MODULE 5

TRANSPORT LAYER & APPLICATION LAYER

CO – Students will be able to illustrate the functions and protocols of the Transport layer and Application layer

Prepared By Mr. EBIN PM, Chandigarh University, Punjab

TRANSPORT LAYER

- The transport layer is responsible for the delivery of a message from one process to another
- A process is an application program running on a host
- The transport layer header must include a type of address called a service-point address in the OSI model and port number or port addresses in the Internet and TCP/IP protocol suite.
- A transport layer protocol can be either connectionless or connection-oriented.
- In the transport layer, a message is normally divided into transmittable segments.

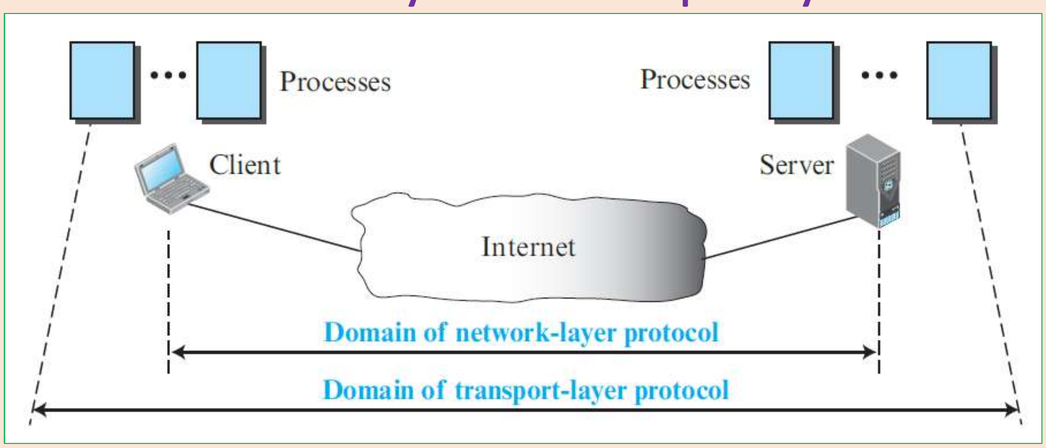
TRANSPORT LAYER SERVICES

❖Process-to-Process Communication

- The first duty of a transport-layer protocol is to provide process-to-process communication.
- A process is an application-layer entity (running program) that uses the services of the transport layer.
- A network-layer protocol can deliver the message only to the destination computer. However, this is an incomplete delivery. The message still needs to be handed to the correct process. This is where a transport-layer protocol takes over. A transport-layer protocol is responsible for delivery of the message to the appropriate process.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 3

Network layer versus transport layer



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 4

❖Addressing: Port Numbers

- For communication, we must define the local host, local process, remote host, and remote process.
- The local host and the remote host are defined using IP addresses To define the processes, we need second identifiers, called port numbers.
- In the TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 (16 bits).
- TCP/IP has decided to use universal port numbers for servers; these are called **well-known port numbers**.
- Every client process knows the well-known port number of the corresponding server process

- Internet Corporation for Assigned Names and Numbers **(ICANN)** divided the port numbers into three ranges

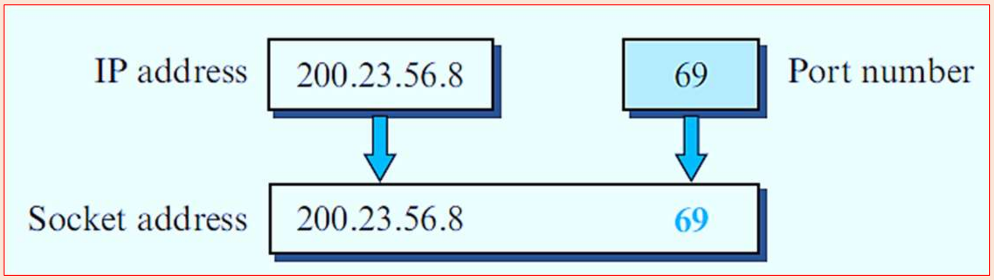
➢ **Well-known ports -** The ports ranging from 0 to 1,023 are assigned and controlled by ICANN. These are the well-known ports.

➢ **Registered ports -** The ports ranging from 1,024 to 49,151 are not assigned or controlled by ICANN. They can only be registered with ICANN to prevent duplication.

➢ **Dynamic ports -** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

Socket Addresses

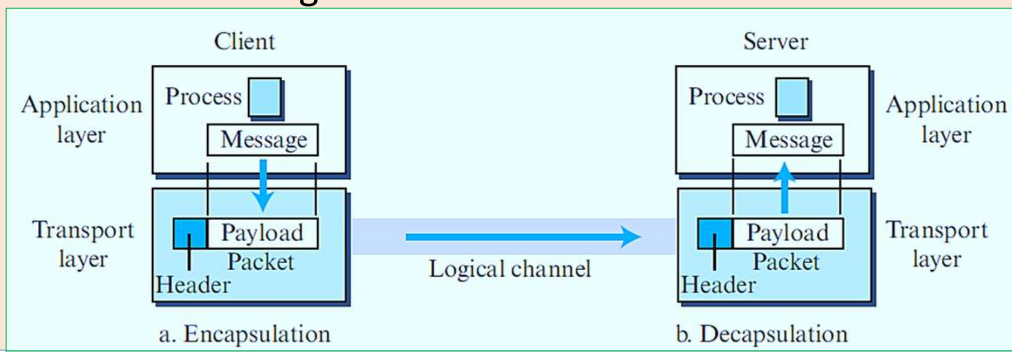
- A transport-layer protocol in the TCP suite needs both the IP address and the port number, at each end, to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.



Encapsulation and Decapsulation

- Encapsulation happens at the sender site.
- When a process has a message to send, it passes the message to the transport layer along with a pair of socket addresses and some other pieces of information, which depend on the transport-layer protocol.
- The transport layer receives the data and adds the transport-layer header.
- The packets at the transport layers in the Internet are called user datagrams, segments, or packets, depending on what transport-layer protocol we use.

- Decapsulation happens at the receiver site.
- When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer.
- The sender socket address is passed to the process in case it needs to respond to the message received.



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 9

❖Multiplexing and Demultiplexing

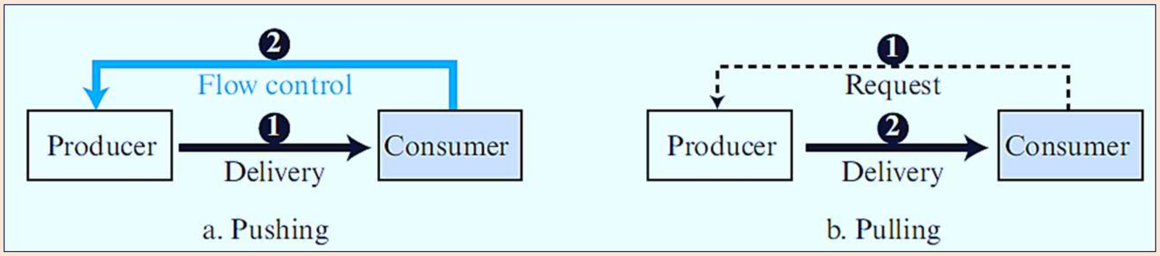
- Whenever an entity accepts items from more than one source, this is referred to as multiplexing (many to one).
- whenever an entity delivers items to more than one source, this is referred to as demultiplexing (one to many).
- The transport layer at the source performs multiplexing; the transport layer at the destination performs demultiplexing.

❖Flow Control

- Delivery of items from a producer to a consumer can occur in one of two ways: pushing or pulling

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 10

- If the sender delivers items whenever they are produced without a prior request from the consumer, the delivery is referred to as **pushing.**
- If the producer delivers the items after the consumer has requested them, the delivery is referred to as **pulling**.



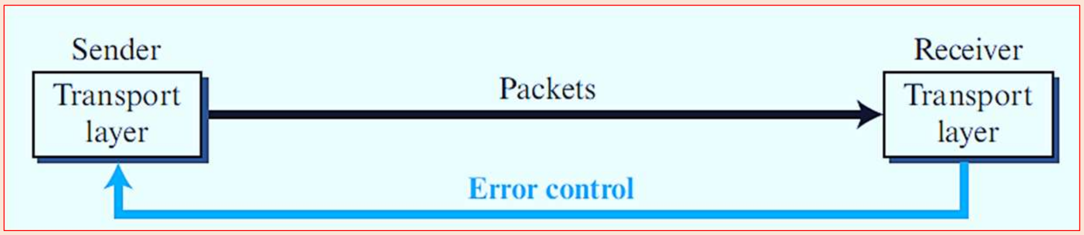
- When the producer pushes the items, the consumer may be overwhelmed and there is a need for flow control, in the opposite direction, to prevent discarding of the items.

- Although flow control can be implemented in several ways, one of the solutions is normally to use two buffers: one at the sending transport layer and the other at the receiving transport layer.
- A buffer is a set of memory locations that can hold packets at the sender and receiver.
- When the buffer of the sending transport layer is full, it informs the application layer to stop passing chunks of messages.
- When the buffer of the receiving transport layer is full, it informs the sending transport layer to stop sending packets. When there are some vacancies, it informs the sending transport layer that it can send packets again.

❖Error Control

- Error control at the transport layer is responsible for

1. Detecting and discarding corrupted packets.
2. Keeping track of lost and discarded packets and resending them.
3. Recognizing duplicate packets and discarding them.
4. Buffering out-of-order packets until the missing packets arrive.



➢Sequence Numbers

- Error control requires that the sending transport layer knows which packet is to be resent and the receiving transport layer knows which packet is a duplicate, or which packet has arrived out of order.
- This can be done if the packets are numbered. We can add a field to the transport-layer packet to hold the sequence number of the packet.
- When a packet is corrupted or lost, the receiving transport layer can somehow inform the sending transport layer to resend that packet using the sequence number.
- The receiving transport layer can also detect duplicate packets if two received packets have the same sequence number.

- The out-of-order packets can be recognized by observing gaps in the sequence numbers.

➢Acknowledgment

- We can use both positive and negative signals as error control.
- positive signals are more common at the transport layer.
- The receiver side can send an acknowledgment (ACK) for each of a collection of packets that have arrived safe and sound.
- The receiver can simply discard the corrupted packets. The sender can detect lost packets if it uses a timer. When a packet is sent, the sender starts a timer. If an ACK does not arrive before the timer expires, the sender resends the packet.

- Duplicate packets can be silently discarded by the receiver. Out-of-order packets can be either discarded (to be treated as lost packets by the sender), or stored until the missing ones arrives.

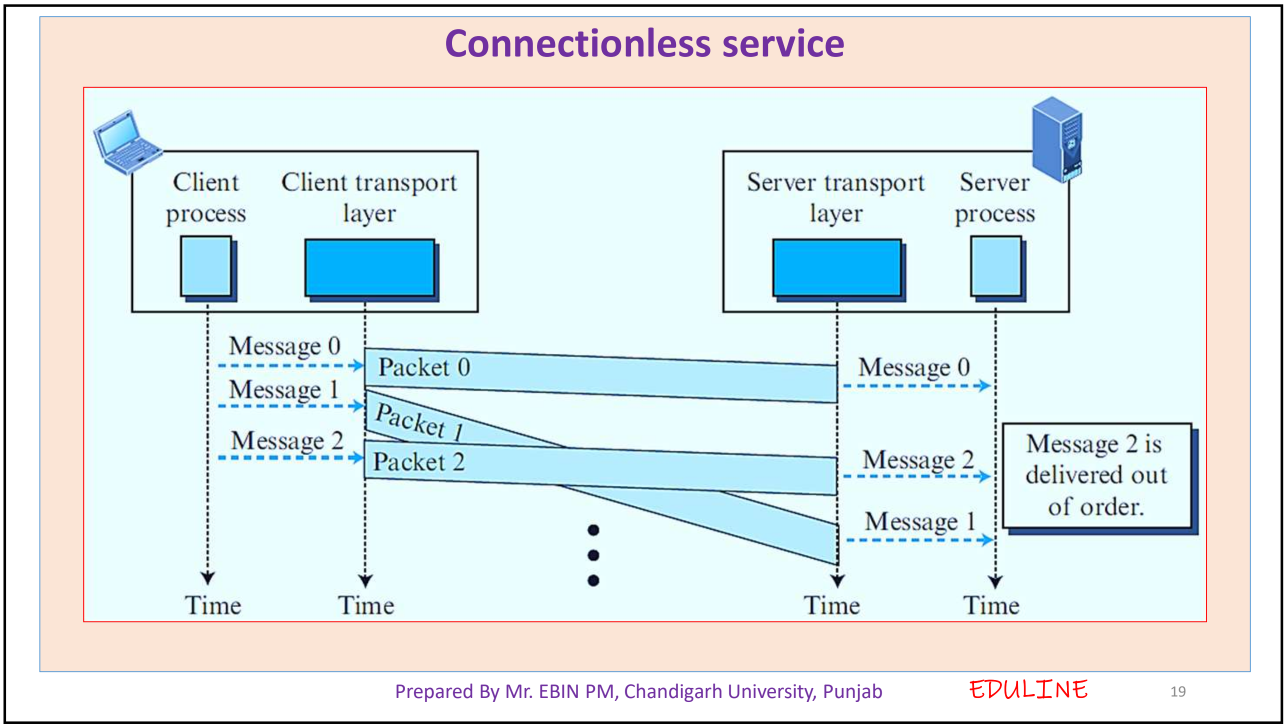
❖Connectionless and Connection-Oriented Services

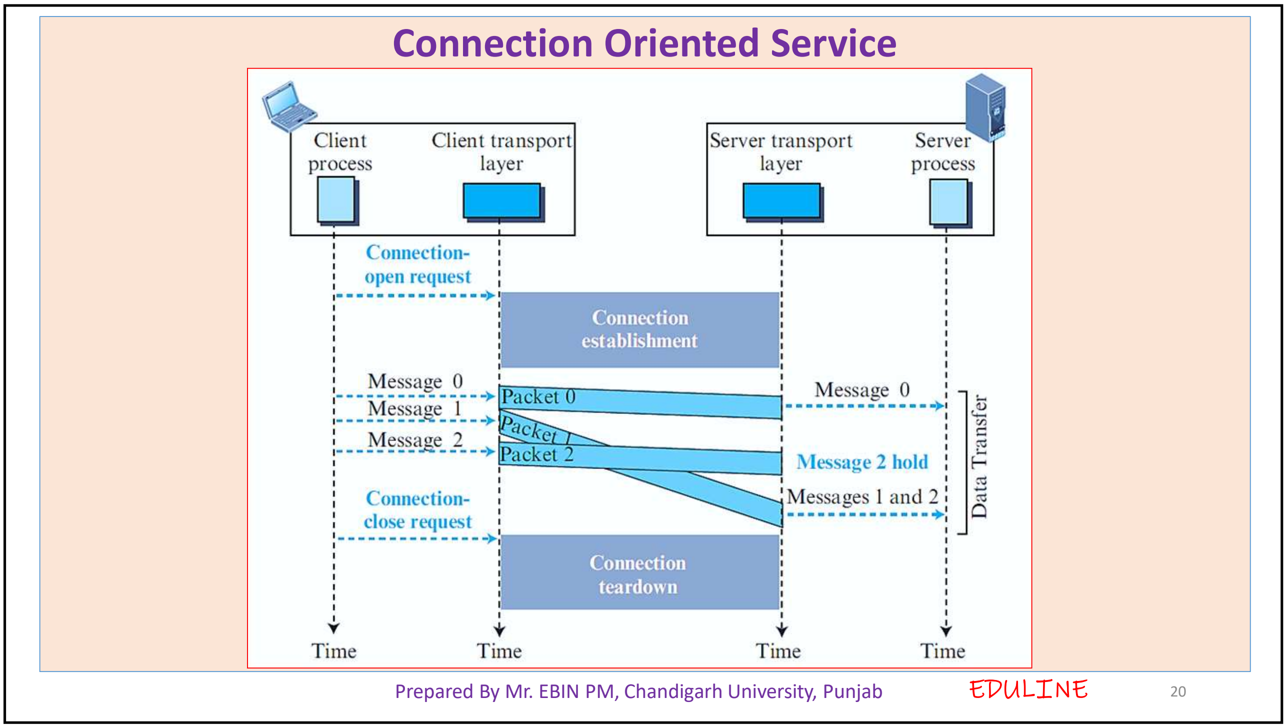
- In a connectionless service, the source process (application program) needs to divide its message into chunks of data of the size acceptable by the transport layer and deliver them to the transport layer one by one.
- The transport layer treats each chunk as a single unit without any relation between the chunks.
- When a chunk arrives from the application layer, the transport layer encapsulates it in a packet and sends it.

- However, since there is no dependency between the packets at the transport layer, the packets may arrive out of order at the destination and will be delivered out of order to the server process.
- The situation would be worse if one of the packets were lost. Since there is no numbering on the packets, the receiving transport layer has no idea that one of the messages has been lost.
- We can say that no flow control, error control, or congestion control can be effectively implemented in a connectionless service.

Connection-Oriented Service

- In a connection-oriented service, the client and the server first need to establish a logical connection between themselves.
- The data exchange can only happen after the connection establishment.
- After data exchange, the connection needs to be torn down.

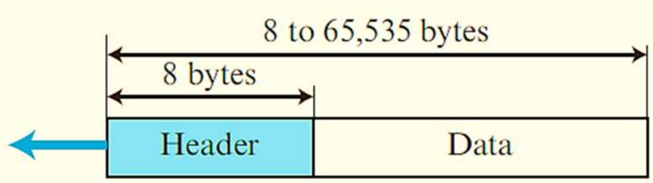

Connectionless service
Client process
Client transport layer
Server transport layer
Server process
Message 0
Message 1
Message 2
Packet 0
Packet 1
Packet 2
Message 0
Message 2
Message 1
Message 2 is delivered out of order.
Time
Time
Time
Time
Prepared By Mr. EBIN PM, Chandigarh University, Punjab
EDULINE
19


Connection Oriented Service
Client process
Client transport layer
Server transport layer
Server process
Connection-open request
Connection establishment
Message 0
Message 1
Message 2
Packet 0
Packet 1
Packet 2
Message 0
Message 2 hold
Messages 1 and 2
Data Transfer
Connection-close request
Connection teardown
Time
Time
Time
Time
Prepared By Mr. EBIN PM, Chandigarh University, Punjab
EDULINE
20

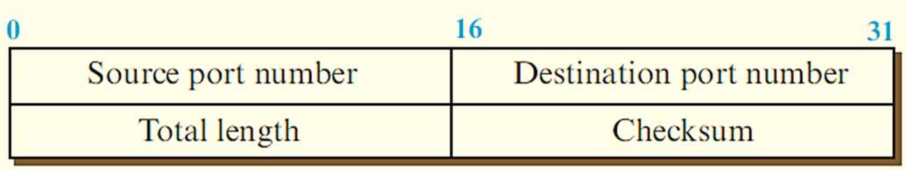
USER DATAGRAM PROTOCOL (UDP)

- The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.
- UDP is a very simple protocol using a minimum of overhead.
- If a process wants to send a small message and does not care much about reliability, it can use UDP.
- Sending a small message using UDP takes much less interaction between the sender and receiver than using TCP.
- UDP packets, called **user datagrams**, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).

- The first two fields define the source and destination port numbers. The third field defines the total length of the user datagram, header plus data.



a. UDP user datagram



b. Header format

❖UDP Services

➢Process-to-Process Communication

- UDP provides process-to-process communication using socket addresses, a combination of IP addresses and port numbers

➢Connectionless Services

- UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- The user datagrams are not numbered. There is no connection establishment and no connection termination. This means that each user datagram can travel on a different path.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 23

➢Flow Control

- UDP is a very simple protocol. There is no flow control, and hence no window mechanism. The receiver may overflow with incoming messages.

➢Error Control

- There is no error control mechanism in UDP except for the checksum.
- This means that the sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is silently discarded

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 24

➢ Congestion Control

- Since UDP is a connectionless protocol, it does not provide congestion control.
- UDP assumes that the packets sent are small and sporadic and cannot create congestion in the network. This assumption may or may not be true today, when UDP is used for interactive real-time transfer of audio and video.

➢ Encapsulation and Decapsulation

- To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 25

➢ Multiplexing and Demultiplexing

- In a host running a TCP/IP protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and demultiplexes.

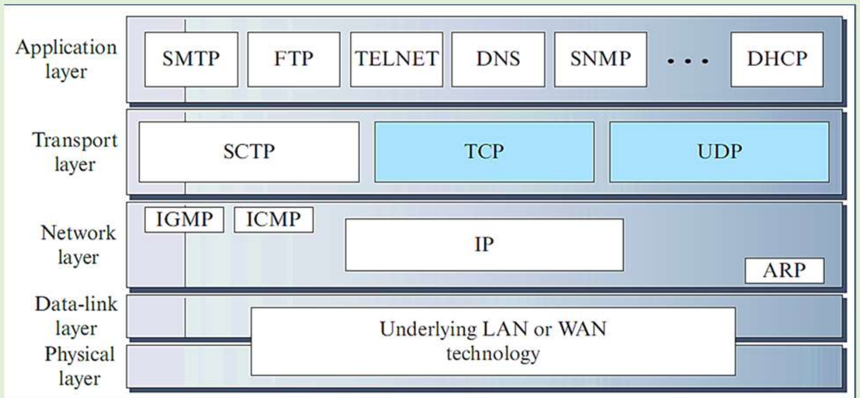
❖ Applications of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow- and error-control mechanisms
- UDP is a suitable transport protocol for multicasting
- UDP is used for management processes such as SNMP

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 26

- UDP is used for some route updating protocols such as Routing Information Protocol(RIP).
- UDP is normally used for interactive real-time applications

Position of transport-layer protocols in the TCP/IP protocol suite



Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 27

TRANSMISSION CONTROL PROTOCOL (TCP)

- TCP is a connection-oriented, reliable protocol.
- TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.
- TCP uses a combination of GBN and SR protocols to provide reliablity.
- TCP uses checksum (for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.
- TCP is the most common transport-layer protocol in the Internet.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 28

❖TCP Services

➢Process-to-Process Communication

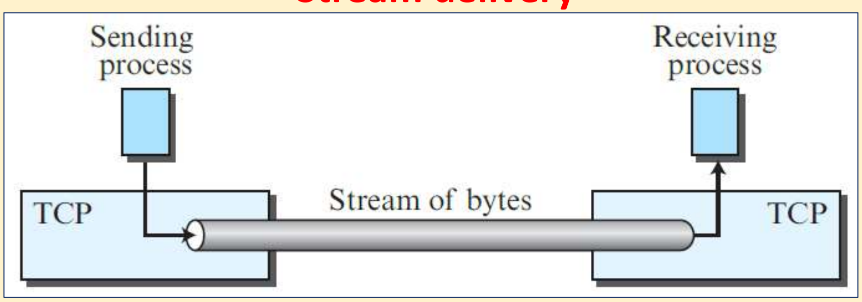
- As with UDP, TCP provides process-to-process communication using port numbers

➢Stream Delivery Service

- TCP is a stream-oriented protocol. It allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet. The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 29

Stream delivery



▪ Sending and Receiving Buffers

- Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction. One way to implement a buffer is to use a circular array.

Prepared By Mr. EBIN PM, Chandigarh University, Punjab EDULINE 30

- **Segments -** At the transport layer, TCP groups a number of bytes together into a packet called a segment.

➢ Full-Duplex Communication

- TCP offers full-duplex service, where data can flow in both directions at the same time. Each TCP endpoint then has its own sending and receiving buffer, and segments move in both directions.

➢ Multiplexing and Demultiplexing

- Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver. However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes

➢ Connection-Oriented Service

TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

1. The two TCP's establish a logical connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated

- Note that this is a logical connection, not a physical connection

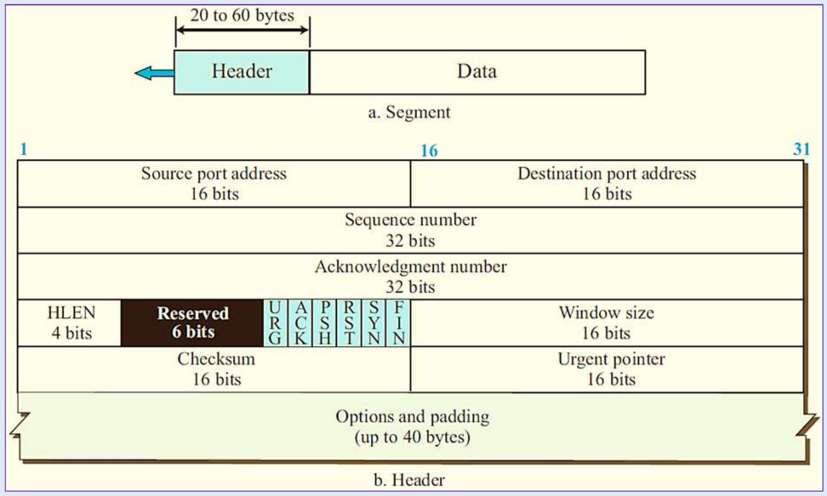
➢ Reliable Service

- TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP SEGMENT

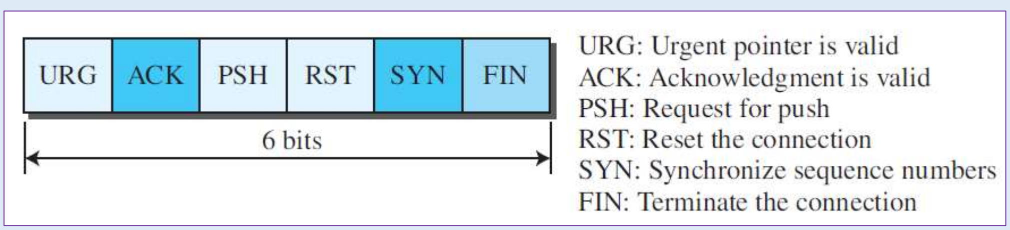
- A packet in TCP is called a **segment**

TCP segment format



- **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. During connection establishment each party uses a random number generator to create an initial sequence number (ISN).

- **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
- **Control -** This field defines 6 different control bits or flags



- **Window size -** This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits
- **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.
- **Urgent pointer** - This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.

TCP CONNECTION

- In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination

❖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

➢Three-Way Handshaking

- The connection establishment in TCP is called three-way handshaking

- 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**. Now the server TCP is ready to accept a connection from any machine in the world.
- 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.

➢STEP 1

- The client sends the first segment (SYN segment) in which only the SYN flag is set. This segment is for synchronization of sequence numbers.

- The client chooses a random number as the first sequence number and sends this number to the server. This sequence number is called the initial sequence number (ISN).
- Note that the SYN segment is a control segment and carries no data. However, it consumes one sequence number because it needs to be acknowledged.
- A SYN segment cannot carry data, but it consumes one sequence number.

➢STEP 2

- The server sends the second segment, a SYN + ACK segment with two flag bits set as: SYN and ACK. This segment has a dual purpose.

- First, it is a SYN segment for communication in the other direction. The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client.
- The server also acknowledges the receipt of the SYN segment from the client by setting the ACK flag and displaying the next sequence number it expects to receive from the client.
- A SYN + ACK segment cannot carry data, but it does consume one sequence number.

➢STEP 3

- The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.

➢Data Transfer

- After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgments in both directions.

➢Pushing Data

- The sending TCP uses a buffer to store the stream of data coming from the sending application program. The sending TCP can select the segment size.
- The receiving TCP also buffers the data when they arrive and delivers them to the application program when the application program is ready or when it is convenient for the receiving TCP.
- This type of flexibility increases the efficiency of TCP.

➢Urgent Data

- There are occasions in which an application program needs to send urgent bytes, some bytes that need to be treated in a special way by the application at the other end.
- The solution is to send a segment with the URG bit set.
- The sending application program tells the sending TCP that the piece of data is urgent.
- The sending TCP creates a segment and inserts the urgent data at the beginning of the segment. The rest of the segment can contain normal data from the buffer.

❖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.

Connection establishment using three-way handshaking