Ministry of Higher Education and Scientific

Research Al-Mustaqbal University College

Department of Technical Computer Engineering

Lecture Number: 8, 9

Computer Networks 3rd Stage

Lecturer: Dr. Hussein Ali Ameen

hussein_awadh@uomus.edu.iq

2022-2023

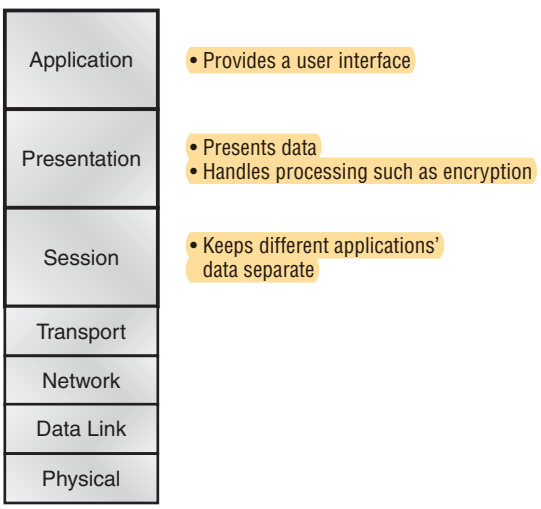
The OSI Reference Model

One of the greatest functions of the OSI specifications is to assist in data transfer between disparate hosts—meaning, for example, that they enable us to transfer data between a Unix host and a PC or a Mac.

The OSI isn't a physical model, though. Rather, it's a set of guidelines that application developers can use to create and implement applications that run on a network. It also provides a framework for creating and implementing networking standards, devices, and internetworking schemes.

The OSI has seven different layers, divided into two groups. The top three layers define how the applications within the end stations will communicate with each other and with users. The bottom four layers define how data is transmitted end to end. Figure 1.6 shows the three upper layers and their functions, and Figure 1.7 shows the four lower layers and their functions.

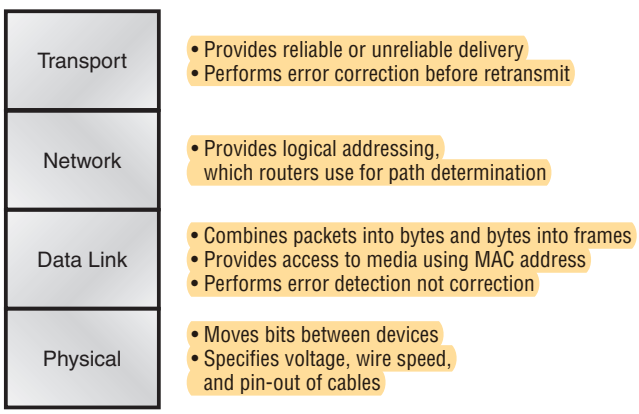
FIGURE 1.6 The upper layers



When you study Figure 1.6, understand that the user interfaces with the computer at the Application layer and also that the upper layers are responsible for applications communicating between hosts. Remember that none of the upper layers knows anything about networking or network addresses. That's the responsibility of the four bottom layers.

In Figure 1.7, you can see that it's the four bottom layers that define how data is transferred through a physical wire or through switches and routers. These bottom layers also determine how to rebuild a data stream from a transmitting host to a destination host's application.

FIGURE 1.7 The lower layers



The following network devices operate at all seven layers of the OSI model:

- Network management stations (NMSs)
- Web and application servers
- Gateways (not default gateways)
- Network hosts

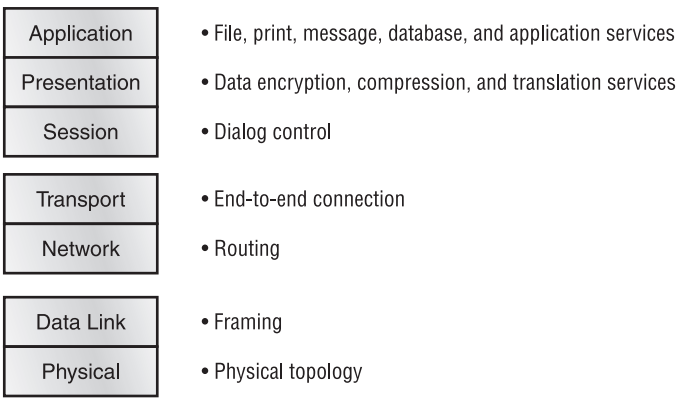
Basically, the ISO is pretty much the Emily Post of the network protocol world. Just as Ms. Post wrote the book setting the standards—or protocols—for human social interaction, the ISO developed the OSI reference model as the precedent and guide for an open network protocol set. Defining the etiquette of communication models, it remains today the most popular means of comparison for protocol suites.

The OSI reference model has seven layers:

- Application layer (layer 7)
- Presentation layer (layer 6)
- Session layer (layer 5)
- Transport layer (layer 4)
- Network layer (layer 3)
- Data Link layer (layer 2)
- Physical layer (layer 1)

Figure 1.8 shows a summary of the functions defined at each layer of the OSI model. With this in hand, you're now ready to explore each layer's function in detail.

FIGURE 1.8 Layer functions



The Application Layer

The *Application layer* of the OSI model marks the spot where users actually communicate to the computer. This layer only comes into play when it's apparent that access to the network is going to be needed soon. Take the case of Internet Explorer (IE). You could uninstall every trace of networking components from a system, such as TCP/IP, NIC card, and so on, and you could still use IE to view a local HTML document—no problem. But things would definitely get messy if you tried to do something like view an HTML document that must be retrieved using HTTP or nab a file with FTP or TFTP. That's because IE will respond to requests such as those by attempting to access the Application layer. And what's happening is that the Application layer is acting as an interface between the actual application program—which isn't at all a part of the layered structure—and the next layer down by providing ways for the application to send information down through the protocol stack. In other words, IE doesn't truly reside within the Application layer—it interfaces with Application layer protocols when it needs to deal with remote resources.

The Application layer is also responsible for identifying and establishing the availability of the intended communication partner and determining whether sufficient resources for the intended communication exist.

These tasks are important because computer applications sometimes require more than only desktop resources. Often, they'll unite communicating components from more than one network application. Prime examples are file transfers and email, as well as enabling remote access, network management activities, client/server processes, and information location. Many network applications provide services for communication over enterprise networks, but for present and future internetworking, the need is fast developing to reach beyond the limits of current physical networking.

It's important to remember that the Application layer is acting as an interface between the actual application programs. This means that Microsoft Word, for example, does not reside at the Application layer but instead interfaces with the Application layer protocols. Chapter 2 will present some programs that actually reside at the Application layer—for example, FTP and TFTP.

The Presentation Layer

The *Presentation layer* gets its name from its purpose: It presents data to the Application layer and is responsible for data translation and code formatting.

This layer is essentially a translator and provides coding and conversion functions. A successful data-transfer technique is to adapt the data into a standard format before transmission. Computers are configured to receive this generically formatted data and then convert the data back into its native format for actual reading (for example, EBCDIC to ASCII). By providing translation services, the Presentation layer ensures that data transferred from the Application layer of one system can be read by the Application layer of another one.

The OSI has protocol standards that define how standard data should be formatted. Tasks like data compression, decompression, encryption, and decryption are associated with this layer. Some Presentation layer standards are involved in multimedia operations too.

The Session Layer

The *Session layer* is responsible for setting up, managing, and then tearing down sessions between Presentation layer entities. This layer also provides dialog control between devices, or nodes. It coordinates communication between systems and serves to organize their communication by offering three different modes: *simplex*, *half duplex*, and *full duplex*. To sum up, the Session layer basically keeps different applications' data separate from other applications' data.

The Transport Layer

The *Transport layer* segments and reassembles data into a data stream. Services located in the Transport layer segment and reassemble data from upper-layer applications and unite it into the same data stream. They provide end-to-end data transport services and can establish a logical connection between the sending host and destination host on an internetwork.

Some of you are probably familiar with TCP and UDP already. (But if you're not, no worries—I'll tell you all about them in Chapter 2.) If so, you know that both work at the Transport layer and that TCP is a reliable service and UDP is not. This means that application developers have more options because they have a choice between the two protocols when working with TCP/IP protocols.

The Transport layer is responsible for providing mechanisms for multiplexing upper-layer applications, establishing sessions, and tearing down virtual circuits. It also hides details of any network-dependent information from the higher layers by providing transparent data transfer.

The term *reliable networking* can be used at the Transport layer. It means that acknowledgments, sequencing, and flow control will be used.

The Transport layer can be connectionless or connection-oriented. However, Cisco is mostly concerned with you understanding the connection-oriented portion of the Transport layer. The following sections will provide the skinny on the connection-oriented (reliable) protocol of the Transport layer.

Flow Control

Data integrity is ensured at the Transport layer by maintaining *flow control* and by allowing users to request reliable data transport between systems. Flow control prevents a sending host on one side of the connection from overflowing the buffers in the receiving host—an event that can result in lost data. Reliable data transport employs a connection-oriented communications session between systems, and the protocols involved ensure that the following will be achieved:

- The segments delivered are acknowledged back to the sender upon their reception.
- Any segments not acknowledged are retransmitted.
- Segments are sequenced back into their proper order upon arrival at their destination.
- A manageable data flow is maintained in order to avoid congestion, overloading, and data loss.

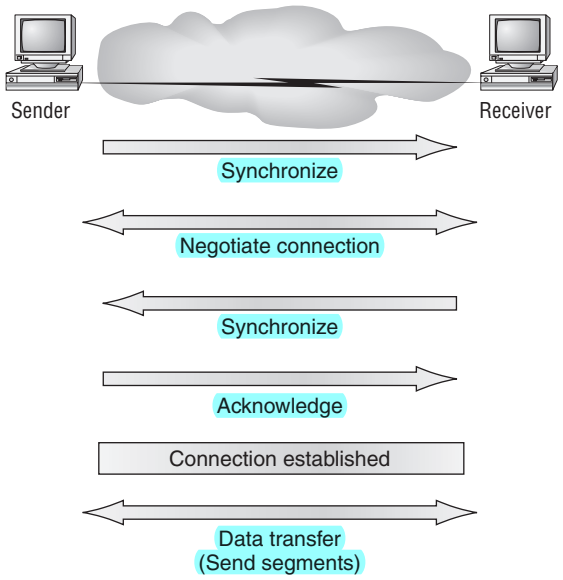
The purpose of flow control is to provide a means for the receiver to govern the amount of data sent by the sender.

Connection-Oriented Communication

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

Figure 1.9 depicts a typical reliable session taking place between sending and receiving systems. Looking at it, you can see that both hosts' application programs begin by notifying their individual operating systems that a connection is about to be initiated. The two operating systems communicate by sending messages over the network confirming that the transfer is approved and that both sides are ready for it to take place. After all of this required synchronization takes place, a connection is fully established and the data transfer begins (this virtual circuit setup is called overhead!).

FIGURE 1.9 Establishing a connection-oriented session



While the information is being transferred between hosts, the two machines periodically check in with each other, communicating through their protocol software to ensure that all is going well and that the data is being received properly.

Let me sum up the steps in the connection-oriented session—the three-way handshake—pictured in Figure 1.9:

- The first "connection agreement" segment is a request for synchronization.
- The second and third segments acknowledge the request and establish connection parameters—the rules—between hosts. These segments request that the receiver's sequencing is synchronized here as well so that a bidirectional connection is formed.
- The final segment is also an acknowledgment. It notifies the destination host that the connection agreement has been accepted and that the actual connection has been established. Data transfer can now begin.

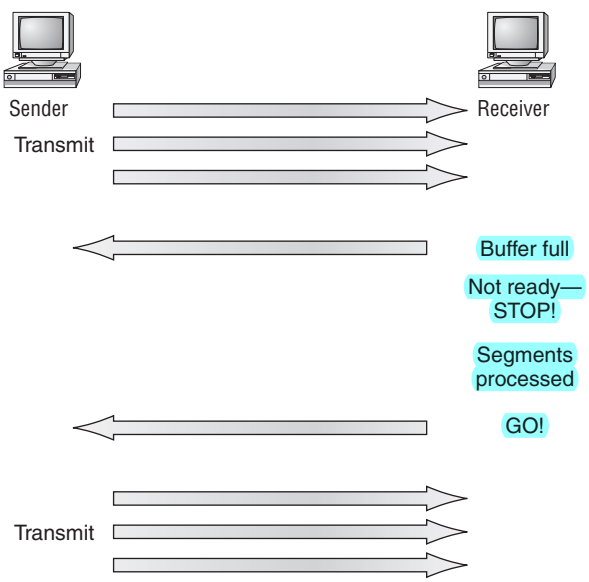
Sounds pretty simple, but things don't always flow so smoothly. Sometimes during a transfer, congestion can occur because a high-speed computer is generating data traffic a lot faster than the network can handle transferring. A bunch of computers simultaneously sending datagrams through a single gateway or destination can also botch things up nicely. In the latter case, a gateway or destination can become congested even though no single source caused the problem. In either case, the problem is basically akin to a freeway bottleneck—too much traffic for too small a capacity. It's not usually one car that's the problem; there are simply too many cars on that freeway.

Okay, so what happens when a machine receives a flood of datagrams too quickly for it to process? It stores them in a memory section called a *buffer*. But this buffering action can solve

the problem only if the datagrams are part of a small burst. If not, and the datagram deluge continues, a device's memory will eventually be exhausted, its flood capacity will be exceeded, and it will react by discarding any additional datagrams that arrive.

No huge worries here, though. Because of the transport function, network flood control systems really work quite well. Instead of dumping resources and allowing data to be lost, the transport can issue a "not ready" indicator to the sender, or source, of the flood (as shown in Figure 1.10). This mechanism works kind of like a stoplight, signaling the sending device to stop transmitting segment traffic to its overwhelmed peer. After the peer receiver processes the segments already in its memory reservoir—its buffer—it sends out a "ready" transport indicator. When the machine waiting to transmit the rest of its datagrams receives this "go" indictor, it resumes its transmission.

FIGURE 1.10 Transmitting segments with flow control



In fundamental, reliable, connection-oriented data transfer, datagrams are delivered to the receiving host in exactly the same sequence they're transmitted—and the transmission fails if this order is breached! If any data segments are lost, duplicated, or damaged along the way, a failure will transmit. This problem is solved by having the receiving host acknowledge that it has received each and every data segment.

A service is considered connection-oriented if it has the following characteristics:

- A virtual circuit is set up (e.g., a three-way handshake).
- It uses sequencing.
- It uses acknowledgments.
- It uses flow control.

The types of flow control are buffering, windowing, and congestion avoidance.

Windowing

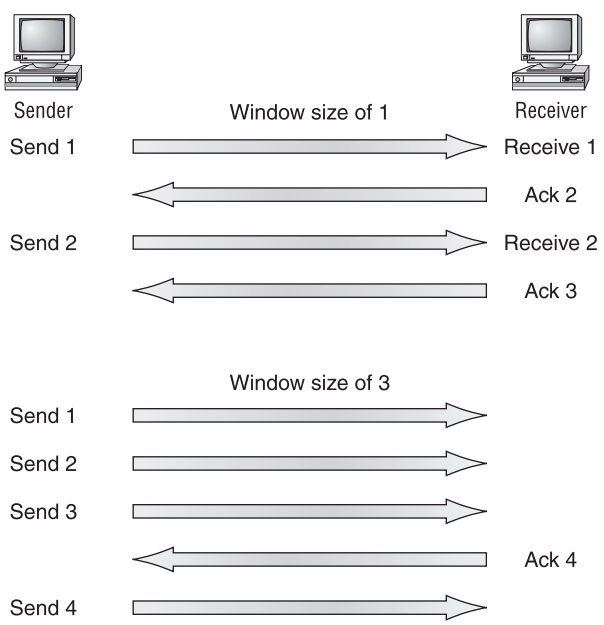
Ideally, data throughput happens quickly and efficiently. And as you can imagine, it would be slow if the transmitting machine had to wait for an acknowledgment after sending each segment. But because there's time available *after* the sender transmits the data segment and *before* it finishes processing acknowledgments from the receiving machine, the sender uses the break as an opportunity to transmit more data. The quantity of data segments (measured in bytes) that the transmitting machine is allowed to send without receiving an acknowledgment for them is called a *window*.

Windows are used to control the amount of outstanding, unacknowledged data segments.

So the size of the window controls how much information is transferred from one end to the other. While some protocols quantify information by observing the number of packets, TCP/IP measures it by counting the number of bytes.

As you can see in Figure 1.11, there are two window sizes—one set to 1 and one set to 3.

FIGURE 1.11 Windowing



When you've configured a window size of 1, the sending machine waits for an acknowledgment for each data segment it transmits before transmitting another. If you've configured a window size of 3, it's allowed to transmit three data segments before an acknowledgment is received.

In our simplified example, both the sending and receiving machines are workstations. In reality this is not done in simple numbers but in the amount of bytes that can be sent.

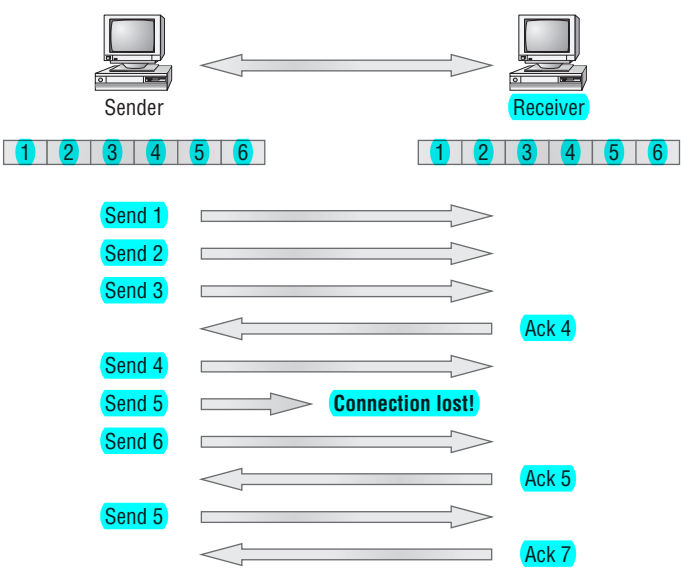
If a receiving host fails to receive all the segments that it should acknowledge, the host can improve the communication session by decreasing the window size.

Acknowledgments

Reliable data delivery ensures the integrity of a stream of data sent from one machine to the other through a fully functional data link. It guarantees that the data won't be duplicated or lost. This is achieved through something called *positive acknowledgment with retransmission*—a technique that requires a receiving machine to communicate with the transmitting source by sending an acknowledgment message back to the sender when it receives data. The sender documents each segment it sends and waits for this acknowledgment before sending the next segment. When it sends a segment, the transmitting machine starts a timer and retransmits if it expires before an acknowledgment is returned from the receiving end.

In Figure 1.12, the sending machine transmits segments 1, 2, and 3. The receiving node acknowledges it has received them by requesting segment 4. When it receives the acknowledgment, the sender then transmits segments 4, 5, and 6. If segment 5 doesn't make it to the destination, the receiving node acknowledges that event with a request for the segment to be resent. The sending machine will then resend the lost segment and wait for an acknowledgment, which it must receive in order to move on to the transmission of segment 7.

FIGURE 1.12 Transport layer reliable delivery



The Network Layer

The *Network layer* (also called layer 3) manages device addressing, tracks the location of devices on the network, and determines the best way to move data, which means that the Network layer must transport traffic between devices that aren't locally attached. Routers (layer 3 devices) are specified at the Network layer and provide the routing services within an internetwork.

It happens like this: First, when a packet is received on a router interface, the destination IP address is checked. If the packet isn't destined for that particular router, it will look up the destination network address in the routing table. Once the router chooses an exit interface, the packet will be sent to that interface to be framed and sent out on the local network. If the router can't find an entry for the packet's destination network in the routing table, the router drops the packet.

Two types of packets are used at the Network layer: data and route updates.

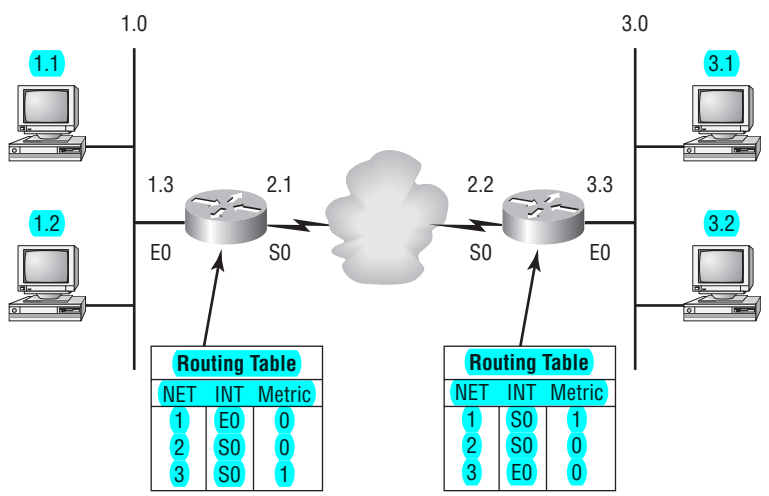
Data packets Used to transport user data through the internetwork. Protocols used to support data traffic are called *routed protocols*; examples of routed protocols are IP and IPv6. You'll learn about IP addressing in Chapters 2 and 3 and IPv6 in Chapter 13 .

Route update packets Used to update neighboring routers about the networks connected to all routers within the internetwork. Protocols that send route update packets are called routing protocols; examples of some common ones are RIP, RIPv2, EIGRP, and OSPF. Route update packets are used to help build and maintain routing tables on each router.

In Figure 1.13, I've given you an example of a routing table. The routing table used in a router includes the following information:

Network addresses Protocol-specific network addresses. A router must maintain a routing table for individual routing protocols because each routing protocol keeps track of a network with a different addressing scheme (IP, IPv6, and IPX, for example). Think of it as a street sign in each of the different languages spoken by the residents that live on a particular street. So, if there were American, Spanish, and French folks on a street named Cat, the sign would read Cat/Gato/Chat.

FIGURE 1.13 Routing table used in a router

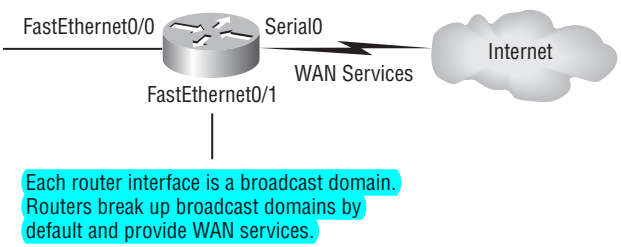


Interface The exit interface a packet will take when destined for a specific network.

Metric The distance to the remote network. Different routing protocols use different ways of computing this distance. I'm going to cover routing protocols in Chapters 6 and 7, but for now, know that some routing protocols (namely RIP) use something called a *hop count* (the number of routers a packet passes through en route to a remote network), while others use bandwidth, delay of the line, or even tick count (1/18 of a second).

And as I mentioned earlier, routers break up broadcast domains, which means that by default, broadcasts aren't forwarded through a router. Do you remember why this is a good thing? Routers also break up collision domains, but you can also do that using layer 2 (Data Link layer) switches. Because each interface in a router represents a separate network, it must be assigned unique network identification numbers, and each host on the network connected to that router must use the same network number. Figure 1.14 shows how a router works in an internetwork.

FIGURE 1.14 A router in an internetwork

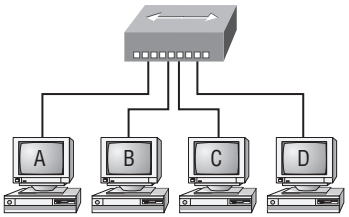


Here are some points about routers that you should really commit to memory:

- Routers, by default, will not forward any broadcast or multicast packets.
- Routers use the logical address in a Network layer header to determine the next hop router to forward the packet to.
- Routers can use access lists, created by an administrator, to control security on the types of packets that are allowed to enter or exit an interface.
- Routers can provide layer 2 bridging functions if needed and can simultaneously route through the same interface.
- Layer 3 devices (routers in this case) provide connections between virtual LANs (VLANs).
- Routers can provide quality of service (QoS) for specific types of network traffic.

Switching and VLANs and are covered in Chapter 8, "LAN Switching and STP," and Chapter 9, "Virtual LANs (VLANs)."

FIGURE 1.17 A hub in a network



All devices in the same collision domain.
All devices in the same broadcast domain.
Devices share the same bandwidth.

Hubs, like repeaters, don't examine any of the traffic as it enters and is then transmitted out to the other parts of the physical media. Every device connected to the hub, or hubs, must listen if a device transmits. A physical star network—where the hub is a central device and cables extend in all directions out from it—is the type of topology a hub creates. Visually, the design really does resemble a star, whereas Ethernet networks run a logical bus topology, meaning that the signal has to run through the network from end to end.

Hubs and repeaters can be used to enlarge the area covered by a single LAN segment, although I do not recommend this. LAN switches are affordable for almost every situation.

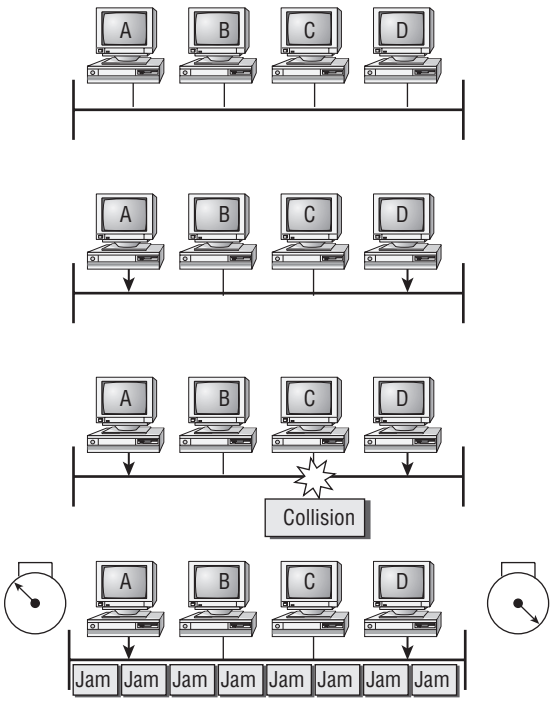
Ethernet Networking

Ethernet is a contention media access method that allows all hosts on a network to share the same bandwidth of a link. Ethernet is popular because it's readily scalable, meaning that it's comparatively easy to integrate new technologies, such as Fast Ethernet and Gigabit Ethernet, into an existing network infrastructure. It's also relatively simple to implement in the first place, and with it, troubleshooting is reasonably straightforward. Ethernet uses both Data Link and Physical layer specifications, and this section of the chapter will give you both the Data Link layer and Physical layer information you need to effectively implement, troubleshoot, and maintain an Ethernet network.

Ethernet networking uses *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)*, a protocol that helps devices share the bandwidth evenly without having two devices transmit at the same time on the network medium. CSMA/CD was created to overcome the problem of those collisions that occur when packets are transmitted simultaneously from different nodes. And trust me—good collision management is crucial, because when a node transmits in a CSMA/CD network, all the other nodes on the network receive and examine that transmission. Only bridges and routers can effectively prevent a transmission from propagating throughout the entire network!

So, how does the CSMA/CD protocol work? Let's start by taking a look at Figure 1.18.

FIGURE 1.18 CSMA/CD



Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

When a host wants to transmit over the network, it first checks for the presence of a digital signal on the wire. If all is clear (no other host is transmitting), the host will then proceed with its transmission. But it doesn't stop there. The transmitting host constantly monitors the wire to make sure no other hosts begin transmitting. If the host detects another signal on the wire, it sends out an extended jam signal that causes all nodes on the segment to stop sending data (think busy signal). The nodes respond to that jam signal by waiting a while before attempting to transmit again. Backoff algorithms determine when the colliding stations can retransmit. If collisions keep occurring after 15 tries, the nodes attempting to transmit will then timeout. Pretty clean!

When a collision occurs on an Ethernet LAN, the following happens:

- A jam signal informs all devices that a collision occurred.
- The collision invokes a random backoff algorithm.
- Each device on the Ethernet segment stops transmitting for a short time until the timers expire.
- All hosts have equal priority to transmit after the timers have expired.