USN | | | | | | | | | | |

Internal Assessment Test 1 – October. 2022

Sub:	Computer Networks					Sub Code:	18EC71	Branch	ECE
Date :	20-10-2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII (A,B,C,D)	OBE	

Answer any five full questions

		MARKS	CO	RBT
1	Describe ISO-OSI reference model of computer network. Discuss the function of each layer	[5+5]	CO1	L1
2	Explain stop and wait protocol. Also discuss acknowledgement, timer and sequence no with the help of flow diagram.	[5+5]	CO2	L3
3	(a) Compare TCP/IP and OSI reference model. (b) Explain encapsulation and de-capsulation with neat figure.	[6+4]	CO1	L2
4	Explain Address Resolution Protocol (ARP) operation. Explain ARP request and response packet format with neat diagrams.	[5+5]	CO2	L3
5	Define link layer addressing with suitable example	[10]	CO1	L2
6	What is framing? Explain bit and character stuffing with help of example.	[10]	CO2	L2
7	(a) Explain the major components of data communication with a neat diagram. (b) Explain network topologies with figures, also mention merits and demerits.	[5+5]	CO1	L2

Course Instructor **Chief Course Instructor** **HOD**

USN | | | | | | | | | | |

Internal Assessment Test 1 – October. 2022

Sub:	Computer Networks					Sub Code:	18EC71	Branch	ECE
Date :	20-10-2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII (A,B,C,D)	OBE	

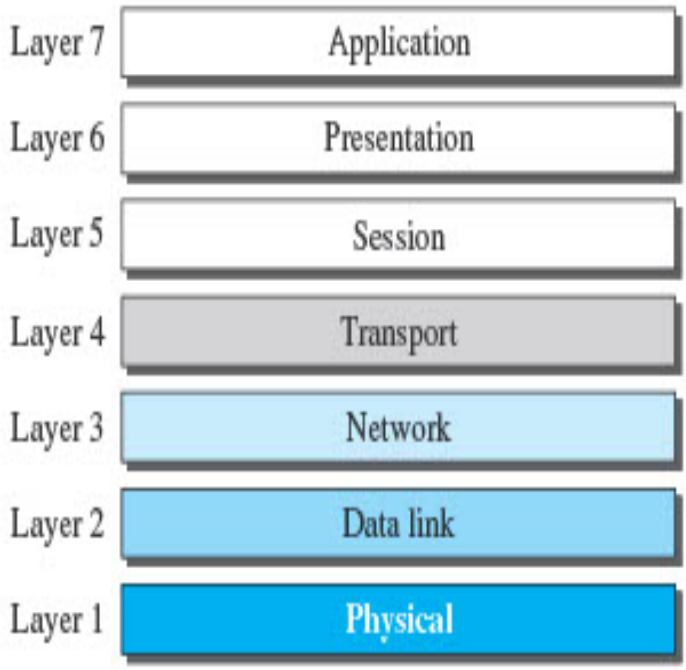
Answer any five full questions

		MARKS	CO	RBT
1	Describe ISO-OSI reference model of computer network. Discuss the function of each layer	[5+5]	CO1	L1
2	Explain stop and wait protocol. Also discuss acknowledgement, timer and sequence no with the help of flow diagram.	[5+5]	CO2	L3
3	(c) Compare TCP/IP and OSI reference model. (d) Explain encapsulation and de-capsulation with neat figure.	[6+4]	CO1	L2
4	Explain Address Resolution Protocol (ARP) operation. Explain ARP request and response packet format with neat diagrams.	[5+5]	CO2	L3
5	(a) Define link layer addressing with suitable example	[10]	CO1	L2
6	What is framing? Explain bit and character stuffing with help of example.	[10]	CO2	L2
7	(a) Explain the major components of data communication with a neat diagram. (b) Explain network topologies with figures, also mention merits and demerits.	[5+5]	CO1	L2

Q1) Solution :

Although, when speaking of the Internet, everyone talks about the TCP/IP protocol suite, this suite is not the only suite of protocols defined.
Established in 1947, the International Organization for Standardization (ISO) is a multinational body dedicated to worldwide agreement on international standards.
Almost three-fourths of the countries in the world are represented in the ISO.
An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model.
It was first introduced in the late 1970s.
ISO is the organization; OSI is the model.
An open system is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture.
The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software.
The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable.
The OSI model was intended to be the basis for the creation of the protocols in the OSI stack.
The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems.
It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network (see Figure 2.11).

Figure 2.11 *The OSI model*



Physical Layer

We can say that the physical layer is responsible for carrying individual bits in a frame across the link.
Although the physical layer is the lowest level in the TCP/IP protocol suite, the communication between two devices at the physical layer is still a logical communication because there is another, hidden layer, the transmission media, under the physical layer.
Two devices are connected by a transmission medium (cable or air).
We need to know that the transmission medium does not carry bits; it carries electrical or optical signals.
So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a bit.
There are several protocols that transform a bit to a signal.

Data-link Layer

We have seen that an internet is made up of several links (LANs and WANs) connected by routers.
There may be several overlapping sets of links that a datagram can travel from the host to the destination.
The routers are responsible for choosing the best links.
However, when the next link to travel is determined by the router, the data-link layer is responsible for taking the datagram and moving it across the link.
The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN.
We can also have different protocols used with any link type.
In each case, the data-link layer is responsible for moving the packet through the link.
TCP/IP does not define any specific protocol for the data-link layer.
It supports all the standard and proprietary protocols.
Any protocol that can take the datagram and carry it through the link suffices for the network layer.
The data-link layer takes a datagram and encapsulates it in a packet called a frame.
Each link-layer protocol may provide a different service.
Some link-layer protocols provide complete error detection and correction, some provide only error correction.

Network Layer

The network layer is responsible for creating a connection between the source computer and the destination computer.
The communication at the network layer is host-to-host.
However, since there can be several routers from the source to the destination, the routers in the path are responsible for choosing the best route for each packet.
We can say that the network layer is responsible for host-to-host communication and routing the packet through possible routes.
Again, we may ask ourselves why we need the network layer. We could have added the routing duty to the transport layer and dropped this layer.
One reason, as we said before, is the separation of different tasks between different layers. The second reason is that the routers do not need the application and transport layers. Separating the tasks allows us to use fewer protocols on the routers.
The network layer in the Internet includes the main protocol, Internet Protocol (IP), that defines the format of the packet, called a datagram at the network layer.
IP also defines the format and the structure of addresses used in this layer.
IP is also responsible for routing a packet from its source to its destination, which is achieved by each router forwarding the datagram to the next router in its path.
IP is a connectionless protocol that provides no flow control, no error control, and no congestion control services.
This means that if any of these services is required for an application, the application should rely only on the transport-layer protocol.
The network layer also includes unicast (one-to-one) and multicast (one-to-many) routing protocols.
A routing protocol does not take part in routing (it is the responsibility of IP), but it creates forwarding tables for routers to help them in the routing process.
The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks.
The Internet Control Message Protocol (ICMP) helps IP to report some problems when routing a packet.
The Internet Group Management Protocol (IGMP) is another protocol that helps IP in multitasking.

The Dynamic Host Configuration Protocol (DHCP) helps IP to get the network-layer address for a host.
The Address Resolution Protocol (ARP) is a protocol that helps IP to find the link-layer address of a host or a router when its network-layer address is given.

Transport Layer

The logical connection at the transport layer is also end-to-end.
The transport layer at the source host gets the message from the application layer, encapsulates it in a transport layer packet (called a segment or a user datagram in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host.
In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host.
We may ask why we need an end-to-end transport layer when we already have an end-to-end application layer. The reason is the separation of tasks and duties, which we discussed earlier.
The transport layer should be independent of the application layer.
In addition, we will see that we have more than one protocol in the transport layer, which means that each application program can use the protocol that best matches its requirement.
The main protocol, Transmission Control Protocol (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data.
It creates a logical pipe between two TCPs for transferring a stream of bytes.
TCP provides flow control (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination), error control (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and congestion control to reduce the loss of segments due to congestion in the network.
The other common protocol, User Datagram Protocol (UDP), is a connectionless protocol that transmits user datagrams without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one (the meaning of the term connectionless).
UDP is a simple protocol that does not provide flow, error, or congestion control. Its simplicity, which means small overhead, is attractive to an application program that needs to send short messages and cannot afford the retransmission of the packets involved in TCP, when a packet is corrupted or lost.
A new protocol, Stream Control Transmission Protocol (SCTP) is designed to respond to new applications that are emerging in the multimedia.

Application Layer

As Figure 2.6 shows, the logical connection between the two application layers is end to-end.
The two application layers exchange messages between each other as though there were a bridge between the two layers.
However, we should know that the communication is done through all the layers.
Communication at the application layer is between two processes (two programs running at this layer).
To communicate, a process sends a request to the other process and receives a response.
Process-to-process communication is the duty of the application layer.
The application layer in the Internet includes many predefined protocols, but a user can also create a pair of processes to be run at the two hosts.
The Hypertext Transfer Protocol (HTTP) is a vehicle for accessing the World Wide Web (WWW).
The Simple Mail Transfer Protocol (SMTP) is the main protocol used in electronic mail (e-mail) service.
The File Transfer Protocol (FTP) is used for transferring files from one host to another.
The Terminal Network (TELNET) and Secure Shell (SSH) are used for accessing a site remotely.
The Simple Network Management Protocol (SNMP) is used by an administrator to manage the Internet at global and local levels.
The Domain Name System (DNS) is used by other protocols to find the network-layer address of a computer.
The Internet Group Management Protocol (IGMP) is used to collect membership in a group.

Q2)Solution :

Our second protocol is called the Stop-and-Wait protocol, which uses both flow and error control.
In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one.
To detect corrupted frames, we need to add a CRC to each data frame.
When a frame arrives at the receiver site, it is checked.
If its CRC is incorrect, the frame is corrupted and silently discarded.
The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.
Every time the sender sends a frame, it starts a timer.
If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send).
If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.
This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready.
Figure 11.10 shows the outline for the Stop-and-Wait protocol.
Note that only one frame and one acknowledgment can be in the channels at any time.

Figure 11.10 *Stop-and-Wait protocol*

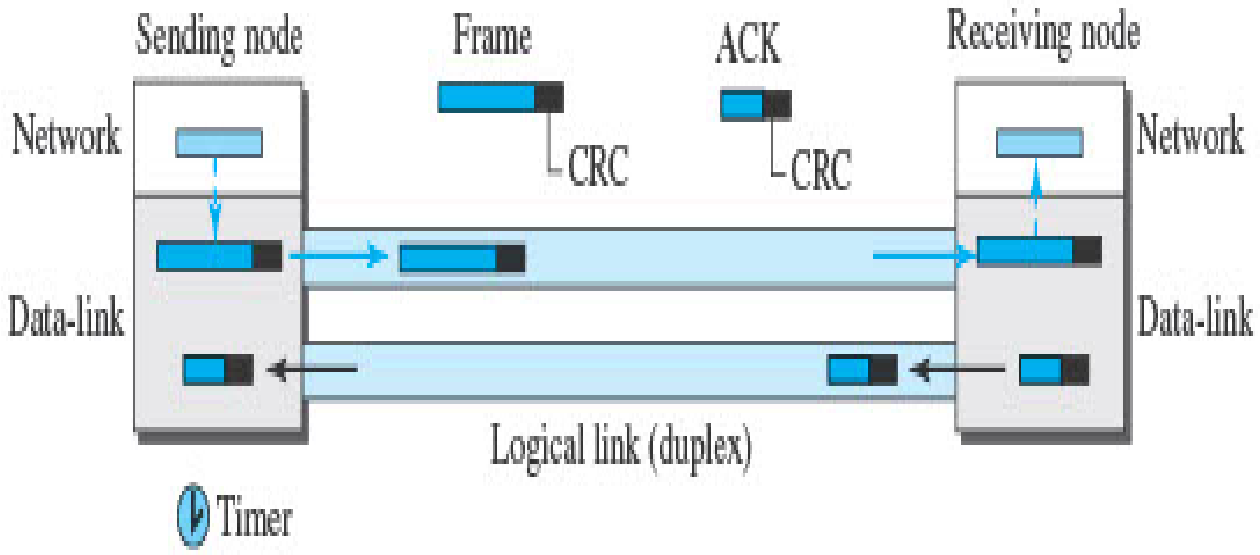
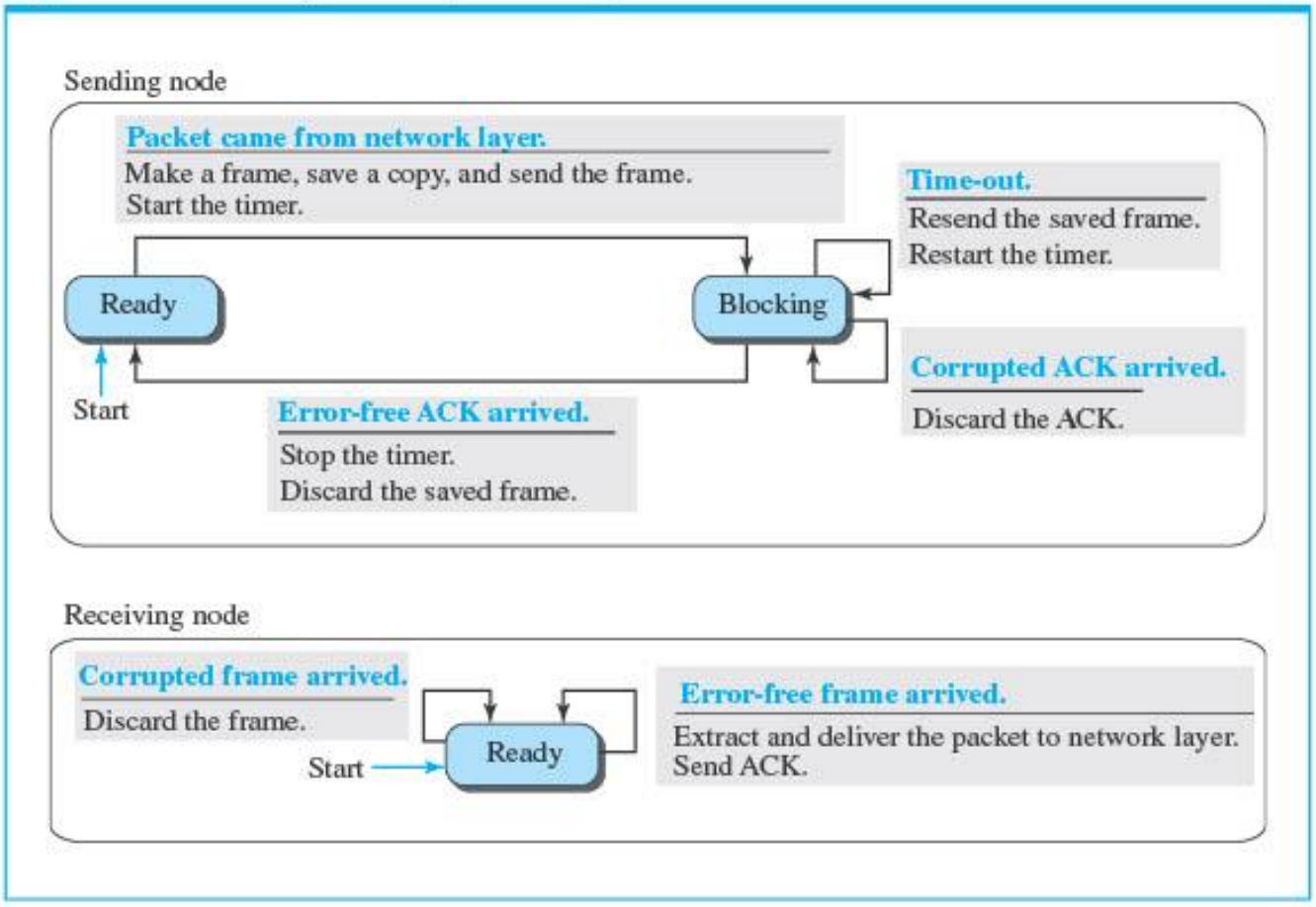


Figure 11.11 shows the FSMs for our primitive Stop-and-Wait protocol.
We describe the sender and receiver states below.

Figure 11.11 *FSM for the Stop-and-Wait protocol*



Sender States
The sender is initially in the ready state, but it can move between the ready and blocking state.
Ready State: When the sender is in this state, it is only waiting for a packet from the network layer.
If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame.
The sender then moves to the blocking state.

Blocking State: When the sender is in this state, three events can occur:
If a time-out occurs, the sender resends the saved copy of the frame and restarts the timer.
If a corrupted ACK arrives, it is discarded.
If an error-free ACK arrives, the sender stops the timer and discards the saved copy of the frame.
It then moves to the ready state.

Receiver
The receiver is always in the ready state.
Two events may occur:
If an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.

If a corrupted frame arrives, the frame is discarded.

Figure 11.12 shows an example.
The first frame is sent and acknowledged.
The second frame is sent, but lost. After time-out, it is resent.
The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent.
However, there is a problem with this scheme.
The network layer at the receiver site receives two copies of the third packet, which is not right.
In the next section, we will see how we can correct this problem using sequence numbers and acknowledgment numbers.

We saw a problem in Example 11.3 that needs to be addressed and corrected.
Duplicate packets, as much as corrupted packets, need to be avoided.
As an example, assume we are ordering some item online.
If each packet defines the specification of an item to be ordered, duplicate packets mean ordering an item more than once.
To correct the problem in Example 11.3, we need to add sequence numbers to the data frames and acknowledgment numbers to the ACK frames.
However, numbering in this case is very simple. Sequence numbers are 0, 1, 0, 1, 0, 1, . . . ; the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, … In other words, the sequence numbers start with 0, the acknowledgment numbers start with 1.
An acknowledgment number always defines the sequence number of the next frame to receive.

Figure 11.12 *Flow diagram for Example 11.3*

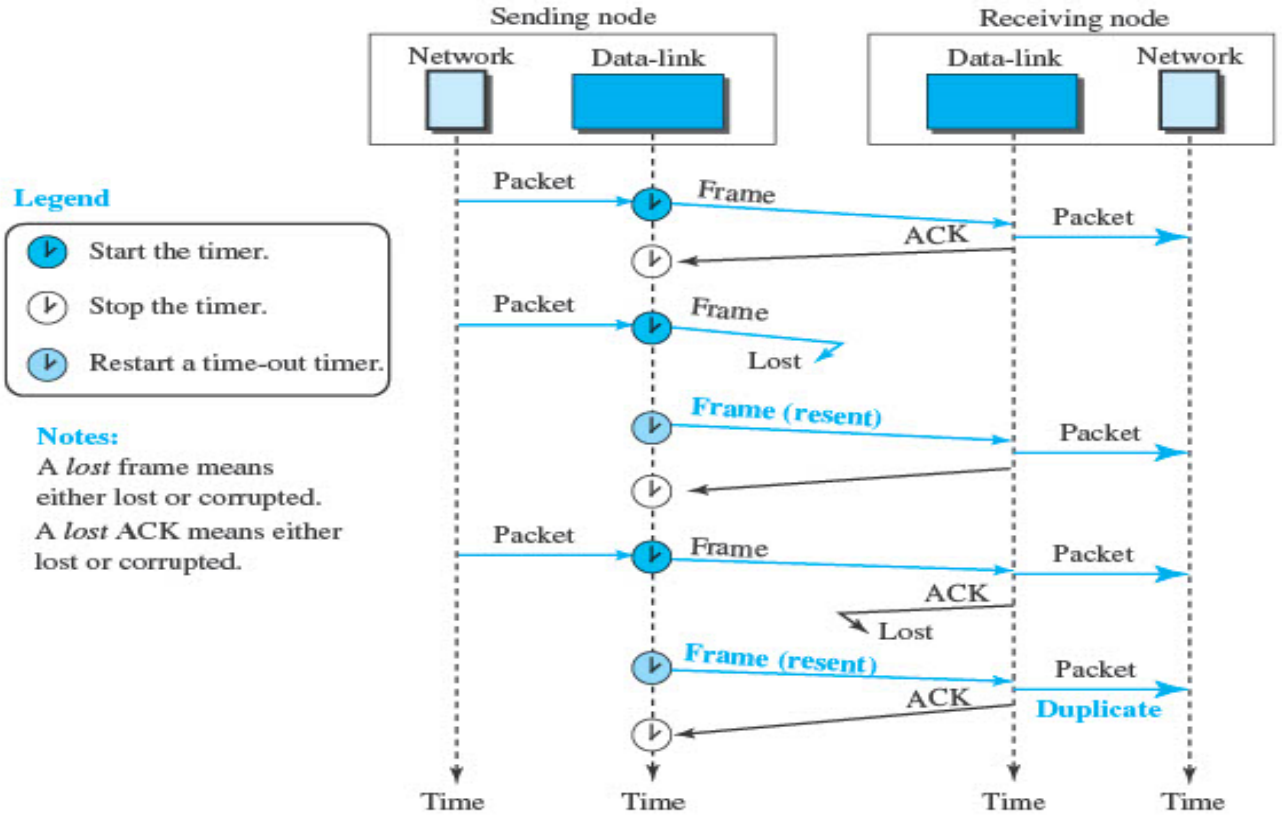
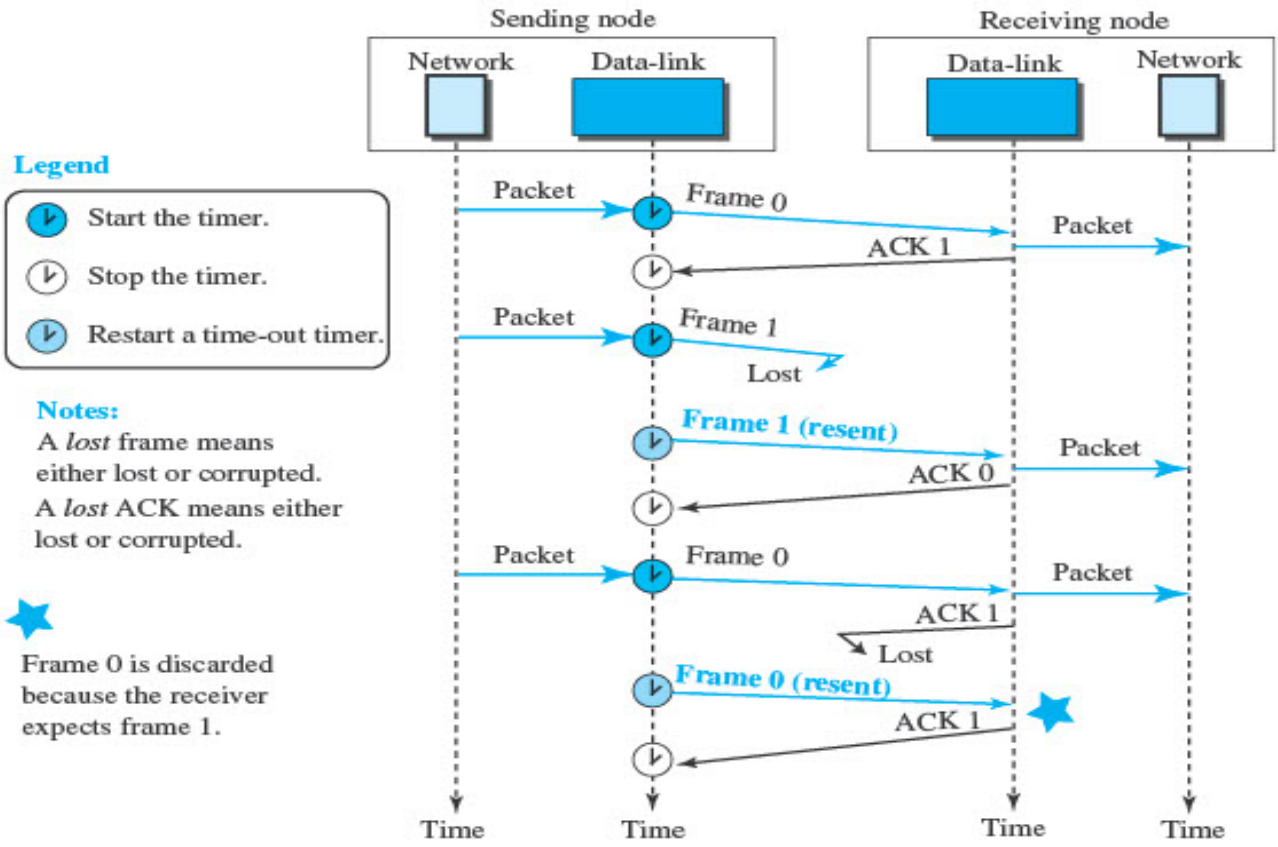


Figure 11.13 shows how adding sequence numbers and acknowledgment numbers can prevent duplicates.
The first frame is sent and acknowledged.
The second frame is sent, but lost. After time-out, it is resent.
The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent.

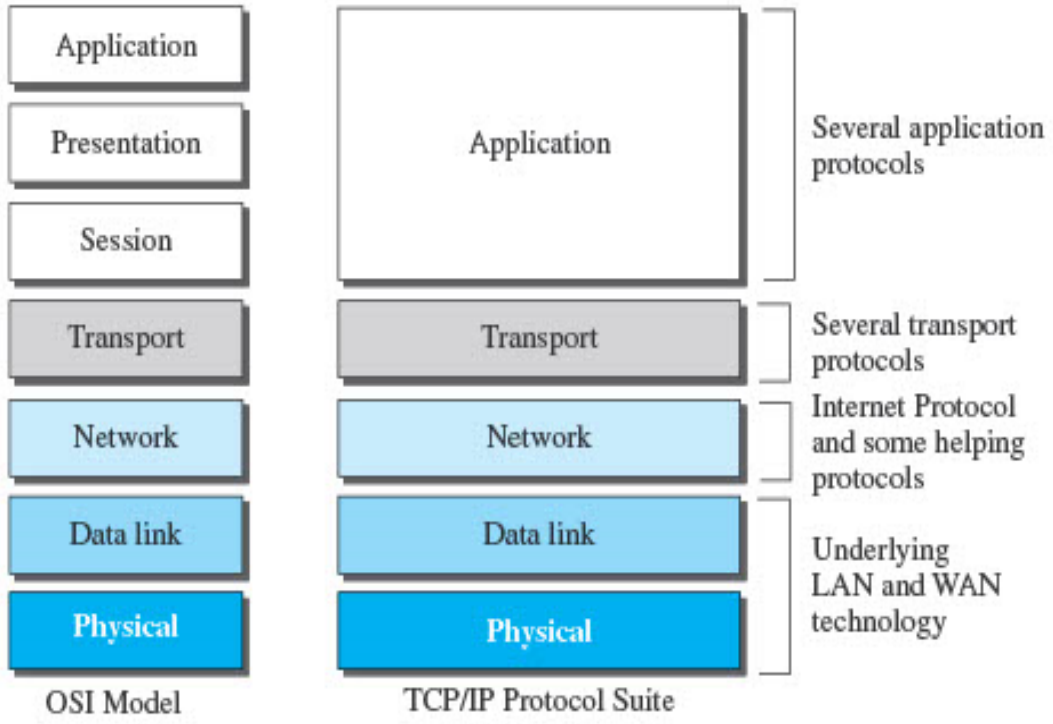
Figure 11.13 *Flow diagram for Example 11.4*



Q3)a Solution :

When we compare the two models, we find that two layers, session and presentation, are missing from the TCP/IP protocol suite.
These two layers were not added to the TCP/IP protocol suite after the publication of the OSI model.
The application layer in the suite is usually considered to be the combination of three layers in the OSI model, as shown in Figure 2.12.

Figure 2.12 *TCP/IP and OSI model*

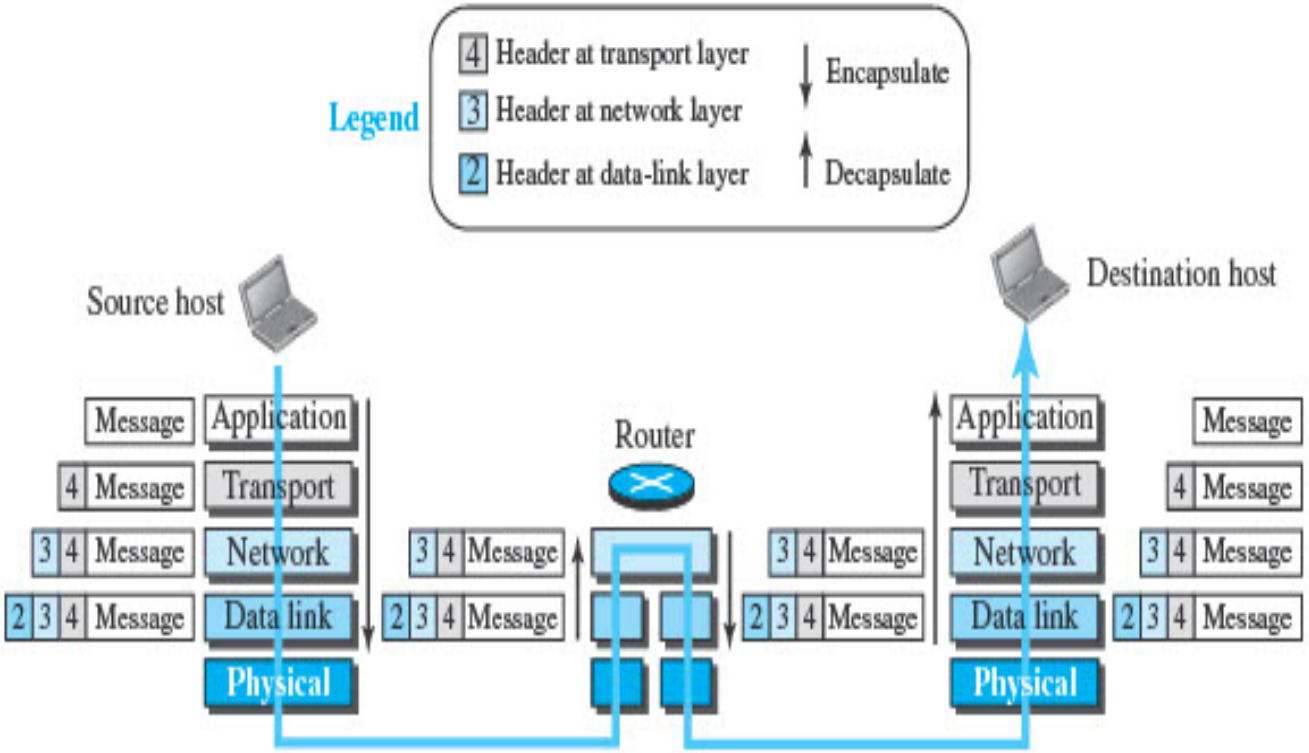


Two reasons were mentioned for this decision.
First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols.
Second, the application layer is not only one piece of software.
Many applications can be developed at this layer.
If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

Q3b) Solution :

One of the important concepts in protocol layering in the Internet is encapsulation/ decapsulation.
Figure 2.8 shows this concept for the small internet in Figure 2.5.
We have not shown the layers for the link-layer switches because no encapsulation/ decapsulation occurs in this device.
In Figure 2.8, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

Figure 2.8 *Encapsulation/Decapsulation*



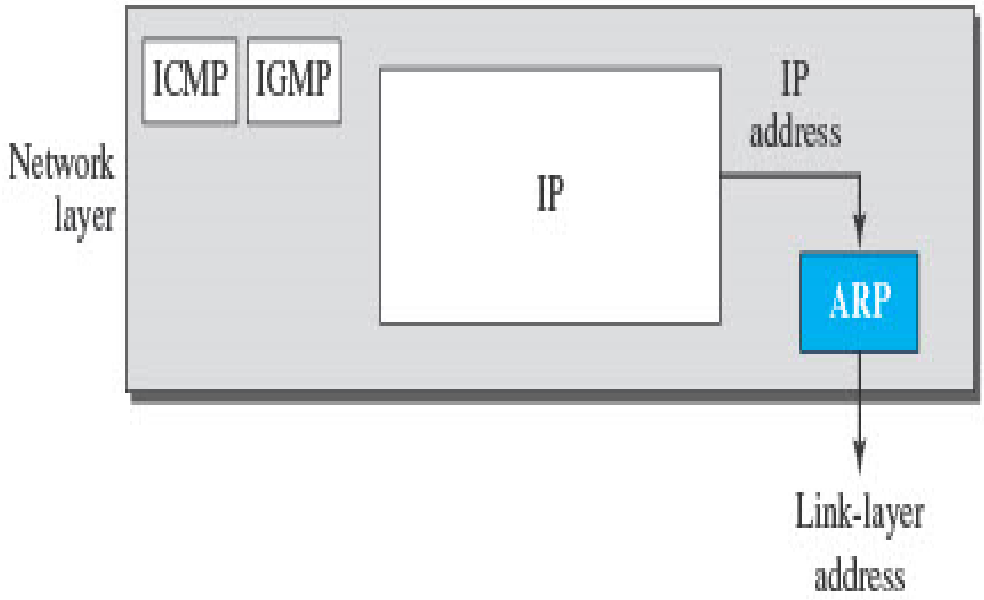
At the source, we have only encapsulation.
At the application layer, the data to be exchanged is referred to as a message. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the message. The message is passed to the transport layer.
The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that want to communicate plus some more information that is needed for the end-to end delivery of the message, such as information needed for flow, error control, or congestion control. The result is the transport-layer packet, which is called the segment (in TCP) and the user datagram (in UDP). The transport layer then passes the packet to the network layer.
The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for error checking of the header, fragmentation information, and so on. The result is the network-layer packet, called a datagram. The network layer then passes the packet to the data-link layer.
The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a frame. The frame is passed to the physical layer for transmission.
At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.
After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.

The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.
The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.
At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer.
It is necessary to say that decapsulation in the host involves error checking.

Q4) Solution :

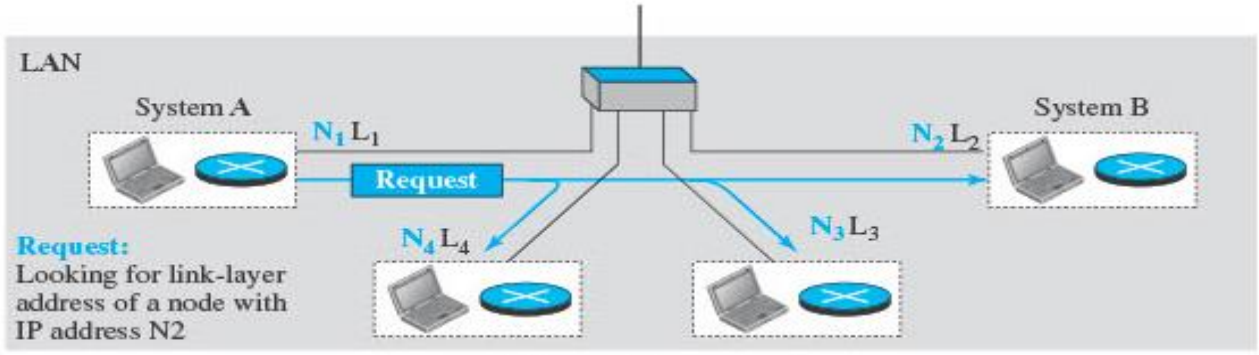
Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node.
The source host knows the IP address of the default router.
Each router except the last one in the path gets the IP address of the next router by using its forwarding table.
The last router knows the IP address of the destination host.
However, the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node.
This is the time when the Address Resolution Protocol (ARP) becomes helpful.
The ARP protocol is one of the auxiliary protocols defined in the network layer, as shown in Figure 9.6.
It belongs to the network layer, but we discuss it in this chapter because it maps an IP address to a logical-link address.
ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

Figure 9.6 *Position of ARP in TCP/IP protocol suite*

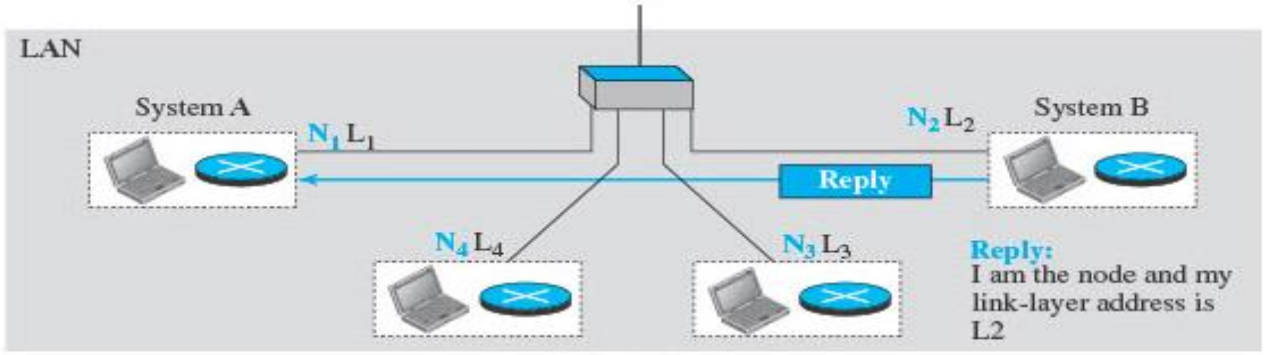


Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet.
The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver.
Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address, which we discuss for each protocol later (see Figure 9.7).

Figure 9.7 *ARP operation*



a. ARP request is broadcast



b. ARP reply is unicast

Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet.
The response packet contains the recipient's IP and link-layer addresses.
The packet is unicast directly to the node that sent the request packet.
In Figure 9.7a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address N2.
System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient.
It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of N2.
This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 9.7b.
System B sends an ARP reply packet that includes its physical address.
Now system A can send all the packets it has for this destination using the physical address it received.

A question that is often asked is this: If system A can broadcast a frame to find the linklayer address of system B, why can't system A send the datagram for system B using a broadcast frame?
In other words, instead of sending one broadcast frame (ARP request), one unicast frame (ARP response), and another unicast frame (for sending the datagram), system A can encapsulate the datagram and send it to the network.
System B receives it and keep it; other systems discard it.

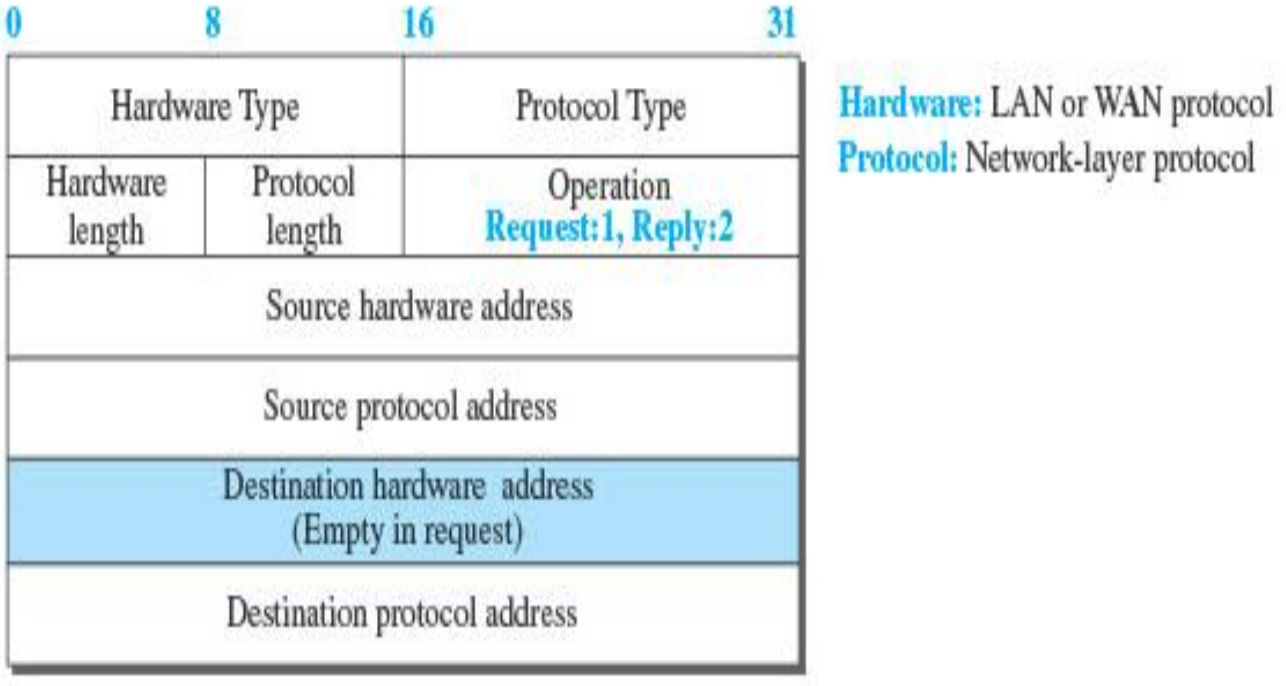
To answer the question, we need to think about the efficiency. It is probable that system A has more than one datagram to send to system B in a short period of time.
For example, if system B is supposed to receive a long e-mail or a long file, the data do not fit in one datagram.
Let us assume that there are 20 systems connected to the network (link): system A, system B, and 18 other systems. We also assume that system A has 10 datagrams to send to system B in one second.

Without using ARP, system A needs to send 10 broadcast frames.
Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the datagram and pass it to their network-layer to find out the datagrams do not belong to them.
This means processing and discarding 180 broadcast frames.

Using ARP, system A needs to send only one broadcast frame.
Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the ARP message and pass the message to their ARP protocol to find that the frame must be discarded.
This means processing and discarding only 18 (instead of 180) broadcast frames.
After system B responds with its own data-link address, system A can store the link-layer address in its cache memory.
The rest of the nine frames are only unicast.
Since processing broadcast frames is expensive (time consuming), the first method is preferable.

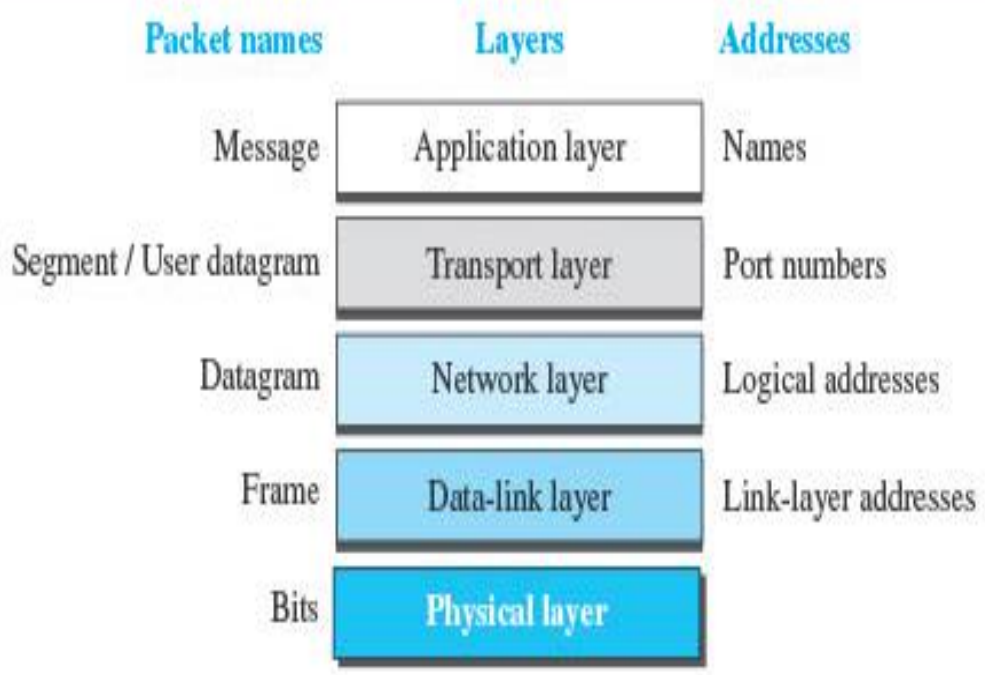
Figure 9.8 shows the format of an ARP packet.
The names of the fields are self explanatory.
The hardware type field defines the type of the link-layer protocol; Ethernet is given the type 1.
The protocol type field defines the network-layer protocol: IPv4 protocol is (0800)16.
The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender.
The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses.
An ARP packet is encapsulated directly into a data-link frame.
The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

Figure 9.8 *ARP packet*



Q5) Solution:

Figure 2.9 *Addressing in the TCP/IP protocol suite*



It is worth mentioning another concept related to protocol layering in the Internet, addressing.
As we discussed before, we have logical communication between pairs of layers in this model.
Any communication that involves two parties needs two addresses: source address and destination address.
Although it looks as if we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address.
Figure 2.9 shows the addressing at each layer. As the figure shows, there is a relationship between the layer, the address used in that layer, and the packet name at that layer. At the application layer, we normally use names to define the site that provides services, such as someorg.com, or the e-mail address, such as somebody@coldmail.com.

At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination.
Port numbers are local addresses that distinguish between several programs running at the same time.

At the network-layer, the addresses are global, with the whole Internet as the scope.
A network-layer address uniquely defines the connection of a device to the Internet.
The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN).
We will come back to these addresses in future chapters.

Q6) Solution :

Definitely, the first service provided by the data-link layer is framing.
The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a frame before sending it to the next node.
The node also needs to decapsulate the datagram from the frame received on the logical channel.
Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer.
Different data-link layers have different formats for framing.
A packet at the data-link layer is normally called a frame.

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination.
The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.
The data-link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another.
Our postal system practices a type of framing.
The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.
In addition, each envelope defines the sender and receiver addresses, which is necessary since the postal system is a many to-many carrier facility.

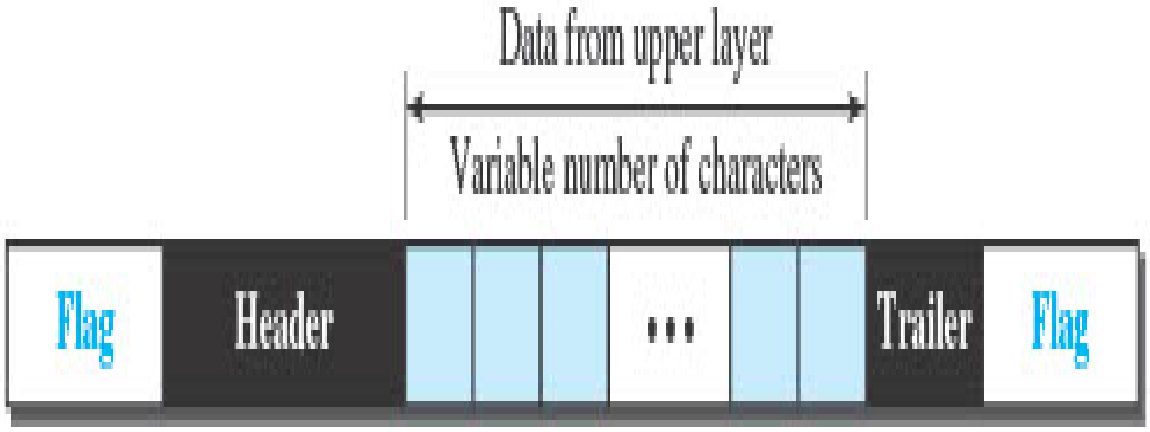
Framing in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address.
The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.
Although the whole message could be packed in one frame, that is not normally done.
One reason is that a frame can be very large, making flow and error control very inefficient.
When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole frame.
When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frames can be of fixed or variable size.
In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter.
An example of this type of framing is the ATM WAN, which uses frames of fixed size called cells.
Our main discussion in this chapter concerns variable-size framing, prevalent in local-area networks.
In variable-size framing, we need a way to define the end of one frame and the beginning of the next.
Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

In character-oriented (or byte-oriented) framing, data to be carried are 8-bit characters from a coding system such as ASCII (see Appendix A).
The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection redundant bits, are also multiples of 8 bits.
To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and the end of a frame.
The flag, composed of protocol-dependent special characters, signals the start or end of a frame.

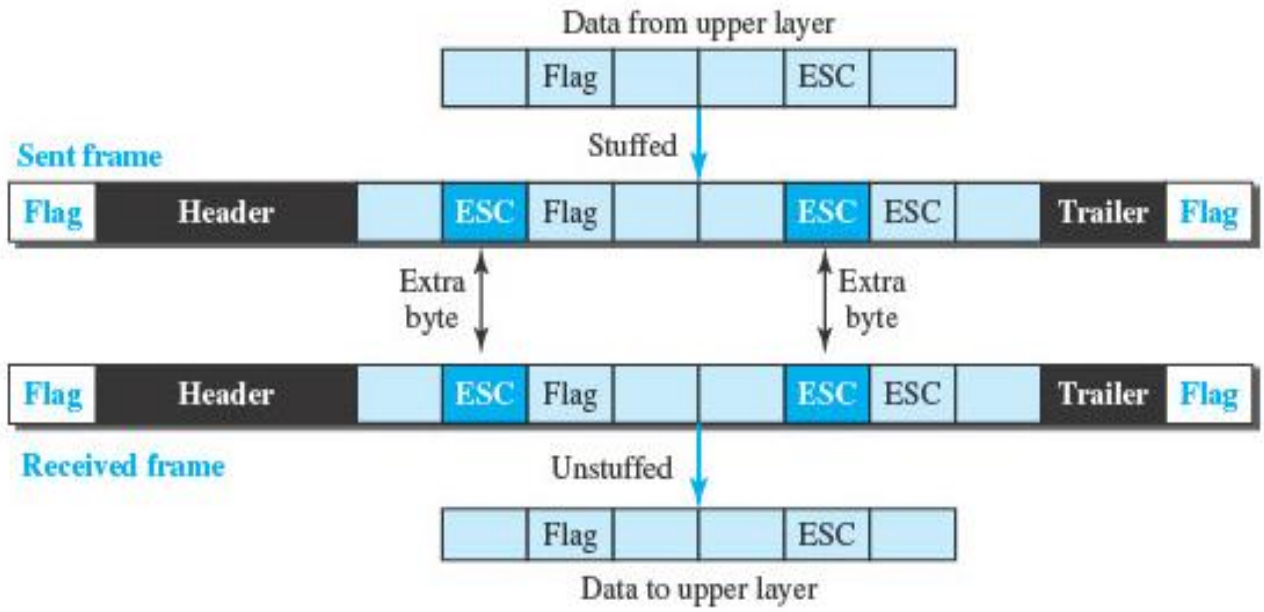
Figure 11.1 shows the format of a frame in a character-oriented protocol.

Figure 11.1 *A frame in a character-oriented protocol*



Character-oriented framing was popular when only text was exchanged by the data-link layers.
The flag could be selected to be any character not used for text communication.
Now, however, we send other types of information such as graphs, audio, and video; any character used for the flag could also be part of the information.
If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame.
To fix this problem, a byte-stuffing strategy was added to character-oriented framing.
In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag.
The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC) and has a predefined bit pattern.
Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag.
Figure 11.2 shows the situation.

Figure 11.2 *Byte stuffing and unstuffing*

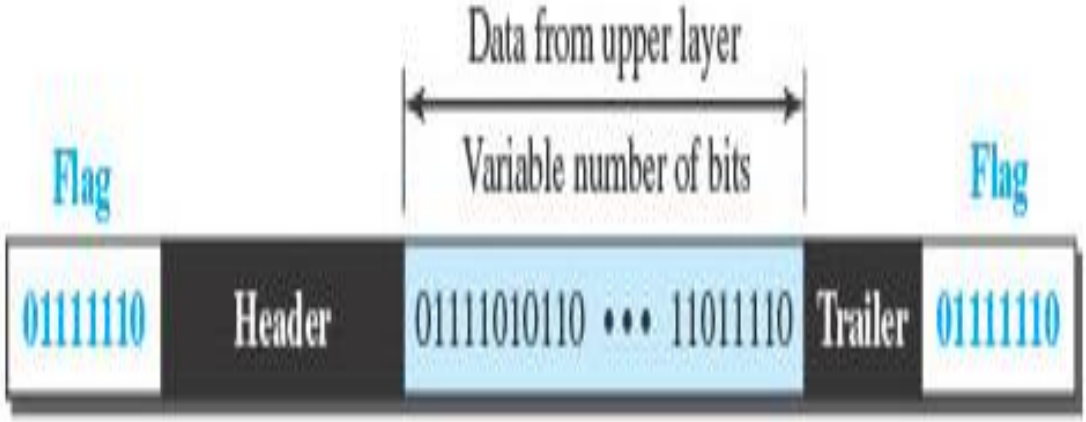


Byte stuffing is the process of adding one extra byte whenever there is a flag or escape character in the text.

Byte stuffing is the process of adding one extra byte whenever there is a flag or escape character in the text.
Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem.
What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag?
The receiver removes the escape character, but keeps the next byte, which is incorrectly interpreted as the end of the frame.
To solve this problem, the escape characters that are part of the text must also be marked by another escape character.
In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.
Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters.
We can say that, in general, the tendency is moving toward the bit-oriented protocols that we discuss next.

In bit-oriented framing, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.
However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other.
Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

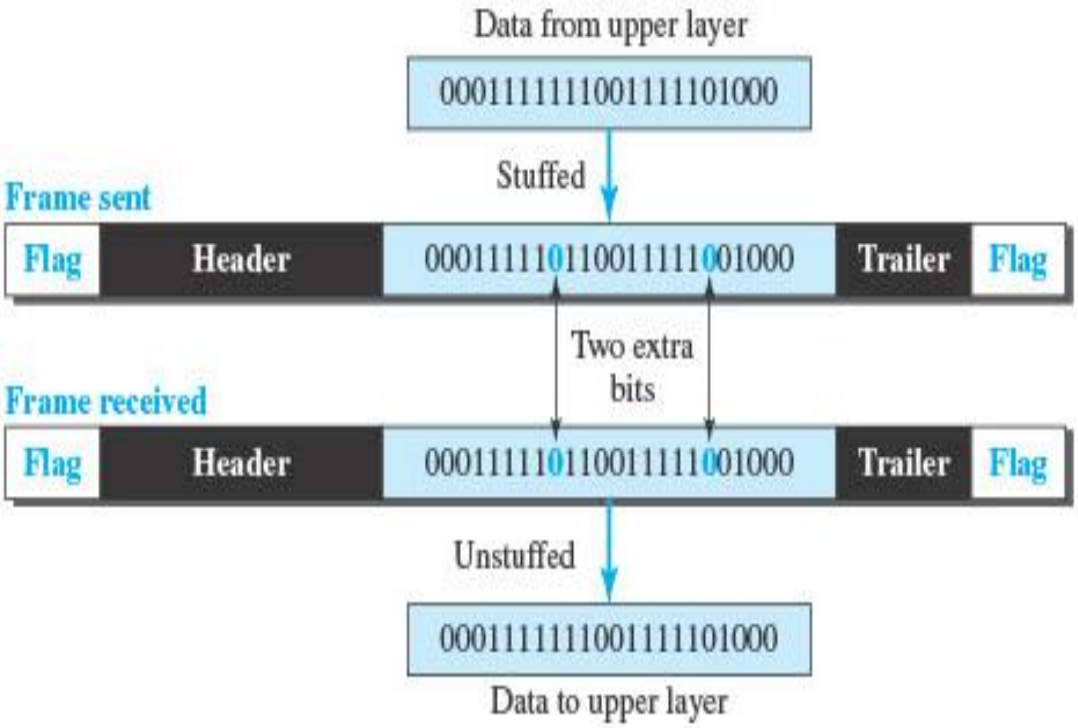
Figure 11.3 *A frame in a bit-oriented protocol*



This flag can create the same type of problem we saw in the character-oriented protocols.
That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame.
We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag.
The strategy is called bit stuffing.
In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added.
This extra stuffed bit is eventually removed from the data by the receiver.
Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.
Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver.
Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.
This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken for a flag by the receiver.
The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

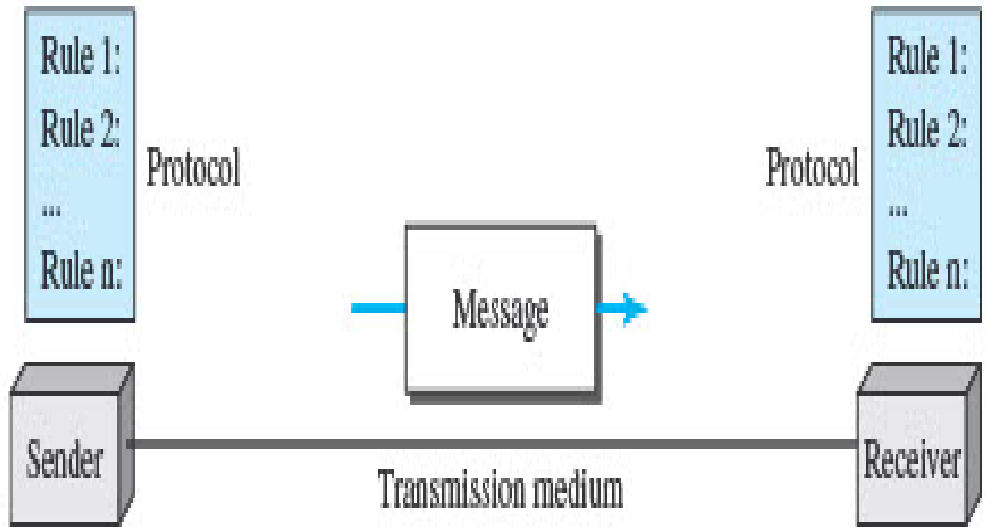
Figure 11.4 *Bit stuffing and unstuffing*



Q7a) Solution :

A data communications system has five components (see Figure 1.1).
Message: The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
Sender: The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
Receiver: The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
Transmission medium: The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
Protocol: A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

Figure 1.1 *Five components of data communication*

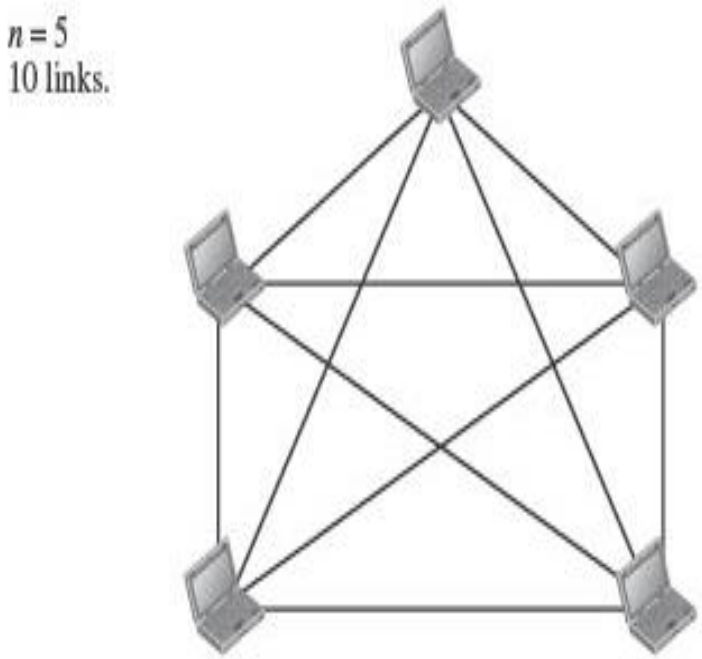


Q7b) Solution :

The term physical topology refers to the way in which a network is laid out physically.
Two or more devices connect to a link; two or more links form a topology.
The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another.
There are four basic topologies possible: mesh, star, bus, and ring.

In a mesh topology, every device has a dedicated point-to-point link to every other device.
The term dedicated means that the link carries traffic only between the two devices it connects.
To find the number of physical links in a fully connected mesh network with n nodes, we first consider that each node must be connected to every other node.
Node 1 must be connected to n – 1 nodes, node 2 must be connected to n – 1 nodes, and finally node n must be connected to n – 1 nodes. We need n (n – 1) physical links.
However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2.
In other words, we can say that in a mesh topology, we need n (n – 1) / 2 duplex-mode links.
To accommodate that many links, every device on the network must have n – 1 input/output (I/O) ports (see Figure 1.4) to be connected to the other n – 1 stations.
One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

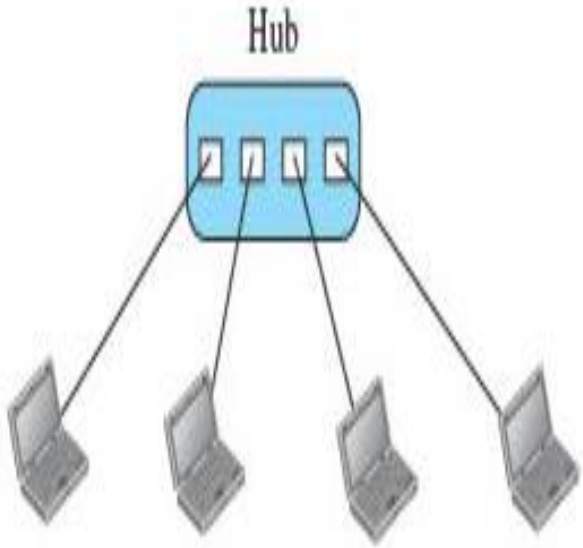
Figure 1.4 *A fully connected mesh topology (five devices)*



In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a hub.
The devices are not directly linked to one another.
Unlike a mesh topology, a star topology does not allow direct traffic between devices.

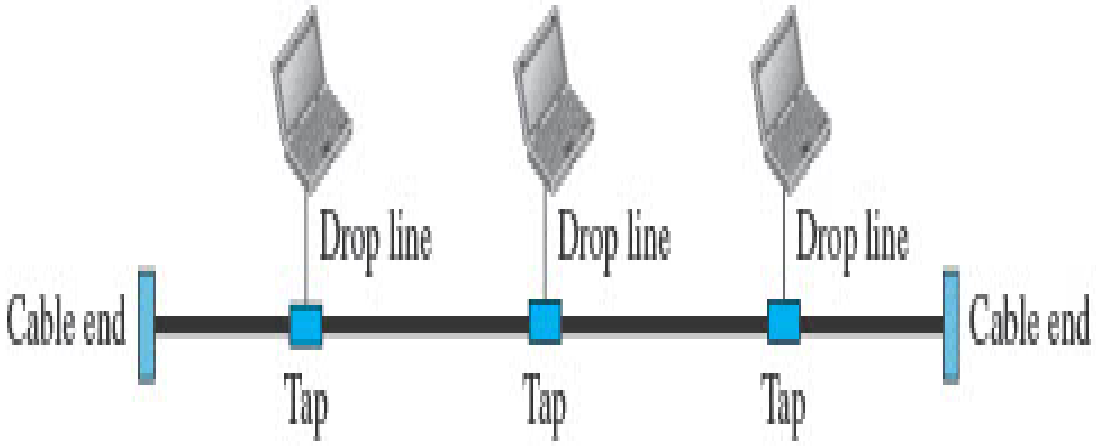
The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device (see Figure 1.5).
The star topology is used in local-area networks (LANs); High-speed LANs often use a star topology with a central hub.

Figure 1.5 *A star topology connecting four stations*



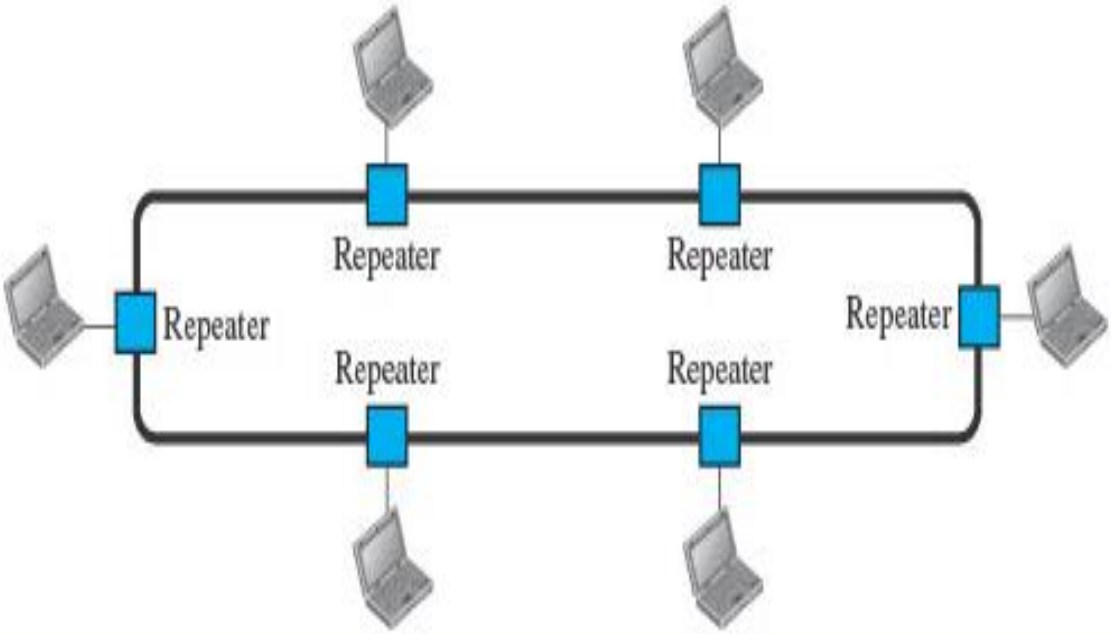
The preceding examples all describe point-to-point connections.
A bus topology, on the other hand, is multipoint.
One long cable acts as a backbone to link all the devices in a network (see Figure 1.6).
Nodes are connected to the bus cable by drop lines and taps.
A drop line is a connection running between the device and the main cable.
A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core.
As a signal travels along the backbone, some of its energy is transformed into heat.
Therefore, it becomes weaker and weaker as it travels farther and farther.
For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.
Bus topology was the one of the first topologies used in the design of early local area networks. Traditional Ethernet LANs can use a bus topology, but they are less popular now.

Figure 1.6 *A bus topology connecting three stations*



In a ring topology, each device has a dedicated point-to-point connection with only the two devices on either side of it.
A signal is passed along the ring in one direction, from device to device, until it reaches its destination.
Each device in the ring incorporates a repeater.
When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along (see Figure 1.7).
Ring topology was prevalent when IBM introduced its local-area network, Token Ring. Today, the need for higher-speed LANs has made this topology less popular.

Figure 1.7 *A ring topology connecting six stations*



*******END*******