CMR INSTITUTE OF TECHNOLOGY
DEPARTMENT OF CSE
VTU QP SOLUTION
BCS502-COMPUTER NETWORKS

MODULE - 1					
1	a.	**Explain fundamental characteristics and data representation in data communication** **SOLUTION** **Data Communication** • Delivery : Deliver the data to correct destination. • Accuracy: Deliver the data accurately. • Timeliness: Deliver the data in timely manner. Like video and audio called real-time transmission. • Jitter: Refers to the variation in packet time arrival. **Data Representation** • Text: Text is represented as a sequence of bits (0 or 1). Process of representing the symbols is coding. [Unicode, ASCII] • Numbers: Also represented by bits (0 or 1). • Images: Represented using bit patterns. Pixels are represented using bits. • Audio: Recording or broadcasting of sound or music. Continuous data that generates continuous signal. • Video: Recording or broadcasting a picture or movie. It can be a continuous or discrete entity.	6	L1	CO1
	b.	**Discuss types of connection and the basic topologies in networks** **SOLUTION**	6	L1	CO1

Physical Topology

- Physical Topology refers to the way in which the network is connected physically.
- Topology of a network is the geometric representation of the nodes.

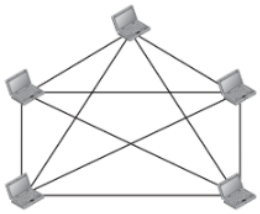
There are four basic topologies:

- Mesh
- Star
- Bus
- Ring

Mesh Topology

- In this topology, every device has a dedicated point-to-point link to every other device.
- Mesh Topology needs 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.

$n = 5$
10 links.

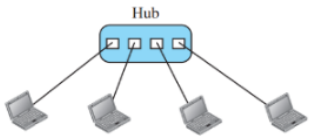


Disadvantages of Mesh Topology

- Amount of cabling and the number of I/O ports required.
- Installation and reconnection are difficult.
- Hardware can be expensive.
- For these reasons a mesh topology is usually implemented in a limited fashion.
- Connection of telephone regional offices

Star Topology

- Each device has a dedicated point-to-point link only to a central controller, usually called a **hub**.
- Star topology does not allow direct traffic between devices.
- The controller acts as an exchange.



Star Topology

- A star topology is less expensive than a mesh topology.
- In a star, each device needs only one link and one I/O port.
- Robust : If one link fails, only that link is affected. All other links remain active.
- One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead.

Bus Topology

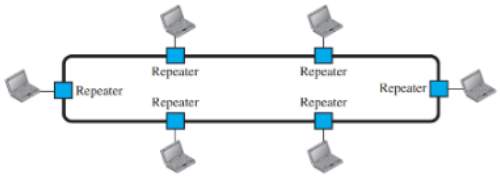
- A Bus topology is a multi-point connection.
- One long cable acts as a backbone to link all the devices in a network.
- Nodes are connected to the bus cable by drop lines and taps.
- A drop line is a connection running between the device and the main cable.

Disadvantages : Bus Topology

- Signal reflection at the taps can cause degradation in quality.
- Adding new devices may therefore require modification or replacement of the backbone.
- In addition, a fault or break in the bus cable stops all transmission.

Ring Topology

- 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.



Ring Topology

- A ring is relatively easy to install and reconfigure.
- Each device is linked to only its immediate neighbors.
- To add or delete a device requires changing only two connections.

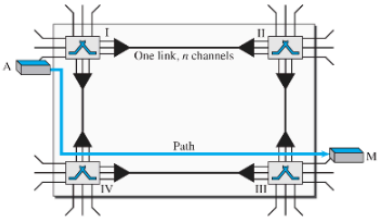
Disadvantages Ring Topology

- Unidirectional traffic can be a disadvantage.
- In a simple ring, a break in the ring can disable the entire network.
- This weakness can be solved by using a dual ring or a switch capable of closing off the break

	c.	**Explain packet switching and circuit switching with neat diagrams** **SOLUTION**	**8**	**L1**	**CO1**

Circuit Switched Networks

Figure 8.3 *A trivial circuit-switched network*

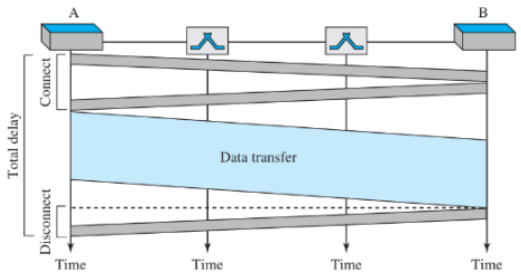


Communication happens via

- **Set Up Phase**
- **Data – transfer Phase**
- **Tear down phase**
- Data is not packetized. No addressing is involved. Happens in Physical Layer.
- Resources : Bandwidth, timeslots, switch buffers, switch processing time, switch input and output ports.

In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer until the teardown phase.

Figure 8.6 *Delay in a circuit-switched network*



Packet Switching

- Messages are divided into packets of fixed or variable size if it is passing through a packet-switched network.
- Size of packet is determined by the network and the governing protocol.
- Packets should wait if other packets are being processed.
- No resource allocation for packet. Resources are allocated on demand.

Packet Switching

- The allocation is done on a first come, first-served basis.
- When a switch receives a packet, the packet must wait if there are other packets being processed.
- Might create delay.

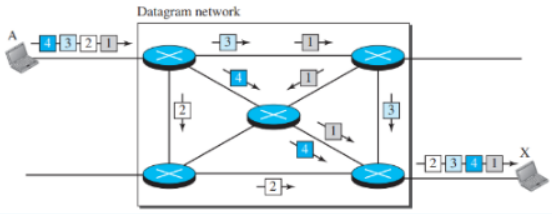
Types of Packet-switched networks

- Datagram networks
- Virtual circuit networks

Datagram Networks

- In datagram network, each packet is treated independently of all others.
- Packets in this approach are referred to as **datagrams**.
- Datagram switching is normally done at the network layer.
- Datagram networks are sometimes referred to **as connectionless networks**.

Figure 8.7 *A datagram network with four switches (routers)*



OR

2	**a.**	**Explain the layers of TCP/IP Protocol Suite.** **SOLUTION**	**8**	**L2**	**CO2**

TCP/IP Protocol Suite

- Transmission Control Protocol/Internet Protocol
- TCP/IP is a protocol suite used in Internet today.
- It is a hierarchical protocol made up of interactive modules, each of which provides a specific functionality.
- Hierarchical means that each upper level protocol is supported by services provided by one or more lower level protocols.
- Initially TCP/IP has four software layers built upon the hardware, now it is 5 layers

TCP/IP : Layered architecture

Figure 2.4 *Layers in the TCP/IP protocol suite*

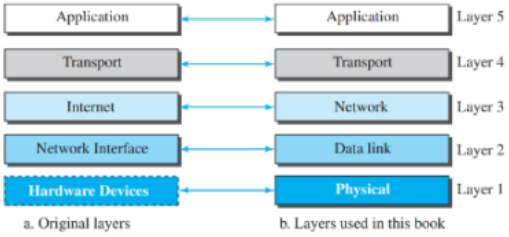
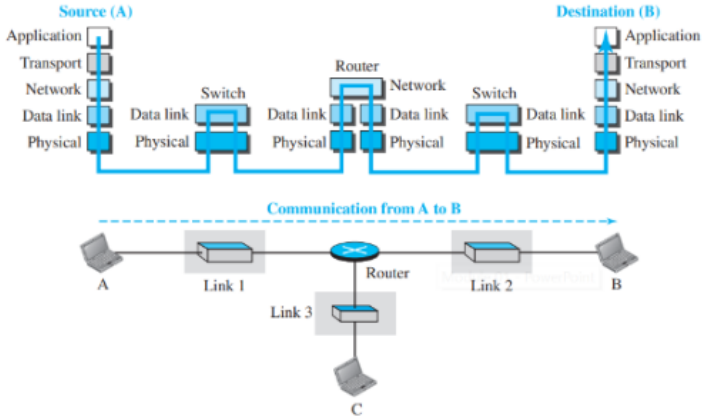
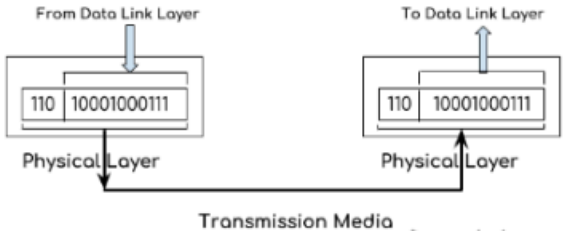


Figure 2.5 *Communication through an internet*



Physical Layer

- Responsible for carrying individual bits in a frame across the link.
- Two devices are connected by the transmission medium (cable or air).
- Transmission medium does not carry bits; it carries electrical or optical signals.



Data Link Layer

- Responsible for taking the datagram and moving it across the link.
- Link can be wired/ wireless LAN, Wired/ wireless WAN.
- Data link layer takes a datagram and encapsulates it in a packet called a frame.

Network Layer

- Responsible for creating a connection between the source computer and destination computer.
- Communication is host-to-host.
- Routers in the path are responsible for choosing the best route for each packet.
- Includes the main protocol , Internet Protocol (IP), defines the format of the packet, called datagram at network layer.
- IP is responsible for routing a packet from source to its destination.

Transport Layer

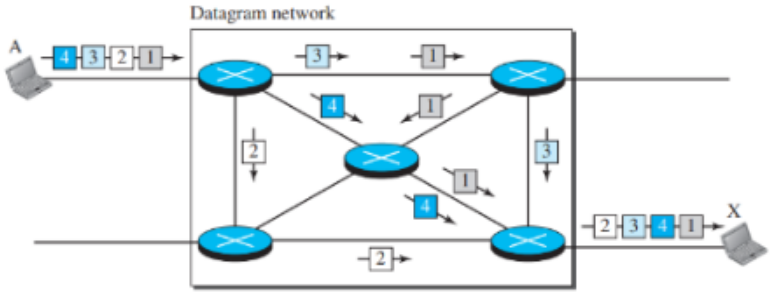
- Transport layer at the source host gets the message from the application layer, encapsulates it to transport layer.
- It is called as segment or user datagram in different protocol.
- TCP is a connection oriented protocol.
- TCP provides flow control , error control, congestion control to reduce the loss of segments due to congestion in the network.
- UDP (User Datagram Protocol), is a connection less protocol and transmits user datagrams without creating a logical connection.

		Application Layer • Two application layers exchange messages between each other. • Communication at the application layer is between two processes. • Process-to-process communication is the duty of the application layer. Protocols include: • HTTP: Hypertext Transfer Protocol • SMTP: Simple Mail Transfer Protocol • FTP: File Transfer Protocol			
	b.	**Explain types of packet switched networks and evaluate the total delay time of both** **SOLUTION** **Types of Packet-switched networks** • Datagram networks • Virtual circuit networks	**12**	**L2**	**CO2**

Datagram Networks

- In datagram network, each packet is treated independently of all others.
- Packets in this approach are referred to as **datagrams**.
- Datagram switching is normally done at the network layer.
- Datagram networks are sometimes referred to **as connectionless networks**.

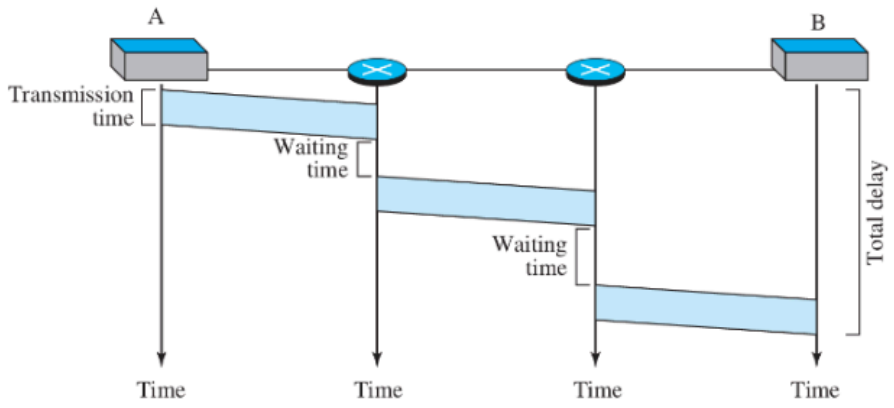
Figure 8.7 *A datagram network with four switches (routers)*



Efficiency

- Efficiency of datagram network is better than a circuit-switched network.
- Resources are allocated only when there are packets to be transferred.
- If source sends a packet and there is a delay of few minutes for another packet, resources are reallocated.

Figure 8.9 *Delay in a datagram network*



$$\text{Total delay} = 3T + 3\tau + w_1 + w_2$$

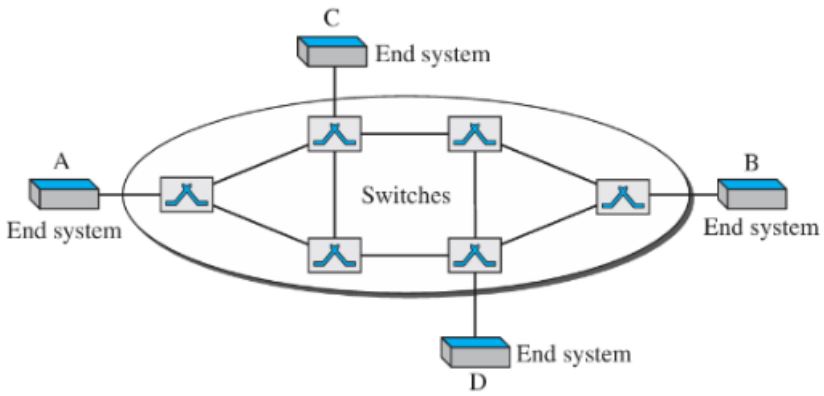
Virtual-Circuit Networks

- Virtual Circuit Networks is in between the circuit-switched network and a datagram network.

Characteristics:

- There are set-up and tear down phases along with data transfer phase.
- Resources are allocated during the setup phase or on demand.
- Data are packetized and each packet carries an address in the header.
- All packets follow the same path during the connection.

Figure 8.10 *Virtual-circuit network*



Addressing

Two types of addressing:

- Global
- Local (Virtual Circuit Identifier)

Global Addressing

- A source and destination needs to have a global address.
- Addresses are unique in the scope of the network.

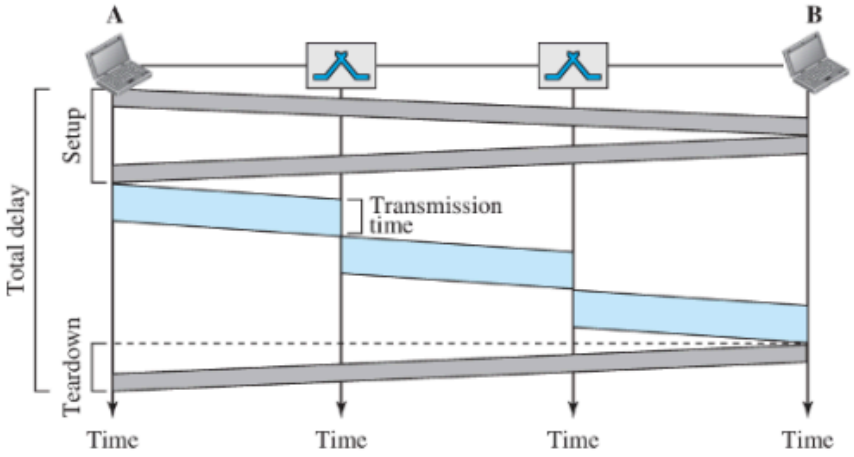
Local Addressing

Virtual-Circuit Identifier

- Identifier used for data transfer VCI or Label.
- It is a small number that has only switch scope.
- When it leaves a switch, it has a different VCI.

Figure 8.11 *Virtual-circuit identifier*

Figure 8.16 *Delay in a virtual-circuit network*



Total delay + 3*T* + 3τ + setup delay + teardown delay

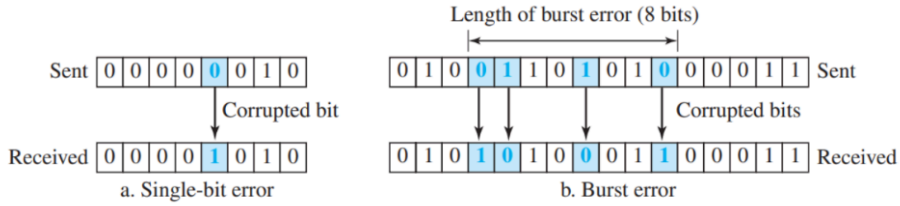
		MODULE -2			
3	**a.**	**Explain types of errors and Hamming distance. Find the Hamming distance between the following:** **(i) d(000,011)** **(ii) d(10101, 11110)**	**8**	**L3**	**CO2**

SOLUTION

Type of errors

- During transmission data is subjected to unpredictable changes because of **interference**.

Figure 10.1 *Single-bit and burst error*



Hamming Distance

- Hamming distance between two words (same size) is the differences between the corresponding bits.
- Hamming distance between two words ***x*** and ***y*** is shown as ***d(x,y).***
- EX: codeword 00000 is sent and 01101 is received, hamming distance is 3.
- d(00000, 01101) = 3
- If the distance between sent and the received codeword is not zero, the codeword is corrupted during transmission

Hamming distance between
d(000,011) =2
d(10101, 11110) = 3)

	b.	**Describe the working of CRC encoder and decoder. Perform division with respect to the following:** **Data word: 1001** **Divisor: 1011** **SOLUTION**	**12**	**L3**	**CO2**

Figure 10.5 *CRC encoder and decoder*

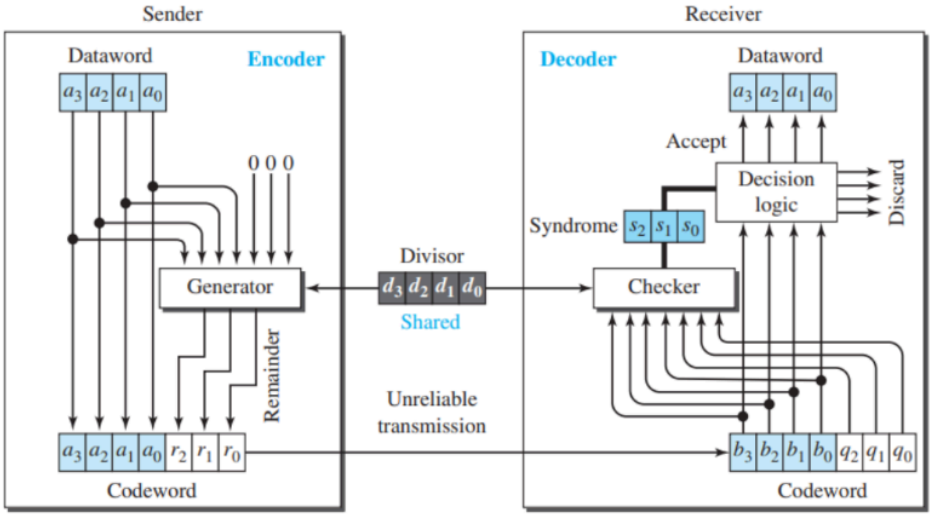


Figure 10.6 *Division in CRC encoder*

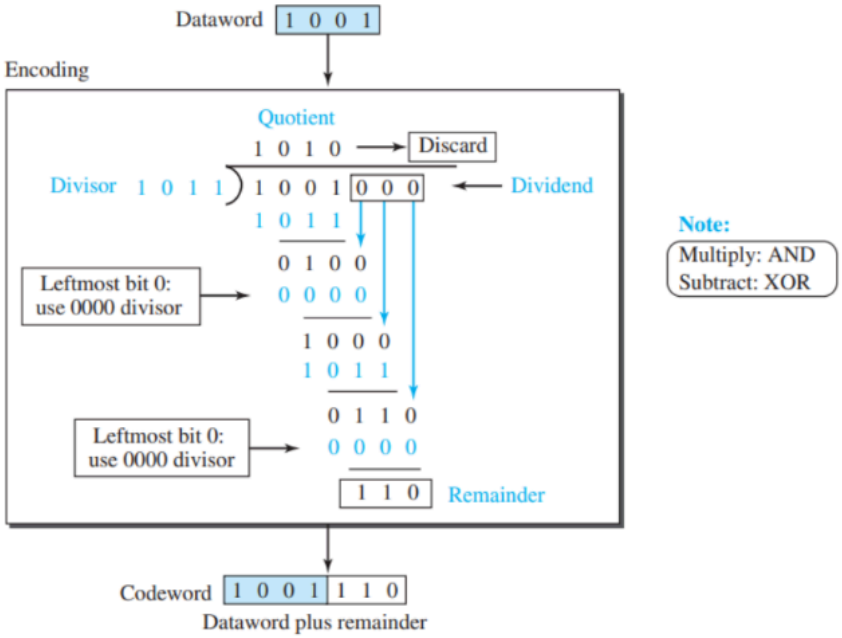
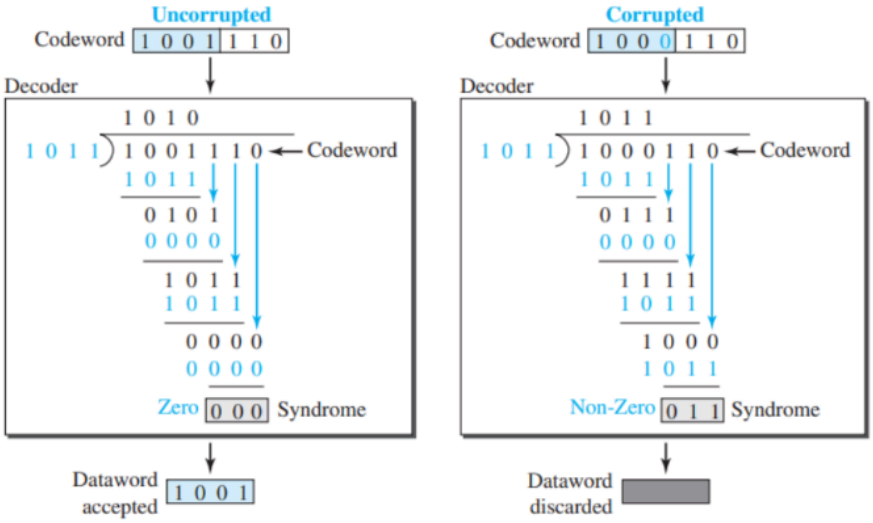


Figure 10.7 *Division in the CRC decoder for two cases*

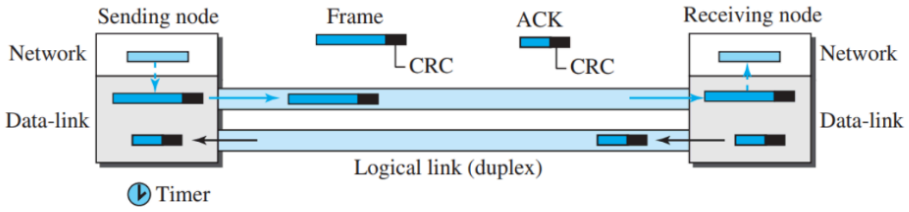


OR

4	**a.**	**Explain stop-and-wait protocol with FSM.** **SOLUTION**	**6**	**L2**	**CO2**

Stop and Wait Protocol

Figure 11.10 *Stop-and-Wait protocol*



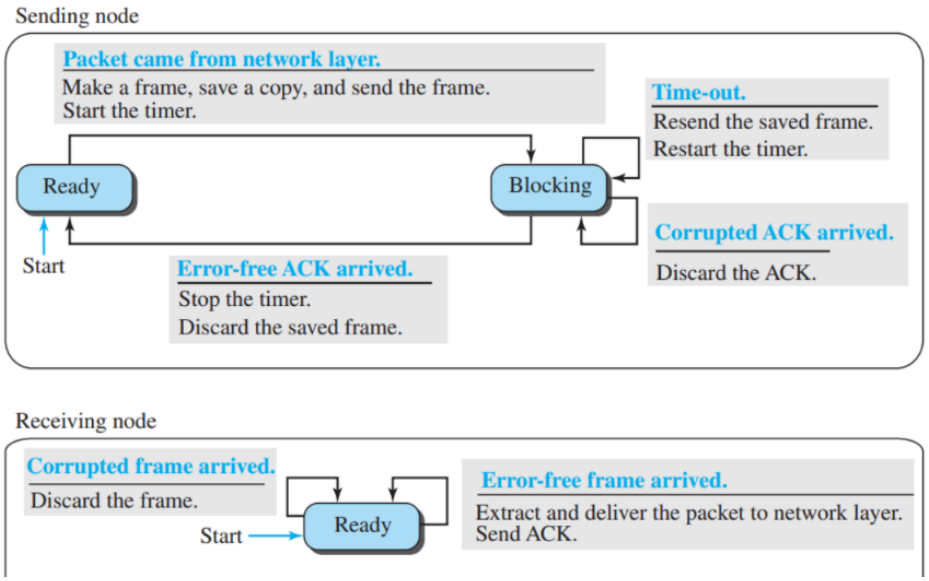
Stop and Wait Protocol

- Uses both flow and error control.
- Sender sends one frame at a time and waits for an acknowledgment before sending the next frame.
- When a frame arrives at receiver side, it is checked.
- If the CRC is incorrect, frame is corrupted and it is discarded.
- Every time the sender sends a frame, it starts a timer.
- If the acknowledgement arrives before the timer, then timer is stopped and next frame is sent.

Stop and Wait Protocol

- If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted.
- Sender needs to keep a copy of the frame until its acknowledgement arrives.
- When the acknowledgement arrives, the sender discards the copy and sends the next frame.

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

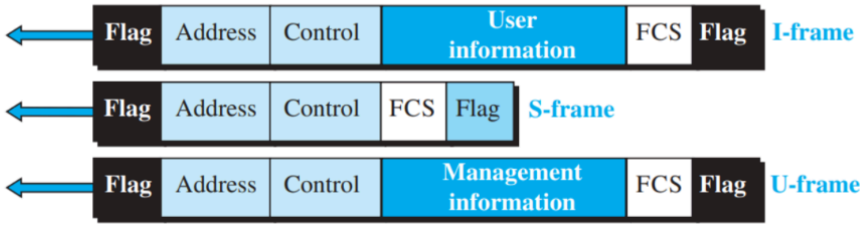


	b.	**Explain the three types of frames in HDLC.** **SOLUTION**	**8**	**L2**	**CO2**

HDLC defines three types of frames:

- Information Frames (I-Frames)
- Supervisory frames (S-Frames)
- Unnumbered frames (U-Frames)

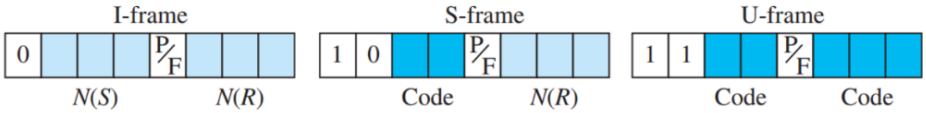
Figure 11.16 *HDLC frames*



Frames

- **Flag field:** Contains flag 01111110, which identifies both the beginning and the end of a frame.
- **Address field:** Contains the address of the secondary station.
- **Control Field**: One or two bytes used for flow or error control.
- **Information Field**: Contains the User's data from the network layer.
- **FCS Field**: Frame Check Sequence is the HDLC error detection field.

Figure 11.17 *Control field format for the different frame types*



Control Field for I frames

- I-frames are designed to carry user data from network layer.
- First bit defines the type.
- If the first bit is 1, it is an I-Frame.
- Next 3 bits, defines a sequence number.
- Last N(R) corresponds to the acknowledgement number.
- P/F is poll or Final. If it is set to 1, it means frame is sent by primary to secondary.

Control Field for S-Frames

- Supervisory frames are used for flow and error control.
- S-Frames do not have information fields.
- If the first two bits are 10, then its an S-Frame.
- Last 3 bits, N(R) correspond to acknowledgement number (ACK) or NAK.
- The 2 bits called code are used to define S-Frame.

Control Field for S-Frames

- 00 – Receive Ready (RR): This frame acknowledges the receipt of the frame or group of frames.
- 10 - Receive not Ready (RNR): It acknowledges the frame and announces that the receiver is busy and cannot receive more frames.
- 01 – Reject (REJ): It is a NAK frame. Informs the sender about the loss or damage of the frame.
- 11 – Selective Reject (SREJ): Informs the receiver that error is detected in a specific frame in sequence

Control Field of U-Frames

- U-Frames contain an information field, but used for system management information.
- U-frame codes are divided into two sections:
- a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit.
- Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

	c.	**Discuss controlled-access protocol using reservation method** **SOLUTION**	**6**	**L2**	**CO2**

Controlled Access

- In controlled access, the stations consult one another to find which station has the right to send.
- A station cannot send unless it has been authorized by other stations.

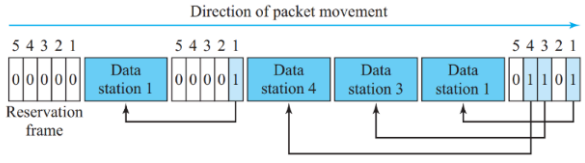
Three Controlled-access methods:

- Reservation
- Polling
- Token Passing

Reservation

- A station needs to make a reservation before sending data.
- Time is divided into intervals.
- In each interval, a reservation frame precedes the data frames sent in that interval.

Figure 12.18 *Reservation access method*



MODULE - 03					
5	a.	**Explain the services offered by network layer** **SOLUTION** ## Introduction to Network Layer • Network layer in TCP/IP protocol suite is responsible for the host-to-host delivery of datagrams. • It provides services to transport layer and receives services from the data-link layer. ## Packetizing • First duty of the network layer is **Packetizing**. • Payload coming from the upper layer is given to network layer. • A header containing the source and destination addresses are added. • The source is not allowed to change the contents of the packet unless its too large to transmit. ## Routers are not allowed to • Decapsulate the packets unless it needs fragmentation. • Change the address of source and destination. Allowed to: • Inspect the address for the purpose of forwarding the packet. • If packet is fragmented, header needs to be copied to all fragments.	6	L2	CO3

Other duties of the network layer are:

- Routing
- Forwarding

Routing

- Network layer is responsible for routing the packet from its **source** to **destination**.
- Responsible for finding the **best one** among those possible routes.
- Have some specific **strategies(Routing algorithms)** for defining the best route.

Forwarding

- Forwarding is an action applied by each router when a packet arrives at one of its interfaces.
- Decision making table called as **forwarding table** or **routing table** is used to forward the data.
- To do that, router uses a piece of information in the packet header i.e; **destination address** or **label**.

		Other Services • Error Control • Flow control • Congestion control • Quality of Service • Security			
	b.	**Define address space. Differentiate between classful addressing and classless addressing** **SOLUTION** **Address Space** • An **Address space** is the total number of addresses used by the protocol. • If a protocol uses ***b*** bits then the address spaces has 2^b. • IPv4 uses 32-bit addresses, the address space is 2^{32} or ***4,294,967,296*** (more than 4 billion)	**8**	**L2**	**co2**
	c.	**Explain Network Address Resolution (NAT) with a neat diagram** **SOLUTION**	**6**	**L2**	**CO3**

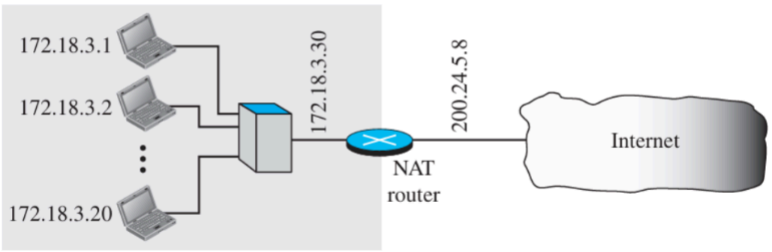
Network Address Translation

- NAT router can use a pool of global addresses.
- Uses the combination of IP address and Port addresses.

Table 18.1 *Five-column translation table*

Private address	*Private port*	*External address*	*External port*	*Transport protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

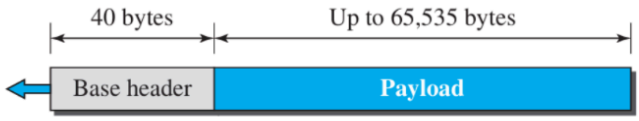
Figure 18.29 *NAT*



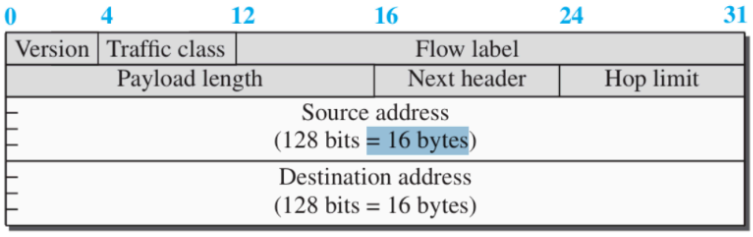
OR

6	**a.**	**Explain IPV6 packet format in detail** **SOLUTION**	**6**	**L2**	**CO2**

Figure 22.6 *IPv6 datagram*



a. IPv6 packet

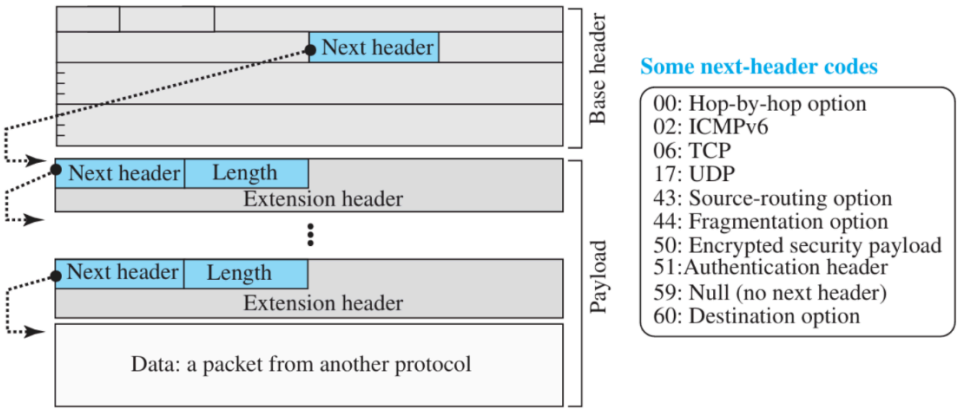


b. Base header

IPV6 Datagram

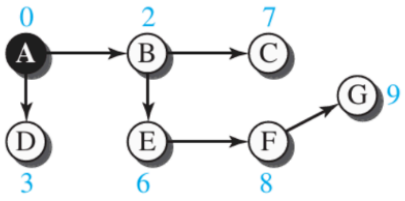
- **Version.** The 4-bit version field defines the version number of the IP. For IPv6, the value is 6.
- **Traffic Class**: Used to distinguish different payloads with different delivery requirements.
- **Flow label**: Provides flow of the data.
- **Payload Length**: Length of IP datagram excluding the header.
- **Next header**: The next header describes the type of the data that follows the base header.

- **Hop Limit.** Time to Live or hop count limit
- **Source and Destination address** : The source address 16byte (128 bit) and destination address (128 bit) 16 bytes.
- **Payload**: Describes the payload.

		Figure 22.7 *Payload in an IPv6 datagram* 			
	b.	**Discuss D-V routing highlighting the importance of distance vector.** **SOLUTION** **Distance Vector Routing Algorithm** • Distance Vector Routing Algorithm is used to find best route between source and destination. • Each node(router) creates its own least cost tree or table based on the information it has about its immediate neighbors. • These nodes interact with each other by exchanging information between immediate neighbors. • Router continuously updates it neighbors.	**7**	**L2**	**CO2**

Distance Vector

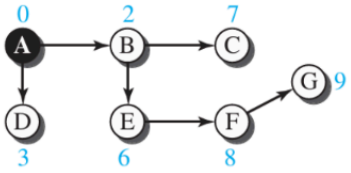
- Distance vector is a one-dimensional vector maintains name and distances of each node or router.



a. Tree for node A

Distance Vector

Figure 20.4 *The distance vector corresponding to a tree*

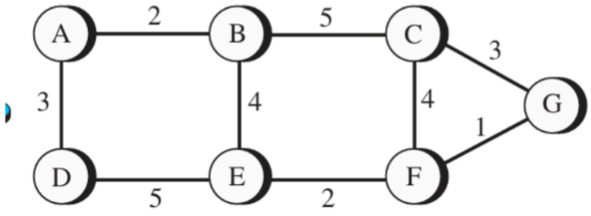


a. Tree for node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

- Distance vector gives least cost to the destinations.
- Each node in internet, when booted creates a distance vector with minimum information obtained from neighborhood.

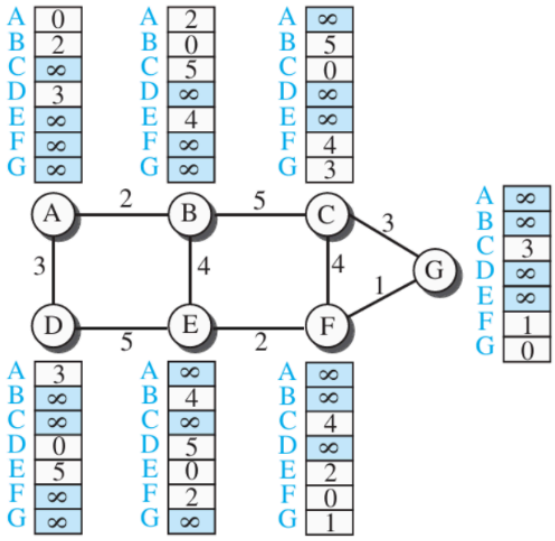


b. The weighted graph

- Later, the node sends some greeting messages out its interfaces and discovers, immediate neighbors and its distance.

- Designs the distance vector by inserting discovered distances in corresponding cells and leaves the value of other cells as infinity.

Figure 20.5 *The first distance vector for an internet*



Information is updated and exchanged as follows:

- After each node has created its vector, it sends a copy of vector to all its immediate neighbors.
- After a node receives a distance vector from its neighbor, it updates its distance vector using Bellman-Ford Equation.
- All the N nodes in the internet must be updated.

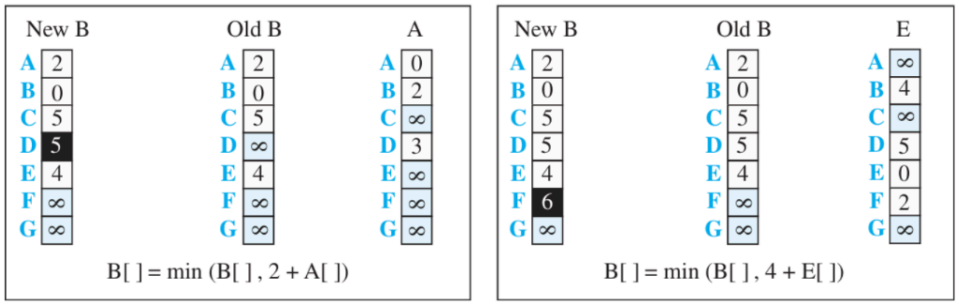
Bellman Ford Equation

$$D_{xy} = \min\{D_{xy}, (c_{xz} + D_{zy})\}$$

General Equation

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

Figure 20.6 *Updating distance vectors*



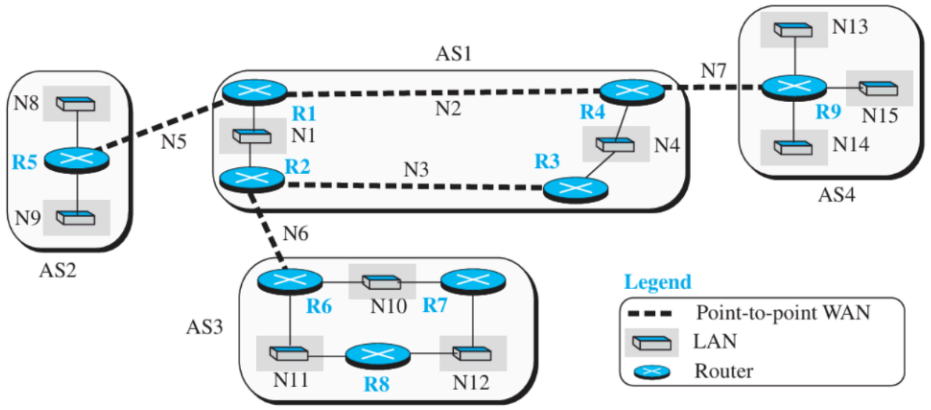
a. First event: B receives a copy of A's vector.

b. Second event: B receives a copy of E's vector.

		Algorithm
		
		for (y = 1 to N)
		{
		if (y is a neighbor)
		D[y] = c[*myself*][y]
		else
		D[y] = ∞
		}
		send vector {D[1], D[2], …, D[N]} to all neighbors			

	c.	**Describe BGP protocol in detail.** **SOLUTION** **Border Gateway Protocol Version4 (BGP4)** • BGP4 is the interdomain routing protocol used in internet today. • BGP4 is based on path-vector algorithm. • Autonomous systems are of three types: • Stud AS • Multihomed AS • Transient AS	**7**	**L2**	**CO$**

Figure 20.24 *A sample internet with four ASs*



BGP4

- Each AS uses an intradomain protocols, RIP or OSPF.
- Each router installs a variation of BGP4, called external BGP (eBGP) on each border router.
- Internal BGP (IBGP) is installed on all routers.
- Border routers will be running three routing protocols (intradomain, eBGP, and iBGP).
- Other routers are running two protocols (intradomain and iBGP).

Operation of External BGP (eBGP)

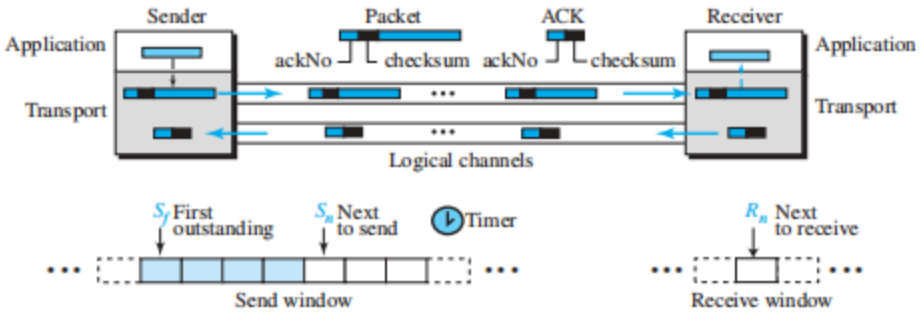
- When software is installed on two routers, they try to create a TCP connection using the well-known ports 179.
- The two routers that run the BGP processes are called:
- BGP peers or BGP speakers.
- eBGP variation of BGP allows two physically connected border routers to exchange the messages.
- Connected is established with the help of WANs

		Operation of Internal BG (IBGP) • Uses the service of TCP on the well known port 179. • Creates the session between any possible pair of routers within AS.			
		MODULE 4			
7	**a.**	**Explain the concept of port numbers mentioning ICANN ranges.** **SOLUTION** **Transport Layer Services: ICANN Ranges** ICANN has divided the port numbers into three ranges: • Well-known: Ports ranging from 0 to 1023 are assigned and controlled by ICANN. • Registered: Ports ranging from 1024 to 49,151 are not assigned or controlled by ICANN. • Dynamic Ports: Ports ranging from 49,152 to 65,535 are neither controlled nor registered. Used as temporary or private port numbers.	**5**	**L2**	**CO3**
	b.	**Explain Go-Back-N protocol.** **SOLUTION**	**9**	**L2**	**CO4**

23.2.3 Go-Back-*N* Protocol (GBN)

To improve the efficiency of transmission (to fill the pipe), multiple packets must be in transition while the sender is waiting for acknowledgment. In other words, we need to let more than one packet be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second. The first is called ***Go-Back-N* (GBN)** (the rationale for the name will become clear later). The key to Go-back-*N* is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet. We keep a copy of the sent packets until the acknowledgments arrive. Figure 23.23 shows the outline of the protocol. Note that several data packets and acknowledgments can be in the channel at the same time.

Figure 23.23 *Go-Back-N protocol*



Sequence Numbers

As we mentioned before, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

Acknowledgment Numbers

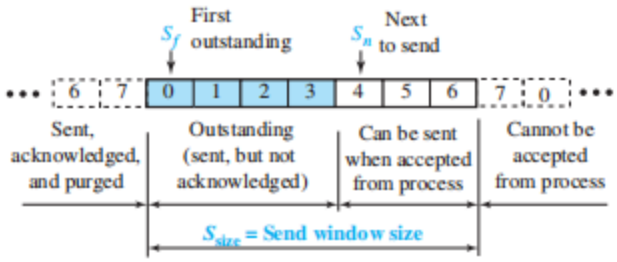
An acknowledgment number in this protocol is cumulative and defines the sequence number of the next packet expected. For example, if the acknowledgment number (ackNo) is 7, it means all packets with sequence number up to 6 have arrived, safe and sound, and the receiver is expecting the packet with sequence number 7.

> **In the Go-Back-*N* protocol, the acknowledgment number is cumulative and defines the sequence number of the next packet expected to arrive.**

Send Window

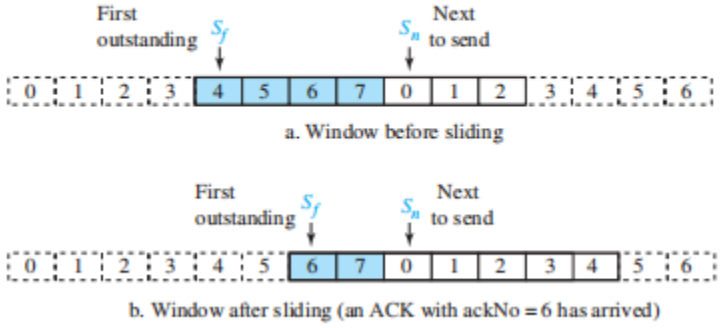
The send window is an imaginary box covering the sequence numbers of the data packets that can be in transit or can be sent. In each window position, some of these sequence numbers define the packets that have been sent; others define those that can be sent. The maximum size of the window is $2^m - 1$, for reasons that we discuss later. In this chapter, we let the size be fixed and set to the maximum value, but we will see later that some protocols may have a variable window size. Figure 23.24 shows a sliding window of size 7 ($m = 3$) for the Go-Back-*N* protocol.

Figure 23.24 *Send window for Go-Back-N*



The send window at any time divides the possible sequence numbers into four regions. The first region, left of the window, defines the sequence numbers belonging to packets that are already acknowledged. The sender does not worry about these packets and keeps no copies of them. The second region, colored, defines the range of sequence numbers belonging to the packets that have been sent, but have an unknown status. The sender needs to wait to find out if these packets have been received or were lost. We call these *outstanding* packets. The third range, white in the figure, defines the range of sequence numbers for packets that can be sent; however, the corresponding data have not yet been received from the application layer. Finally, the fourth region, right of the window, defines sequence numbers that cannot be used until the window slides.

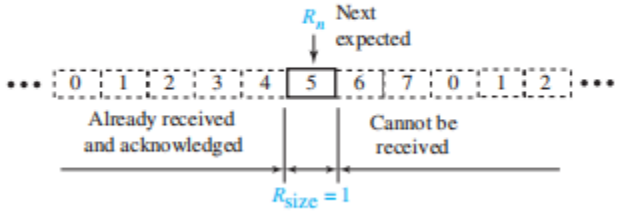
Figure 23.25 *Sliding the send window*



Receive Window

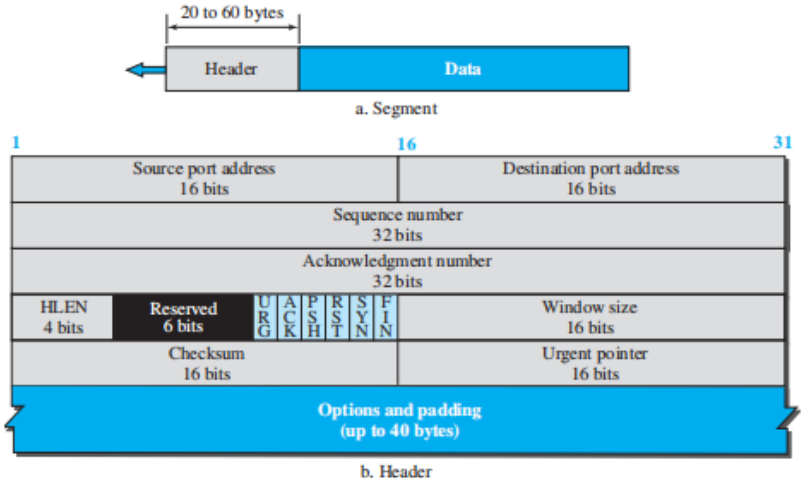
The receive window makes sure that the correct data packets are received and that the correct acknowledgments are sent. In Go-Back-*N*, the size of the receive window is always 1. The receiver is always looking for the arrival of a specific packet. Any packet arriving out of order is discarded and needs to be resent. Figure 23.26 shows the receive window. Note that we need only one variable, R_n (receive window, next packet expected), to define this abstraction. The sequence numbers to the left of the window belong to the packets already received and acknowledged; the sequence numbers to the right of this window define the packets that cannot be received. Any received packet with a sequence number in these two regions is discarded. Only a packet with a sequence number matching the value of R_n is accepted and acknowledged. The receive window also slides, but only one slot at a time. When a correct packet is received, the window slides, $R_n = (R_n + 1)$ modulo 2^m.

Figure 23.26 *Receive window for Go-Back-N*



	c.	**Explain TCP segment format with a neat diagram** **SOLUTION**	**6**	**L3**	**CO3**

Figure 24.7 *TCP segment format*



24.3.3 Segment

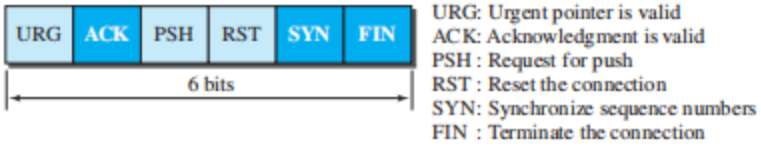
Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a *segment*.

Format

The format of a segment is shown in Figure 24.7. The segment consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. We will discuss some of the header fields in this section. The meaning and purpose of these will become clearer as we proceed through the section.

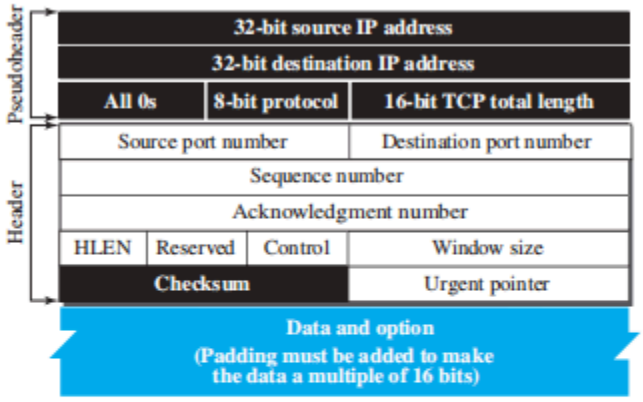
- ***Source port address.*** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- ***Destination port address.*** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
- ***Sequence number.*** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment (discussed later) each party uses a random number generator to create an **initial sequence number** (ISN), which is usually different in each direction.
- ***Acknowledgment number.*** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it returns $x + 1$ as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- ***Header length.*** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 ($5 \times 4 = 20$) and 15 ($15 \times 4 = 60$).
- ***Control.*** This field defines 6 different control bits or flags, as shown in Figure 24.8. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in the figure. We will discuss them further when we study the detailed operation of TCP later in the chapter.

Figure 24.8 *Control field*



- ***Window size.*** This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (*rwnd*) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- ***Checksum.*** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory. The same pseudoheader, serving the same

Figure 24.9 *Pseudoheader added to the TCP datagram*



The use of the checksum in TCP is mandatory.

- ***Urgent pointer.*** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.
- ***Options.*** There can be up to 40 bytes of optional information in the TCP header. We will discuss some of the options used in the TCP header later in the section.

OR

8	a.	**Discuss the connection establishment in TCP.** **SOLUTION**	8	L2	CO3

Connection Establishment

TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

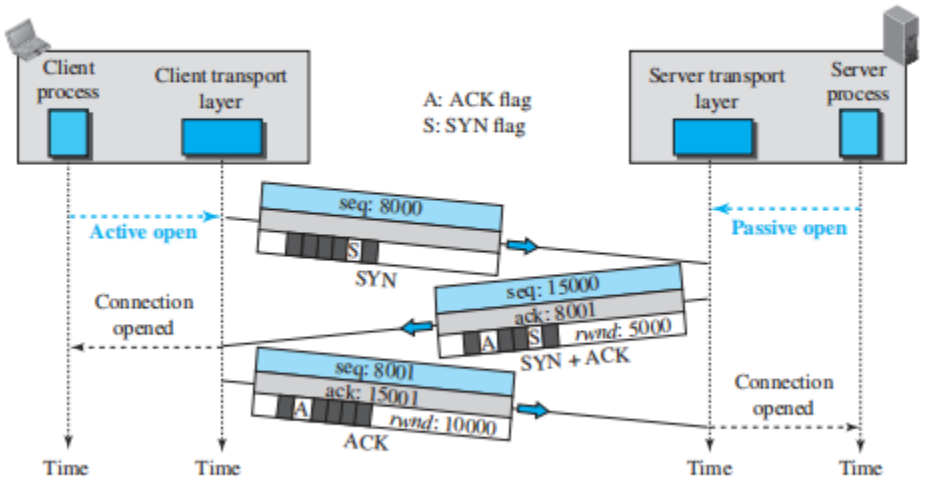
Three-Way Handshaking

The connection establishment in TCP is called ***three-way handshaking.*** In our example, an application program, called the *client,* wants to make a connection with another application program, called the *server,* using TCP as the transport-layer protocol.

The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This request is called a *passive open*. Although the server TCP is ready to accept a connection from any machine in the world, it cannot make the connection itself.

The client program issues a request for an *active open.* A client that wishes to connect to an open server tells its TCP to connect to a particular server. TCP can now start the three-way handshaking process, as shown in Figure 24.10.

Figure 24.10 *Connection establishment using three-way handshaking*

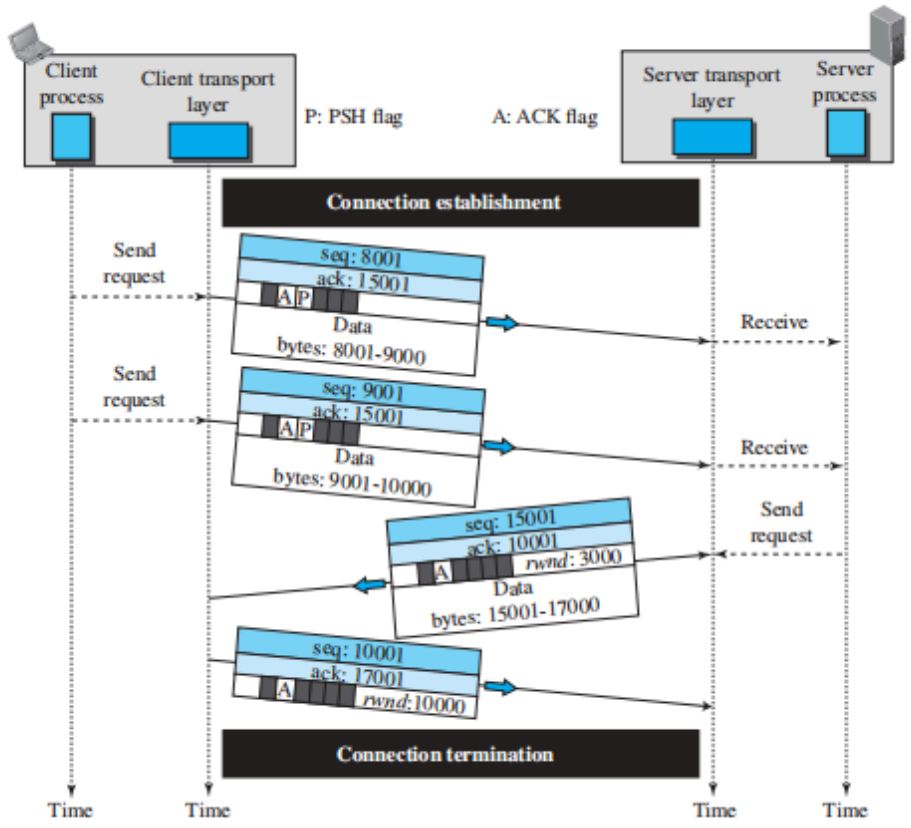


Data Transfer

After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgments in both directions. We will study the rules of acknowledgment later in the chapter; for the moment, it is enough to know that data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data. Figure 24.11 shows an example.

In this example, after a connection is established, the client sends 2,000 bytes of data in two segments. The server then sends 2,000 bytes in one segment. The client sends one more segment. The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there is no more data to be sent. Note the values of the sequence and acknowledgment numbers. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received. We discuss the use of this flag in more detail later. The segment from the server, on the other hand, does not set the push flag. Most TCP implementations have the option to set or not to set this flag.

Figure 24.11 *Data transfer*



Connection Termination

Either of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client. Most implementations today allow two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three-Way Handshaking

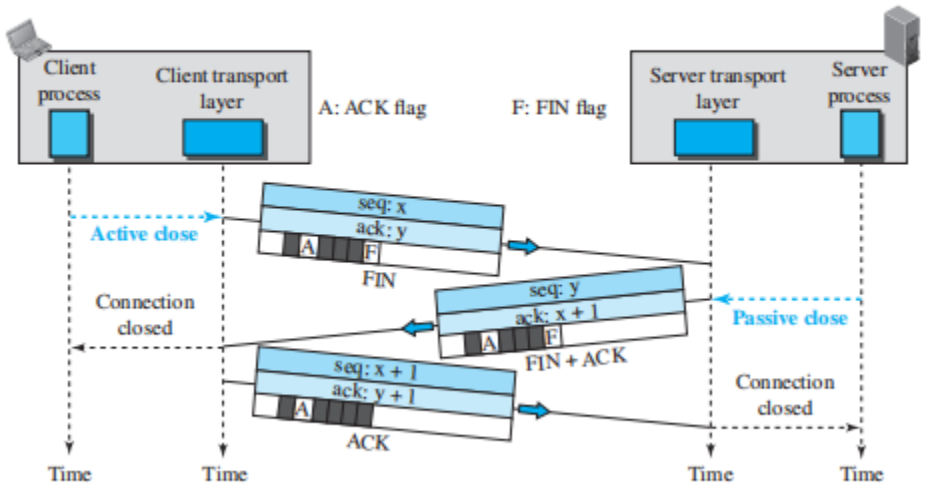
Most implementations today allow *three-way handshaking* for connection termination, as shown in Figure 24.12.

1. In this situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set. Note that a FIN segment can include the last chunk of data sent by the client or it can be just a control segment as shown in the figure. If it is only a control segment, it consumes only one sequence number because it needs to be acknowledged.

> **The FIN segment consumes one sequence number if it does not carry data.**

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN + ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction. This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number because it needs to be acknowledged.
3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server. This segment contains the acknowledgment number, which is one plus the sequence number received in the FIN segment from the server. This segment cannot carry data and consumes no sequence numbers.

Figure 24.12 *Connection termination using three-way handshaking*



	b.	**Explain error control in TCP using acknowledgements** **SOLUTION**	**4**	**L2**	**CO3**

Acknowledgment

TCP uses acknowledgments to confirm the receipt of data segments. Control segments that carry no data, but consume a sequence number, are also acknowledged. ACK segments are never acknowledged.

ACK segments do not consume sequence numbers and are not acknowledged.

Acknowledgment Type

In the past, TCP used only one type of acknowledgment: cumulative acknowledgment. Today, some TCP implementations also use selective acknowledgment.

Cumulative Acknowledgment (ACK) TCP was originally designed to acknowledge receipt of segments cumulatively. The receiver advertises the next byte it expects to receive, ignoring all segments received and stored out of order. This is sometimes referred to as *positive cumulative acknowledgment,* or ACK. The word *positive* indicates that no feedback is provided for discarded, lost, or duplicate segments. The 32-bit ACK field in the TCP header is used for cumulative acknowledgments, and its value is valid only when the ACK flag bit is set to 1.

Selective Acknowledgment (SACK) More and more implementations are adding another type of acknowledgment called *selective acknowledgment,* or SACK. A SACK does not replace an ACK, but reports additional information to the sender. A SACK reports a block of bytes that is out of order, and also a block of bytes that is duplicated, i.e., received more than once. However, since there is no provision in the TCP header for adding this type of information, SACK is implemented as an option at the end of the TCP header. We discuss this new feature when we discuss options in TCP on the book website.

Generating Acknowledgments

When does a receiver generate acknowledgments? During the evolution of TCP, several rules have been defined and used by several implementations. We give the most common rules here. The order of a rule does not necessarily define its importance.

1. When end A sends a data segment to end B, it must include (piggyback) an acknowledgment that gives the next sequence number it expects to receive. This rule decreases the number of segments needed and therefore reduces traffic.
2. When the receiver has no data to send and it receives an in-order segment (with expected sequence number) and the previous segment has already been acknowledged, the receiver delays sending an ACK segment until another segment

		arrives or until a period of time (normally 500 ms) has passed. In other words, the receiver needs to delay sending an ACK segment if there is only one outstanding in-order segment. This rule reduces ACK segments. 3. When a segment arrives with a sequence number that is expected by the receiver, and the previous in-order segment has not been acknowledged, the receiver immediately sends an ACK segment. In other words, there should not be more than two in-order unacknowledged segments at any time. This prevents the unnecessary retransmission of segments that may create congestion in the network. 4. When a segment arrives with an out-of-order sequence number that is higher than expected, the receiver immediately sends an ACK segment announcing the sequence number of the next expected segment. This leads to the *fast retransmission* of missing segments (discussed later). 5. When a missing segment arrives, the receiver sends an ACK segment to announce the next sequence number expected. This informs the receiver that segments reported missing have been received. 6. If a duplicate segment arrives, the receiver discards the segment, but immediately sends an acknowledgment indicating the next in-order segment expected. This solves some problems when an ACK segment itself is lost.			
	c.	Discuss three algorithms for handling congestion in TCP SOLUTION **Congestion Window (TCP sender-side limit)** When we discussed flow control in TCP, we mentioned that the size of the send window is controlled by the receiver using the value of rwnd, which is advertised in each segment traveling in the opposite direction. The use of this strategy guarantees that the receive window is never overflowed with the received bytes (no end congestion)	**8**	**L2**	**CO3**

To control the number of segments to transmit, TCP uses another variable called a congestion window, cwnd, whose size is controlled by the congestion situation in the network .

The cwnd variable and the rwnd variable together define the size of the send window in TCP.

The first is related to the congestion in the middle (network); the second is related to the congestion at the end.

Effective window = min(rwnd, cwnd)

Congestion Detection:
Before discussing how the value of cwnd should be set and changed, we need to describe how a TCP sender can detect the possible existence of congestion in the network.

The TCP sender uses the occurrence of two events as signs of congestion in the network: time-out and receiving three duplicate ACKs.

1. Packet Timeout
If the sender doesn't receive an ACK within the Retransmission Timeout (RTO) period →
it assumes the packet was lost due to congestion (not corruption).

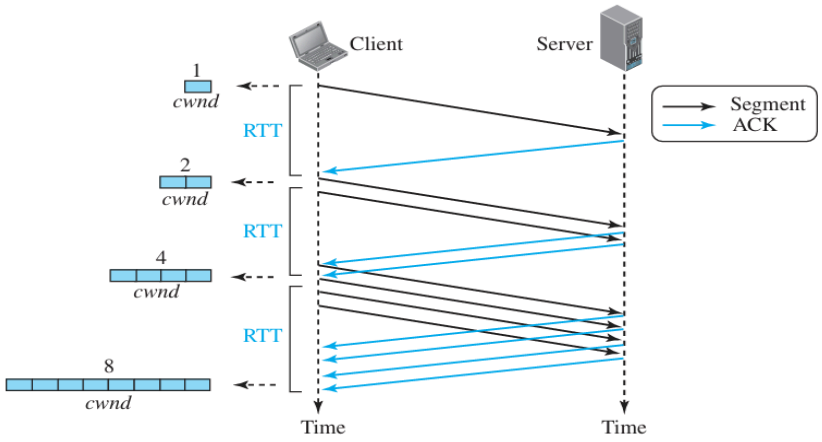
2. Duplicate ACKs (Fast Retransmit)
If a packet is lost but the receiver still gets later packets,
the receiver keeps sending duplicate ACKs (same ACK number repeatedly).

Congestion Policies

Slow Start: Exponential Increase - Congestion Avoidance

The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives.

Figure 24.29 *Slow start, exponential increase*

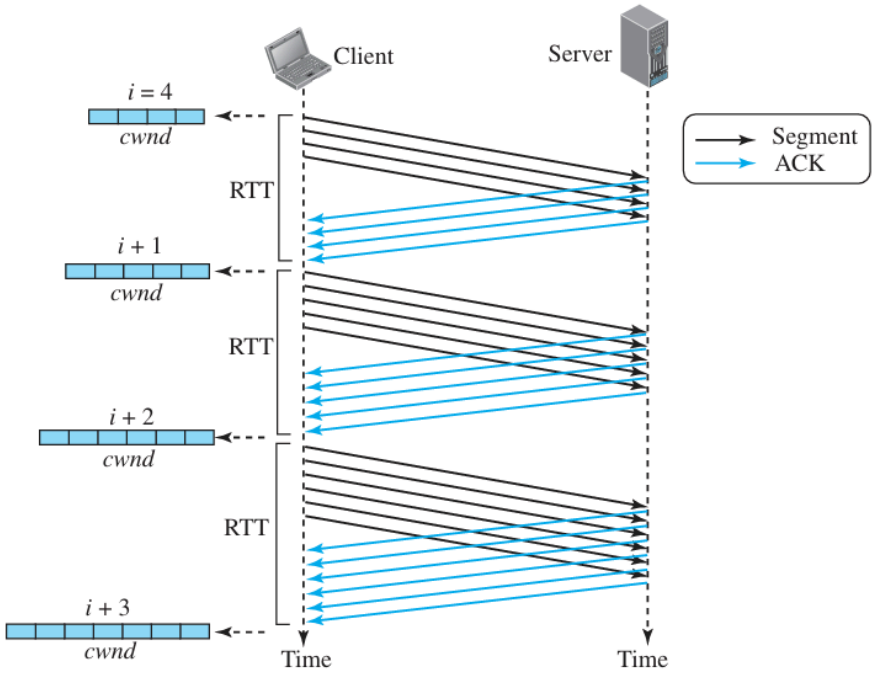


Congestion Avoidance: Additive Increase
If we continue with the slow-start algorithm, the size of the congestion window increases exponentially.

Time = cwnd + 1 = 1 + 1 = 2→ 2
1 = cwnd + 2 = 2 + 2 = 4→ 2^2 = cwnd + 4 = 4 + 4 = 8→ 2^3

TCP defines another algorithm called congestion avoidance, which increases the cwnd additively instead of exponentially.

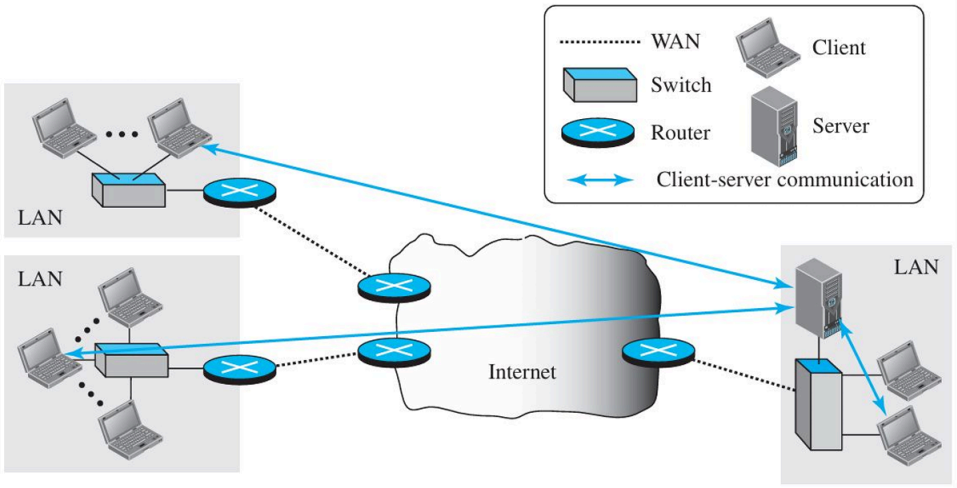
'igure 24.30 *Congestion avoidance, additive increase*



MODULE - 5

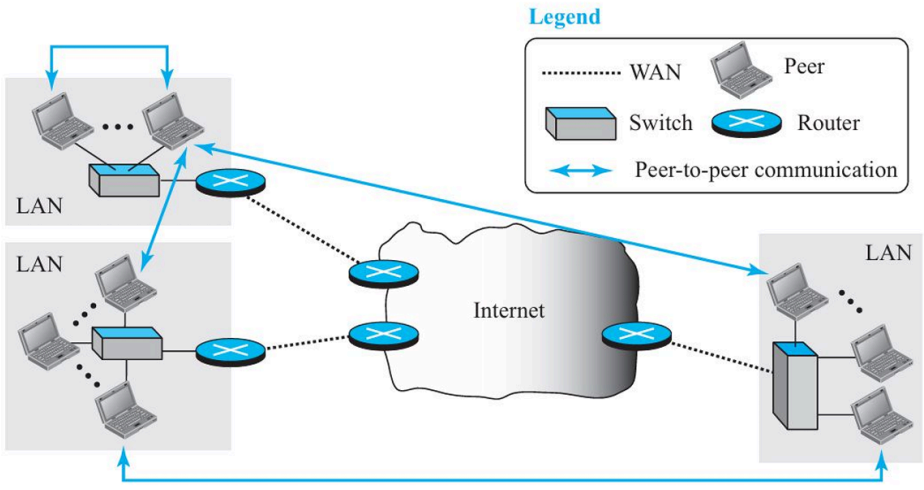
9	**a.**	**Discuss application layer paradigms with neat diagram** **SOLUTION** **Application Layer Paradigms** • To use the internet, we need two application programs to interact with each other. To enable the communication, two paradigms are developed: • Client-server paradigm • Peer-to-peer paradigm **Client-Server Paradigm** • Most popular and traditional paradigm. • In this paradigm, the service provider is an application program called server. It runs continuously, waiting for another application program called client. • Some server process can provide a specific type of service to many clients. • The server must be running all the time	**5**	**L2**	**Co3**

Figure 25.2 *Example of a client-server paradigm*



Peer-to-Peer Paradigm • Often abbreviated as P2P paradigm, does not need for a server process to be running all the time. • Responsibility is shared between the peers.

Figure 25.3 *Example of a peer-to-peer paradigm*



	b.	**Explain the use of sockets in process-to-process communication.** **SOLUTION** **Use of Sockets in Process-to-Process Communication** Sockets play a vital role in enabling process-to-process communication in	**7**	**L2**	**CO3**

computer networks. A socket is defined as an endpoint of communication that allows application processes running on different hosts or the same host to exchange data through a network.

Identification of Processes

In a networked environment, multiple processes may run simultaneously on a single machine. While an IP address identifies a host, it cannot identify a specific process. Sockets solve this problem by combining the IP address and port number, thereby uniquely identifying a process.

Communication Mechanism

For communication, a server process creates a socket and binds it to a known port number. A client process creates its own socket and initiates a connection request to the server's socket. Once the connection is established, data can be exchanged bidirectionally between the two processes.

Interface to Transport Layer

Sockets provide a programming interface between the application layer and the transport layer. Applications use socket APIs to send and receive data without dealing with low-level network details such as routing and error control.

Types of Sockets

Sockets are mainly classified into:

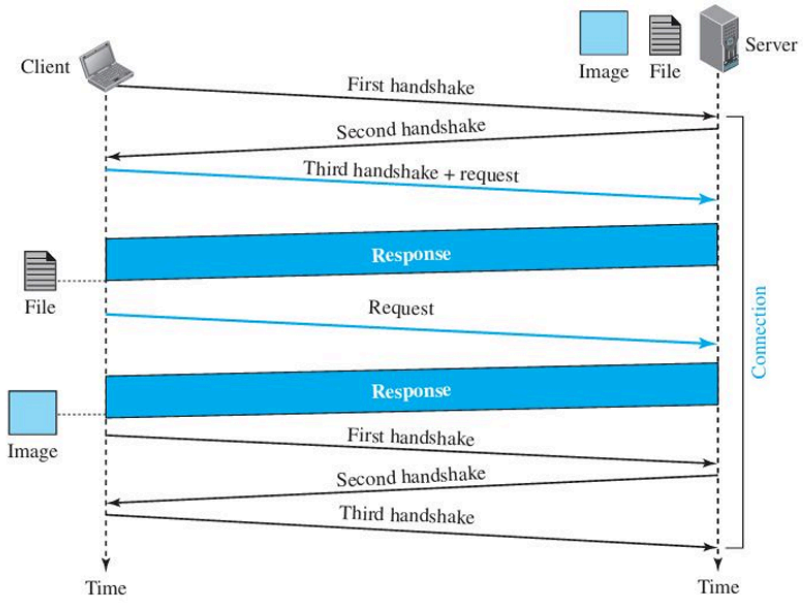
- **Stream sockets (TCP):** Provide reliable, connection-oriented communication with ordered data delivery.
- **Datagram sockets (UDP):** Provide connectionless communication with faster but unreliable data transfer.

Multiplexing and Demultiplexing

Sockets enable the operating system to perform multiplexing and demultiplexing of data. Incoming data packets are delivered to the correct process based on the destination port number associated with the socket.

Applications of Sockets

Sockets are widely used in client-server applications such as web browsers and web servers, email services, file transfer systems, and cloud-based applications, enabling seamless communication between distributed processes.

	c.	**Discuss the connection types in HTTP along with formats of messages.** **SOLUTION** **HTTP:** Hyper Text Transfer Protocol (HTTP) is used to define how the client-server programs can be written to retrieve web pages from the web. • A HTTP client sends a request; a HTTP server returns a response. • Server uses a port number 80; client uses a temporary port number. • Connection is established before data exchange and is terminated after data exchange If the web pages are located on different servers, a new connection is made for every server. If some objects are located on the same server, we have two choices: • To retrieve each object using a new TCP connection. (Non-Persistent Connection) • To make a TCP connection and retrieve them all. (Persistent Connection) **Figure 26.4** *Example 26.4*  **Message Formats** • HTTP protocol defines the format of the request and response messages. • Each message made up of 4 sections. • The first section in the request message is called the request line. • The first section in the response message is called the status line. • The other three sections have the same names in request and response messages. **Message Formats:** Request Message •There are three fields in this line separated by one space and terminated by two characters. •Method: Defines	**8**	**L2**	**CO3**

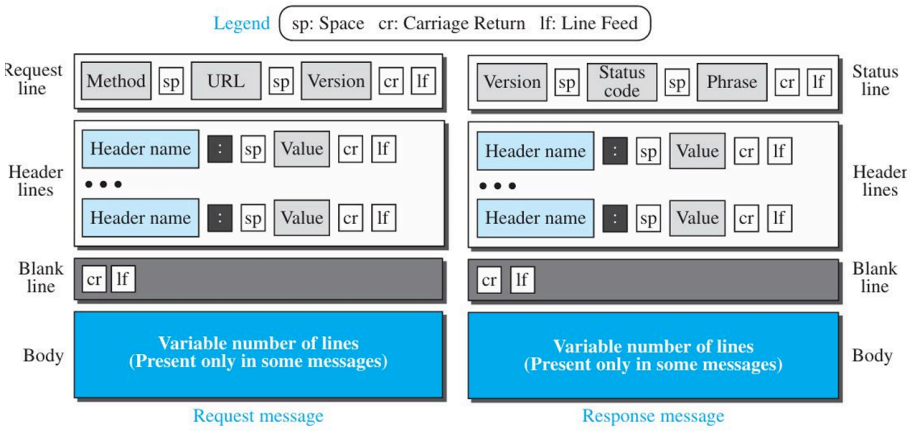
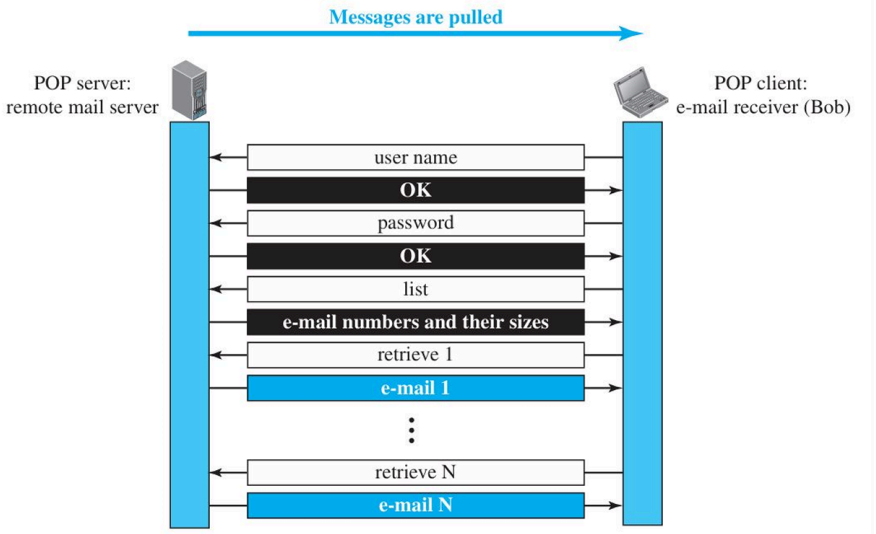
		the request type. Message Formats: Response Message •A response message consists of a status line, header lines and a blank line, and sometimes a body. **Figure 26.5** *Formats of the request and response messages*  			
OR					
10	**a.**	**Explain POP and IMAP protocols** **SOLUTION** **POP:** The client POP3 software is installed on the recipient computer. The server POP3 software is installed on the mail server. • Initiated when user needs to download its e-mail from the mailbox. • The client opens a connection to the server on TCP port 110. • Sends username and password to access the mailbox. **POP has two modes: • Delete mode:** The mail is deleted from the mailbox after each retrieval. • **Keep mode:** The mail remains in the mailbox after retrieval.	**8**	**L2**	**CO4**

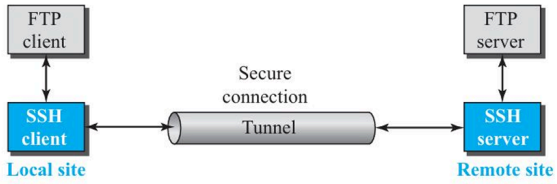
Figure 26.17 *POP3*



IMAP4 • Internet Mail Access Protocol, Version 4 , is similar to POP3. • A user can check the e-mail header prior to downloading. • A user can search the contents of the e-mail for a specific string of characters prior to downloading. • A user can partially download e-mail. • A user can create, delete or rename mailboxes on the mail server. • A user can create a hierarchy of mailboxes in a folder for e-mail storage.

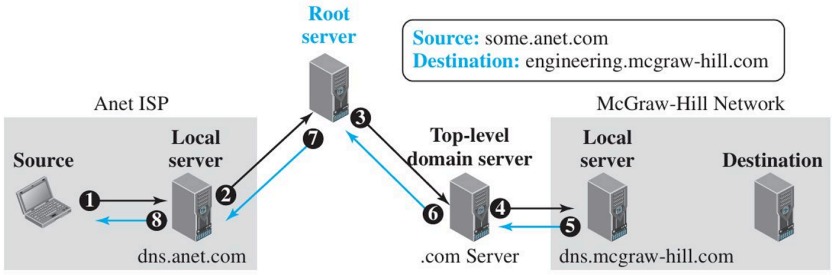
	b.	**Discuss the applications of SSH Protocol** **SOLUTION** **SSH:Secure Shell** Originally designed to replace TELNET. • Used for several purposes such as remote logging and file transfer. • Two versions of SSH • SSH-1 : Deprecated because of security flaws • SSH-2 **Applications** **SSH for remote Logging:** Several free and commercial applications use SSH for remote logging. (Putty, Tectia) • SSH for File Transfer: SFTP (Secure FTP) is used for file transfer. • Port Forwarding: Creates a tunnel through which the messages belonging to other protocol can travel. (SSH Tunnelling)	**4**	**L2**	**CO4**

Figure 26.26 *Port forwarding*



	c.	**Explain resolution in DNS.** **SOLUTION** **Resolution:** If the server has the information, it provides to the resolver. • otherwise, it either refers the resolver to other servers or asks other servers to provide the information. • After the resolver receives the mapping, it delivers the result to the process that requested it. **Resolution can be of two types: • Recursive Resolution • Iterative Resolution**	**8**	**L2**	**CO3**

Figure 26.36 *Recursive resolution*



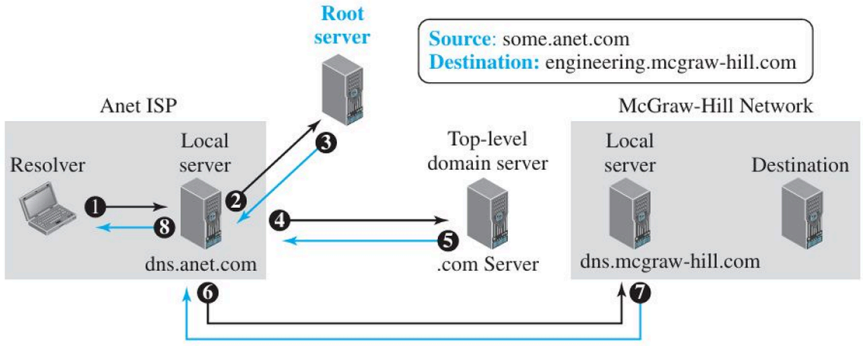
Recursive DNS resolution is a process in which the DNS server takes complete responsibility for resolving a domain name into its corresponding IP address.

When a client sends a recursive query, the DNS server must return either the final IP address or an error message. The client does not contact any other DNS servers during this process.

The recursive DNS server performs multiple queries on behalf of the client by contacting root DNS servers, top-level domain (TLD) servers, and authoritative DNS servers until the required IP address is obtained.

Recursive resolution typically uses caching, which improves performance and reduces network traffic for repeated queries.

Figure 26.37 *Iterative resolution*



Iterative DNS resolution is a process in which a DNS server does not take full responsibility for resolving a domain name.
Instead, each DNS server responds with the best information it has, usually a referral to another DNS server that is closer to the requested domain.

In this method, the client or resolver repeatedly queries different DNS servers (root, TLD, authoritative) until the final IP address is obtained.
Each server only provides partial information and does not guarantee the final resolution.