

Layer 7: How Do Programmers Write Applications That Use the Network? The Application Layer

The *Application layer* in the OSI model defines a set of tools that programs can use to access the network. Application layer programs provide services to the programs that the users themselves see. Web browsing is a good example. Bob launches his web browser to access a web site. Web browsers use the HyperText Transfer Protocol (HTTP) to request data (usually HTML documents) from a web server. HTTP is not an executable program. It is a protocol, a set of rules that enables two other programs—the web browser and the web server—to communicate successfully with each other.

The APIs used by Microsoft networking also operate at the Application layer. An *API* is an Application Program Interface, a special set of commands that enables programmers to create applications (such as Microsoft Word) to request services from an operating system. When Microsoft Word displays My Network Places in a Save As dialog box, for example, it does not access the network directly. Instead, it uses the networking APIs. By providing a standard set of APIs that operate at the OSI's Application layer, Microsoft makes it easy for programmers writing applications like Microsoft Word to access the network without knowing any of the details of the network.

Layer 6: What Language Is This? The Presentation Layer

The job of the *Presentation layer* is to present data from the sending system in a form that the applications on the receiving system can understand. This enables different applications—Word and WordPerfect, for example—to communicate with each other, despite the fact that they use different methods to represent the same data.

Most computer systems store some form of text files for many different uses. A DOS or Windows 9*x* system usually stores text using a series of 8-bit codes known as ASCII (American Standard Code for Information Interchange), but a Windows NT, 2000, or XP system uses 16-bit Unicode to store text. A Windows 9*x* system stores the letter *A* as 01000001, while a Windows XP system stores the same letter *A* as 0000000001000001. The end users, of course, do not care about the difference between ASCII and Unicode—they just want to see the letter *A*. The Presentation layer smoothes over these differences.

The OSI model treats the Presentation layer as a distinct layer, but most real-world network operating systems fold its functions into programs that also handle either Application or Session layer functions. In fact, most network operating systems ignore the Presentation layer completely. Why? Because modern versions of Word and WordPerfect, to use my earlier example, now do this job for themselves!

NOTE Although not purely network protocols, Adobe Systems' PostScript printer language and Acrobat/PDF file format handle a typical Presentation layer problem: enabling users to view or print the same file, even if they use different operating systems or printers. *PostScript* is a device-independent printer language designed to ensure that any two PostScript-compatible printers will produce exactly the same output, regardless of the manufacturer. Adobe's PDF file format takes device independence a step further, enabling any system running an Acrobat viewer to view and print a PDF file precisely the way the author intended, regardless of what operating system or printer it uses. PostScript and Acrobat are both Presentation layer tools that hide the differences between systems.

Layer 5: How Do Machines Keep Track of Who They're Talking To? The Session Layer

The *Session layer* manages the connections between machines on the network. Suppose machine *A* receives one file from machine *B,* another from machine *C,* and sends a third file to machine *D*. Machine *A* needs some means to track its connections so that it sends the right response to the right computer. A computer managing connections is like a short-order cook keeping track of orders. Just as a cook must track which meals go with which ticket, a computer on a network must track which data should be sent out to which machine.

Layer 4: Breaking Data Up and Putting It Back Together: The Transport Layer

The *Transport layer* breaks up data it receives from the upper layers into smaller pieces—the packets—for transport. On the receiving side, the Transport layer reassembles packets from lower layers and hands the reassembled data to higher layers. The Transport layer also provides for error checking.

The Transport layer is the pivotal layer that guarantees smooth communication between the lower layers (1 through 3) and the upper layers (5 through 7). The lower layers concern themselves with moving data from point *A* to point *B* on the network, without regard for the actual content of the data. The upper layers deal with specific types of requests involving that data. This separation allows applications using the network to remain blissfully unconcerned about the workings of the underlying hardware.

Layer 3: How Do Packets Get from A to B? The Network Layer

The *Network layer* adds unique identifiers (such as the IP address described earlier) to the packets. These unique identifiers enable special devices called routers to make sure the packets get to the correct system without worrying about the type of hardware used for transmission.

Layer 2: How Do Devices Use the Wire? The Data Link Layer

The *Data Link layer* defines the rules for accessing and using the Physical layer. The majority of the Data Link functions take place inside the NIC. The Data Link layer specifies the rules for identifying devices on the network, determining which machine should use

the network at a given moment, and checking for errors in the data received from the Physical layer. Note that the functions performed at this layer affect only one sender and recipient. Data Link information does not persist beyond that single transaction, but is re-created each time the packet is transmitted to a new host.

The Data Link layer is divided into two sublayers, which you read about earlier in this chapter: Media Access Control (MAC) and Logical Link Control (LLC). The LLC sublayer is important enough to have its own IEEE standard, called 802.2. (You'll get to know the important IEEE 802.*x* series of networking standards in Chapter 4, "Hardware Concepts"; Chapter 5, "Ethernet Basics"; and beyond.) The LLC sublayer is conceptually *above* the MAC sublayer, that is, between it and the Network layer (OSI Layer 3). The MAC sublayer controls access to the Physical layer, or shared media. It encapsulates (creates the frames for) data sent from the system, adding source and destination MAC addresses, error-checking information, and decapsulates (removes the MAC addresses and CRC from) data received by the system. The LLC sublayer provides an interface with the Network layer protocols. It is responsible for the ordered delivery of frames, including retransmission of missing or corrupt packets, and for flow control (moderating data flow so one system doesn't overwhelm the other).

Layer 1: What Do These Electrical Signals Mean? The Physical Layer

Layer 1, the *Physical layer*, defines the physical form taken by data when it travels across a cable. While other layers deal with ones and zeroes, the physical layer defines the rules for turning those ones and zeroes into actual electrical signals traveling over a copper cable (or light passing through a fiber-optic cable, or radio waves generated by a wireless network, and so on). Figure 3-45 shows a sending NIC turning a string of ones and zeroes into an electrical signal, and a receiving NIC turning it back into the same string of ones and zeroes. Unless both ends of the transmission agree in advance on the physical layer rules, successful communication is not possible. The Physical layer adds no additional information to the data packet—it is concerned solely with transmitting the data provided by the layers above it.

Most networking materials that describe the OSI seven-layer model put NICs squarely into the Data Link layer of the model. It's at the MAC sublayer, after all, that data gets encapsulated into a frame, destination and source MAC addresses get added to that frame, and error checking occurs. What bothers most students with placing NICs solely in the Data Link layer is the obvious other duty of the NIC—putting the ones and zeroes on the network cable. How much more physical can you get?

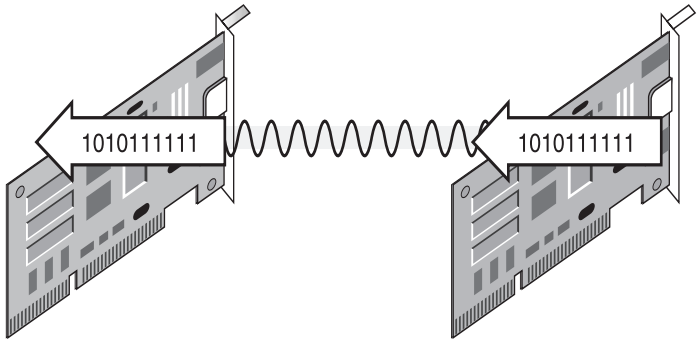


Figure 3-45 The Physical layer turns binary code into a physical signal and then back into ones and zeroes.

Many teachers will finesse this issue by defining the Physical layer in its logical sense—that it defines the rules for the ones and zeroes—and then ignore the fact that the data sent on the cable has to come from *something*. My question for a teacher who does this would be, "What component does the sending?" It's the NIC, of course, the only device capable of sending and receiving the physical signal.

Network cards, therefore, operate at both Layer 2 and Layer 1 of the OSI seven-layer model. If cornered to answer one or the other, however, go with the more common answer, Layer 2.

OSI Is the Key

The networking industry relies heavily on the OSI seven-layer model to describe the many functions that take place in a network. This chapter introduced these layers to you. Throughout the rest of the book, you'll find plenty of references to these layers as you examine every part of the network.

Chapter Review

Questions

1. Where does a hub send data?
   - **A.** Only to the receiving system.
   - **B.** Only to the sending system.
   - **C.** To all the systems connected to the hub.
   - **D.** Only to the server.
2. The unique identifier on a NIC is known as a(n)
   - **A.** IP address
   - **B.** Media access control address
   - **C.** ISO number
   - **D.** Packet ID number
3. What Windows 9*x*/Me utility do you use to find the MAC address for a system?
   - **A.** WINIPCFG
   - **B.** IFCONFIG
   - **C.** PING
   - **D.** MAC
4. On a Windows NT, 2000, or XP system, what utility do you use to find its MAC address? (Select the best answer.)
   - **A.** WINIPCFG
   - **B.** IPCONFIG

C. PING

D. MAC

5. A NIC sends data in discrete chunks called

A. Segments

B. Sections

C. Frames

D. Layers

6. A frame begins with the MAC address of the

A. Receiving system

B. Sending system

C. Network

D. Router

7. A frame ends with a special bit called the cyclic redundancy check (CRC). The CRC's job is

A. To cycle data across the network

B. To verify that the MAC addresses are correct

C. To verify that the data arrived correctly

D. To verify that the IP address is correct

8. Which of the following is an example of a MAC address?

A. 0—255

B. 00–50–56–A3–04–0C

C. SBY3M7

D. 192.168.4.13

9. Which layer of the OSI seven-layer model controls the assembly and disassembly of data?

A. Application layer

B. Presentation layer

C. Session layer

D. Transport layer

10. Which layer of the OSI seven-layer model keeps track of a system's connections to send the right response to the right computer?

A. Application layer

B. Presentation layer

C. Session layer

D. Transport layer

Answers

1. **C.** Data comes into a hub through one wire, and is then sent out through all the other wires. A hub sends data to all the systems connected to it.
2. **B.** The unique identifier on a network interface card is called the Media Access Control (MAC) address.
3. **A.** All versions of Windows 9*x* can use the WINIPCFG command to find the MAC address. The last 9*x* versions (SE and ME) could also use IPCONFIG from the command line.
4. **B.** You can use IPCONFIG/ALL from the command line to determine the MAC address of any system running Windows NT, Windows 2000, and Windows XP. You can also use WINIPCFG from the Start | Run field in Windows XP to see network settings.
5. **C.** Data is sent in discrete chunks called frames. Networks use frames to keep any one NIC from hogging the wire.
6. **A.** The frame begins with the MAC address of the receiving NIC, followed by the MAC address of the sending NIC, followed in turn by the data.
7. **C.** The data is followed by a special bit of checking information called the cyclic redundancy check, which the receiving NIC uses to verify that the data arrived correctly.
8. **B.** A MAC address is a 48-bit value, and no two NICs ever share the same MAC address—ever. 00–50–56–A3–04–0C is a MAC address. Answer D (192.168.4.13) is an IP address.
9. **D.** The Transport layer controls the assembly and disassembly of data.
10. **C.** The Session layer keeps track of a system's connections, to ensure that it sends the right response to the right computer.