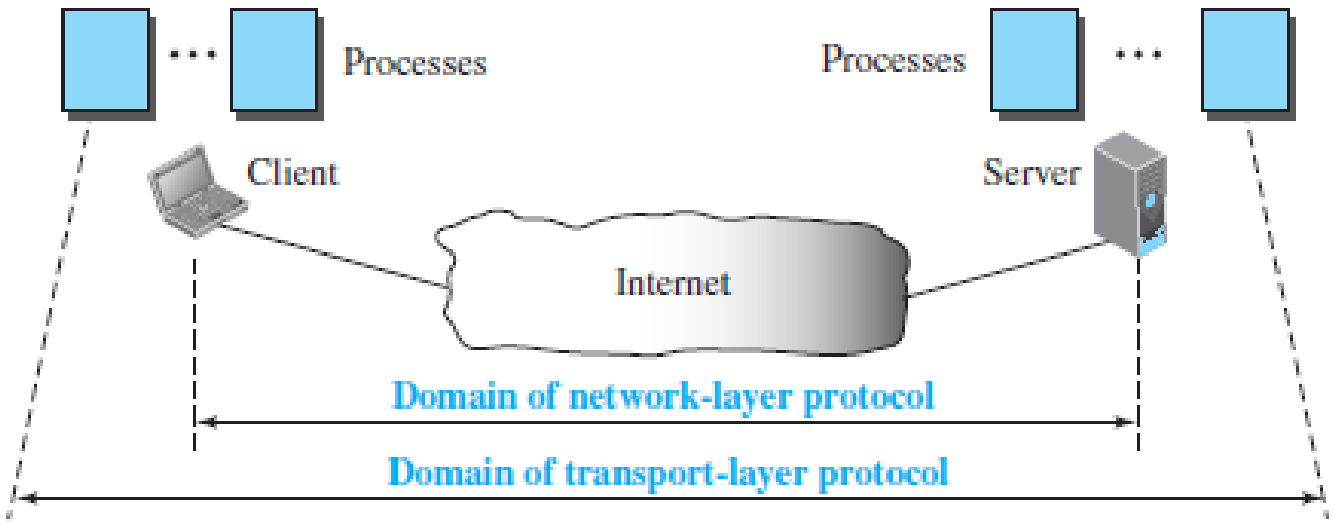
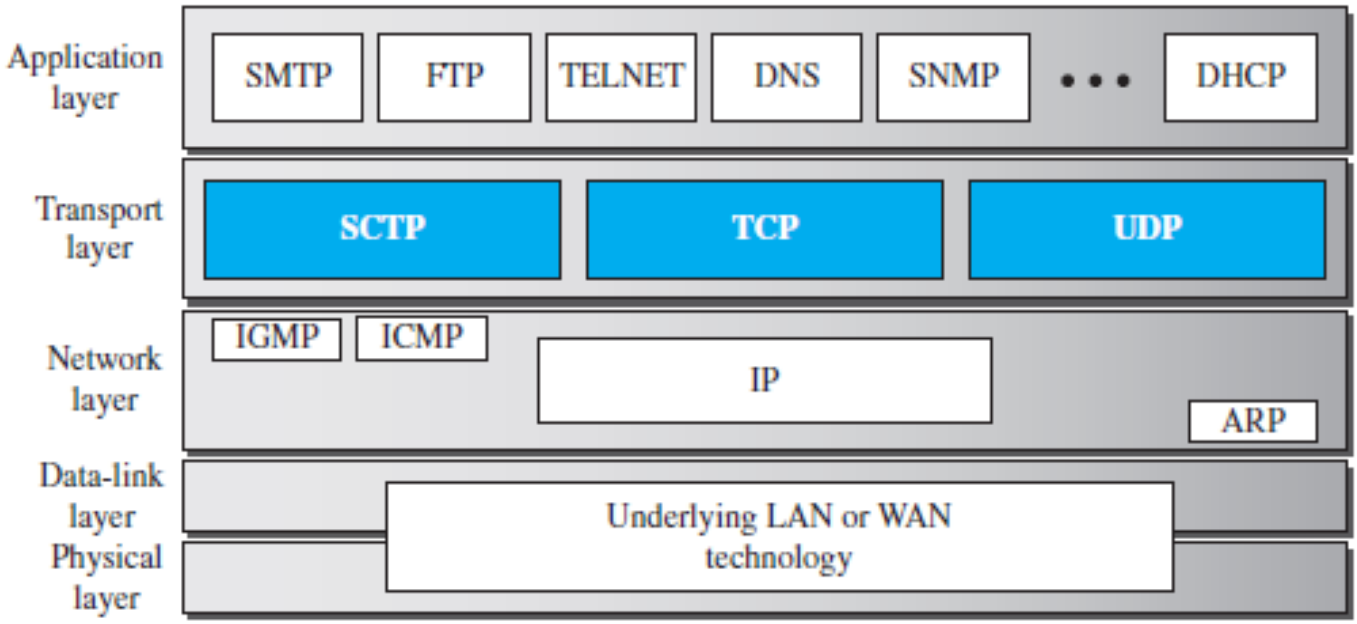
Transport-Layer Services

- ***Process-to-Process Communication:*** A process is an application-layer entity (running program) that uses the services of the transport layer.
- The network layer is responsible for communication at the computer level (host-to-host communication). A network-layer protocol can deliver the message only to the destination computer.
- However, this is an incomplete delivery, as the message still needs to be handed to the correct process.
- A transport-layer protocol is responsible for delivery of the message to the appropriate process. TL provides end to end or process to process communication



Q Which of the following functionalities must be implemented by a transport protocol over and above the network protocol? **(Gate-2003) (1 Marks)**
(A) Recovery from packet losses
(B) Detection of duplicate packets
(C) Packet delivery in the correct order
(D) End to end connectivity
Answer: (D)

- A transport layer protocol can be either connectionless or connection-oriented.
    - A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine.
    - A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data is transferred, the connection is terminated.
- Like the data link layer, the transport layer may be responsible for flow and error control. However, flow and error control at this layer is performed end to end rather than across a single link.
- Reliable Versus Unreliable
    - If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service.
    - On the other hand, if the application program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time applications), then an unreliable protocol can be used.
- In the Internet, there are three common different transport layer protocols.
    - UDP is connectionless and unreliable;
    - TCP and SCTP are connection oriented and reliable. These three can respond to the demands of the application layer programs.
- TCP offers *full-duplex service*, where data can flow in both directions at the same time.

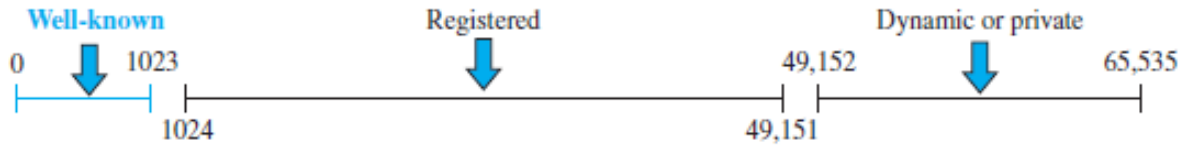


Addressing: Port Numbers

- For communication, we must define the local host, local process, remote host, and remote process.
- Local and Remote host are defined by IP Addresses. To define the processes inside a host, we need second identifiers, called port numbers, they are 16-bits integers ranging from (0 to $2^{16} - 1$) or (0 to 65535).

ICANN Ranges

- ICANN (Internet Corporation for Assigned Names and Numbers) has divided the port numbers into three ranges: well-known, registered, and dynamic (or private).



The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private).

- **Well-known ports:**
  - The ports ranging from 0 to 1023 are assigned and controlled by lANA. These are the well-known ports.
  - The server process must also define itself with a port number. This port number, however, cannot be chosen randomly as the client has to request the data from server.
  - TCP/IP has decided to use universal port numbers for servers; these are called ***well-known port numbers.*** ***well-known port numbers.***

Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	UDP, TCP	DNS
67, 68	UDP	DHCP
69	UDP	TFTP
80	TCP	HTTP (WWW)
110	TCP	POP3
161	UDP	SNMP
443	TCP	HTTPS (SSL)
16384–32767	UDP	RTP-based voice (VoIP) and video

- **Registered ports:**
  - The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.

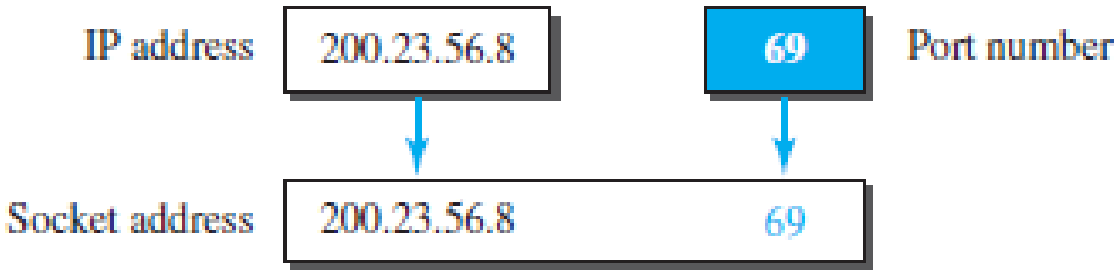
- **Dynamic/**Ephemeral **ports:**
  - The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process.
  - The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host.
  - Ephemeral means "short-lived" and is used because the life of a client is normally short.

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. To use the services of the transport layer in the Internet, we need a pair of socket addresses: the client socket address and the server socket address.
- The combination of an IP address and a port number is called a ***socket address.***

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. To use the services of the transport layer in the Internet, we need a pair of socket addresses: the client socket address and the server socket address.
- The combination of an IP address and a port number is called a ***socket address.***

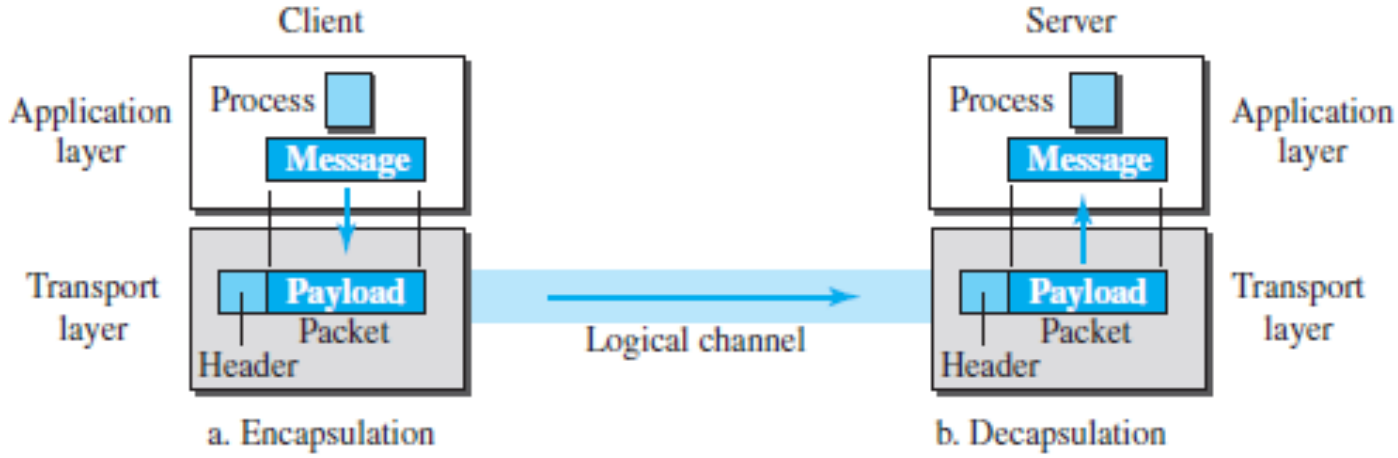


Encapsulation and Decapsulation

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

- 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.

- The transport layer receives the data and adds the transport-layer header. The packets at the transport layer in the Internet are called ***user datagrams, segments,*** or ***packets.***



- 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.

Q What is the maximum size of data that the application layer can pass on to the TCP layer below? **(Gate-2008) (1 Marks)**

(A) Any size
(B) 2^{16} bytes – size of TCP header
(C) 2^{16} bytes
(D) 1500 bytes

Answer: (A)

Q A TCP message consisting of 2100 bytes is passed to IP for delivery across two networks. The first network can carry a maximum payload of 1200 bytes per frame and the second network can carry a maximum payload of 400 bytes per frame, excluding network overhead. Assume that IP overhead per packet is 20 bytes. What is the total IP overhead in the second network for this transmission? **(Gate-2004) (2 Marks)**

(A) 40 bytes **(B)** 80 bytes **(C)** 120 bytes **(D)** 160 bytes

Answer: (C)

TCP (Transmission Control Protocol)

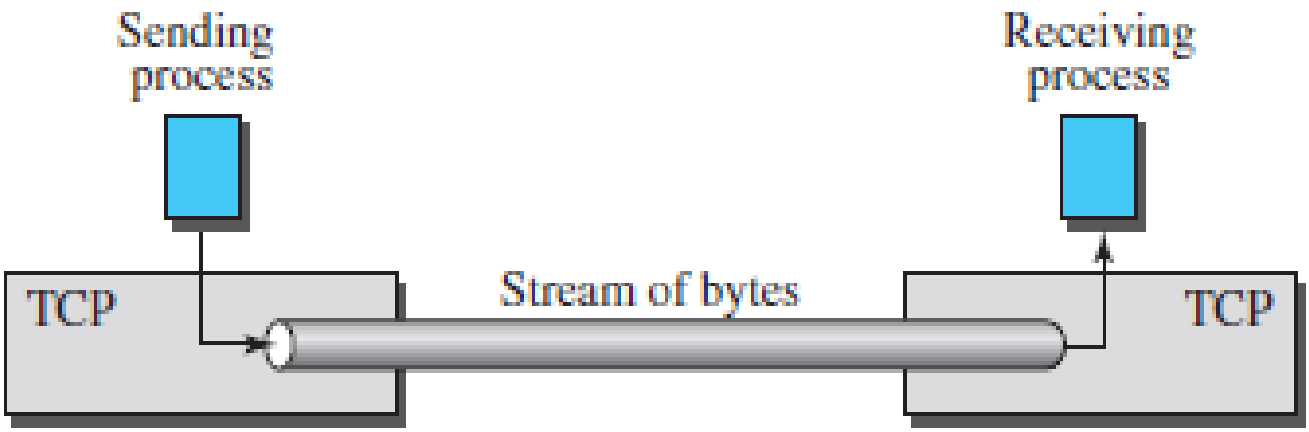
- TCP is a ***reliable connection-oriented protocol***, it must be used in any application where ***reliability is important.***

- It creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

- TCP 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 data across the Internet.

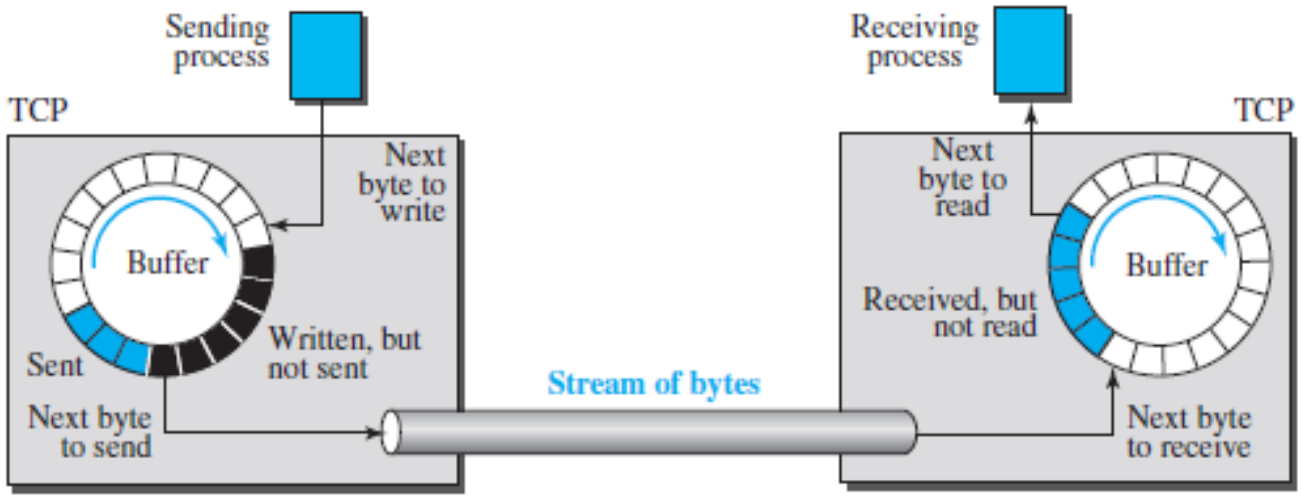
- It adds connection-oriented and reliability features to the services of IP.



Q In TCP, a unique sequence number is assigned to each **(Gate-2004) (1 Marks)**
(A) byte **(B)** word **(C)** segment **(D)** message
Answer: (A)

- Connection oriented means some resources will be reserved at the receiver end, like Bandwidth, CPU time, Buffer etc.

- Sending and Receiving Buffers Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage.

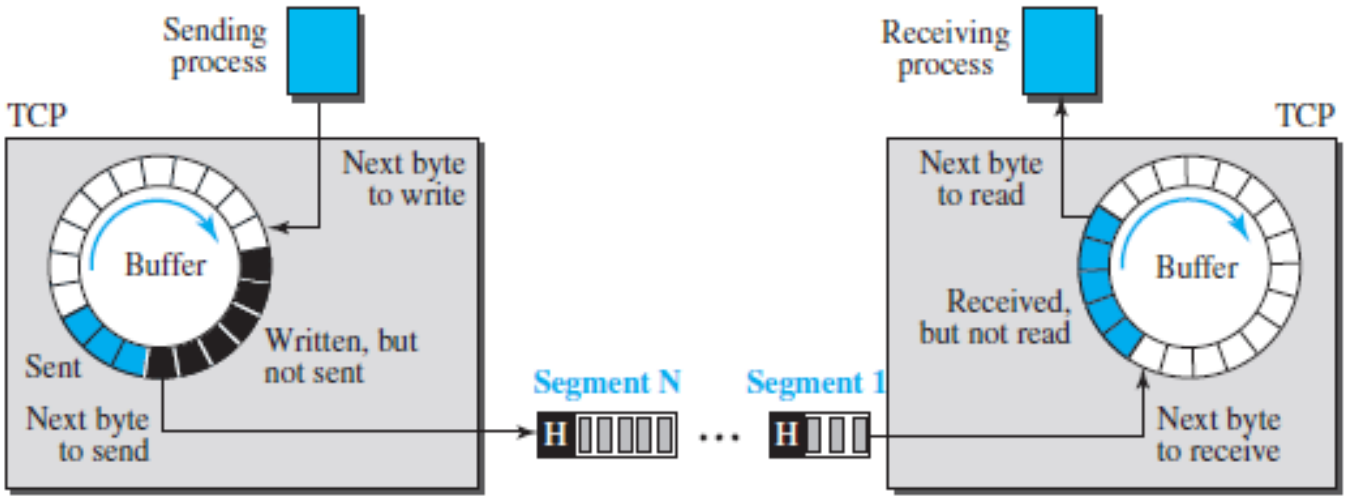


- **Flow Control**
  - The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.
- **Error Control**
  - To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.
- **Congestion Control**
  - TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

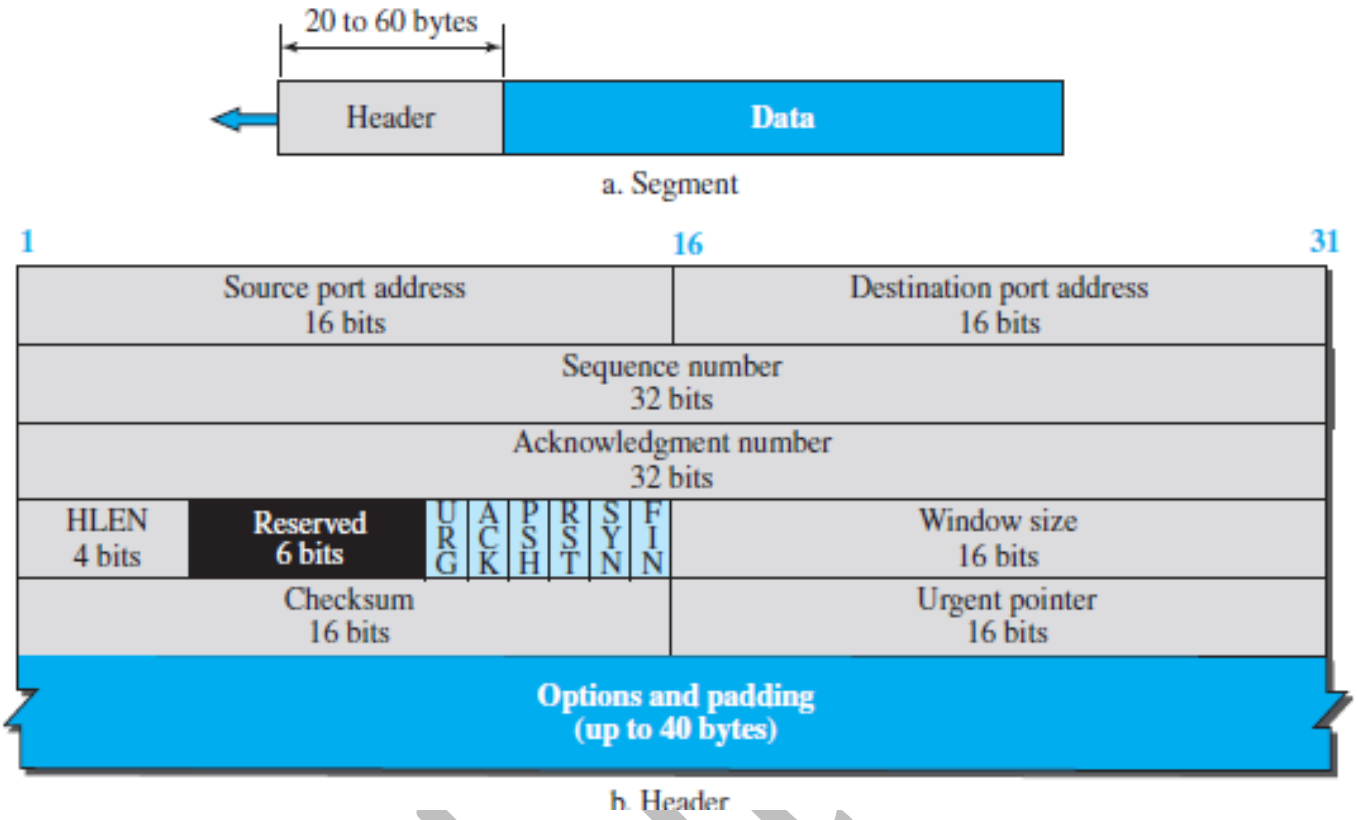
Segments

- The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a *segment*.

- TCP adds a header to each segment (for control purposes) and delivers the segment to the network layer for transmission. The segments are encapsulated in an IP datagram and transmitted.



TCP Header



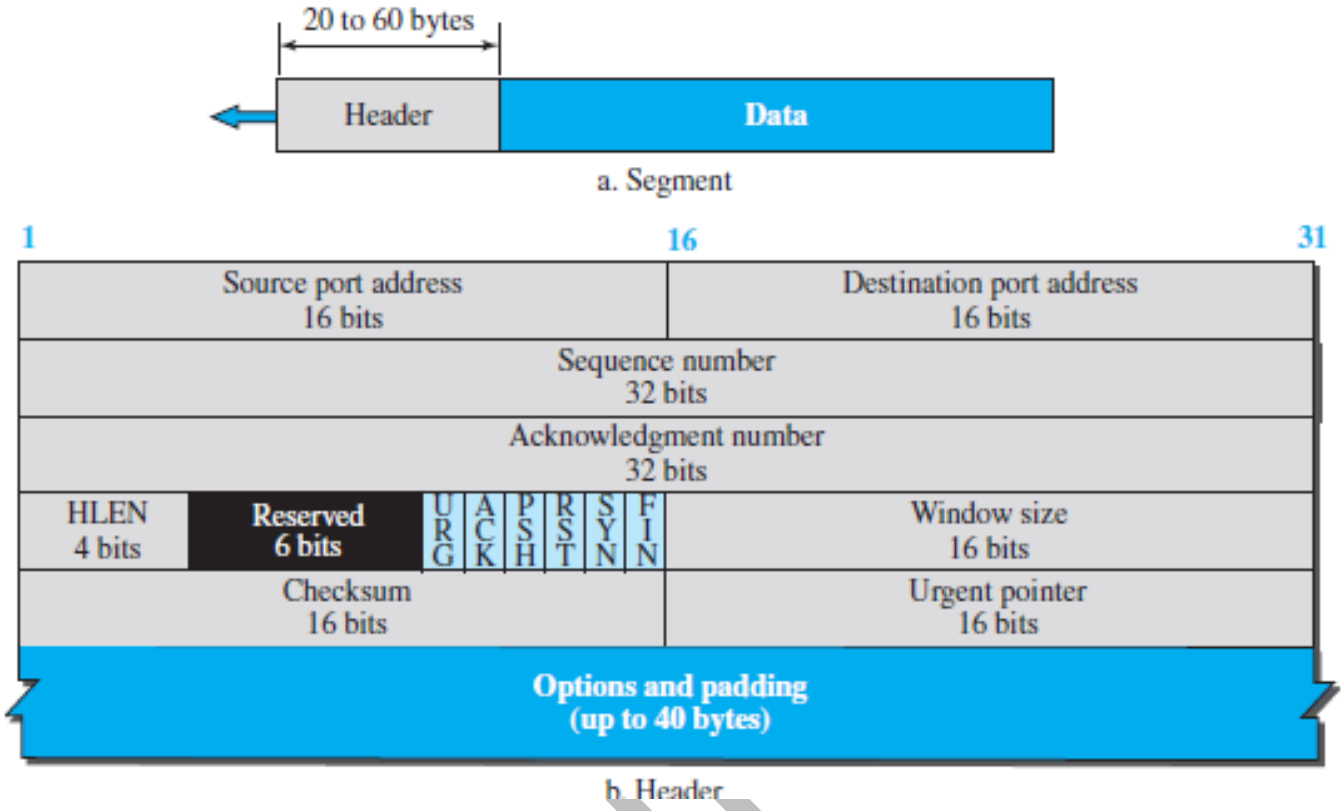
- The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

Q In the TCP/IP protocol suite, which one of the following is NOT part of the IP header? **(Gate-2004) (2 Marks)**

(A) Fragment Offset
(B) Source IP address
(C) Destination IP address
(D) Destination port number

Answer: (D)

Source & Destination port address

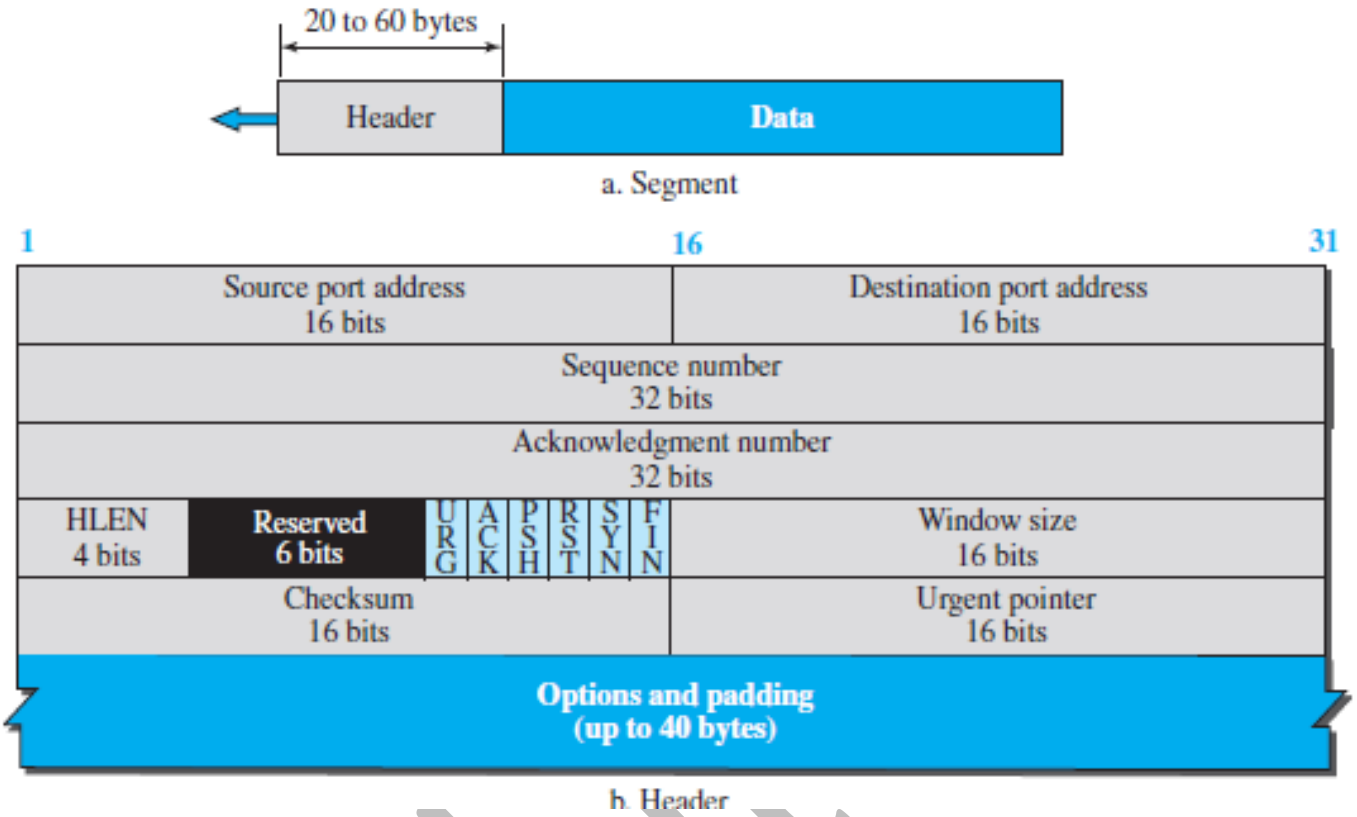


- **Source port address**. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- **Destination port address**. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

Byte Number

- TCP numbers all data bytes (octets) that are transmitted in a connection.
- Numbering is independent in each direction.
- When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them.
- The numbering does not necessarily start from 0.
- TCP chooses an arbitrary number between 0 and $2^{32} - 1$ for the number of the first byte.

Sequence number



- TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. Sequence number is 32-bit field defines the number assigned to the first byte of data contained in this segment.
- So, maximum number of possible sequence numbers = 2^{32}. These sequence numbers lie in the range [0, $2^{32} - 1$].

- In IP every packet is counted not Byte, in DLL every bit is counted with HDLC protocol.

- During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction. Sequence number should be started at random, to remove duplication problem.

- The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes (real or imaginary) carried by the previous segment.

Example: Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

Segment 1	→	Sequence Number:	10001	Range:	10001	to	11000
Segment 2	→	Sequence Number:	11001	Range:	11001	to	12000
Segment 3	→	Sequence Number:	12001	Range:	12001	to	13000
Segment 4	→	Sequence Number:	13001	Range:	13001	to	14000
Segment 5	→	Sequence Number:	14001	Range:	14001	to	15000

- This does not imply that only 2^{32} bytes = 4 GB data can be sent using TCP. The concept of wrap around allows to send unlimited data using TCP.
- After all the 2^{32} sequence numbers are used up and more data is to be sent, the sequence numbers can be wrapped around and used again from the starting.

Wrap Around Time

- Time taken to use up all the 2^{32} sequence numbers is called as **wrap around time**.
- It depends on the bandwidth of the network i.e. the rate at which the bytes go out.

$$\textit{Wrap Around Time} \propto \textit{1 / Bandwidth}$$

If bandwidth of the network = x bytes/sec, then

$$\textit{Wrap Around Time} = 2^{32} / x \textit{ sec.}$$

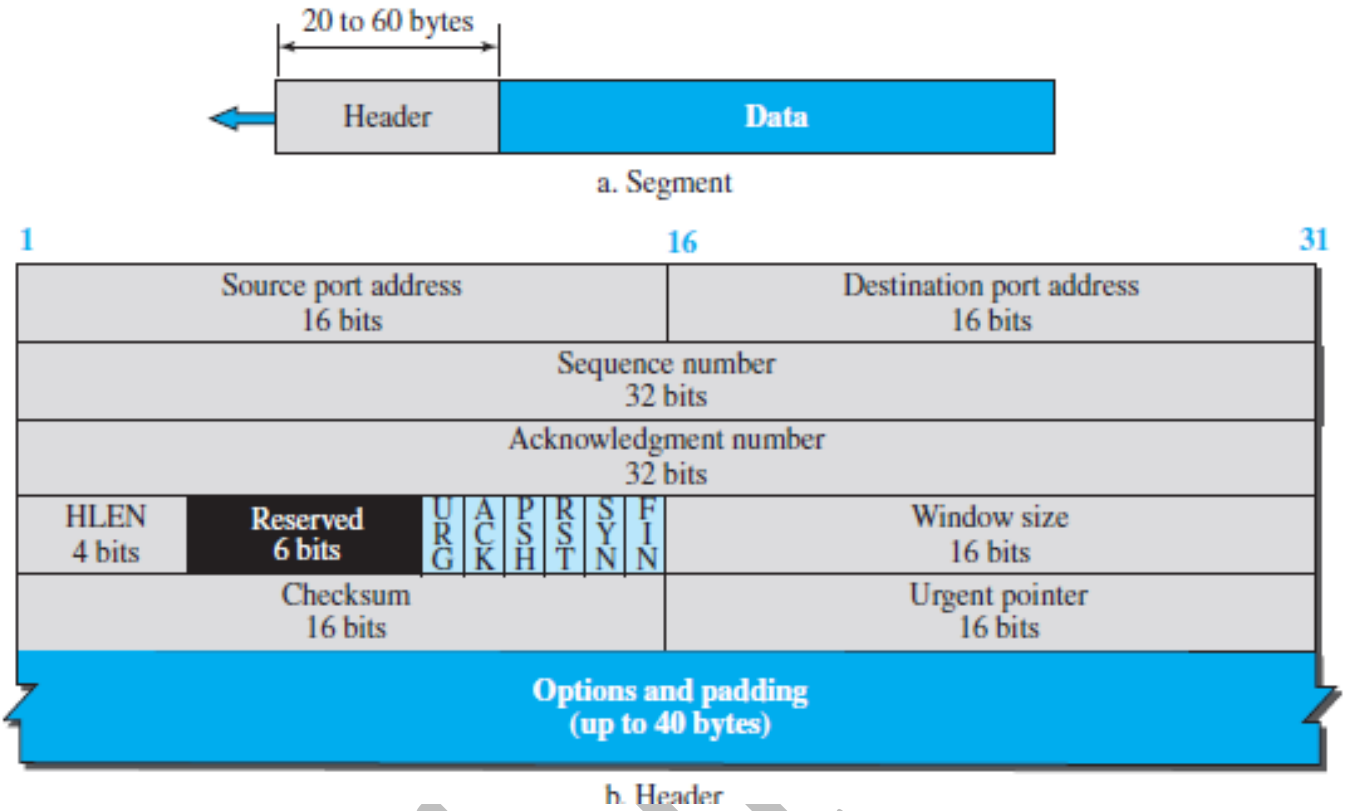
Life Time of TCP Segment

- Life time of a TCP segment is 180 seconds or 3 minutes.
- It means after sending a TCP segment, it might reach the receiver taking 3 minutes in the worst case.

- In the last we will do rap around, wrap around time is the time taken to wrap around
  - if WAT > LT then there is no problem
  - if wat < LT then destination will get same sequence no again and again, to solve this problem additional bits can be put in options, called time-stamp from the time of the clock least significant 32 bits are taken

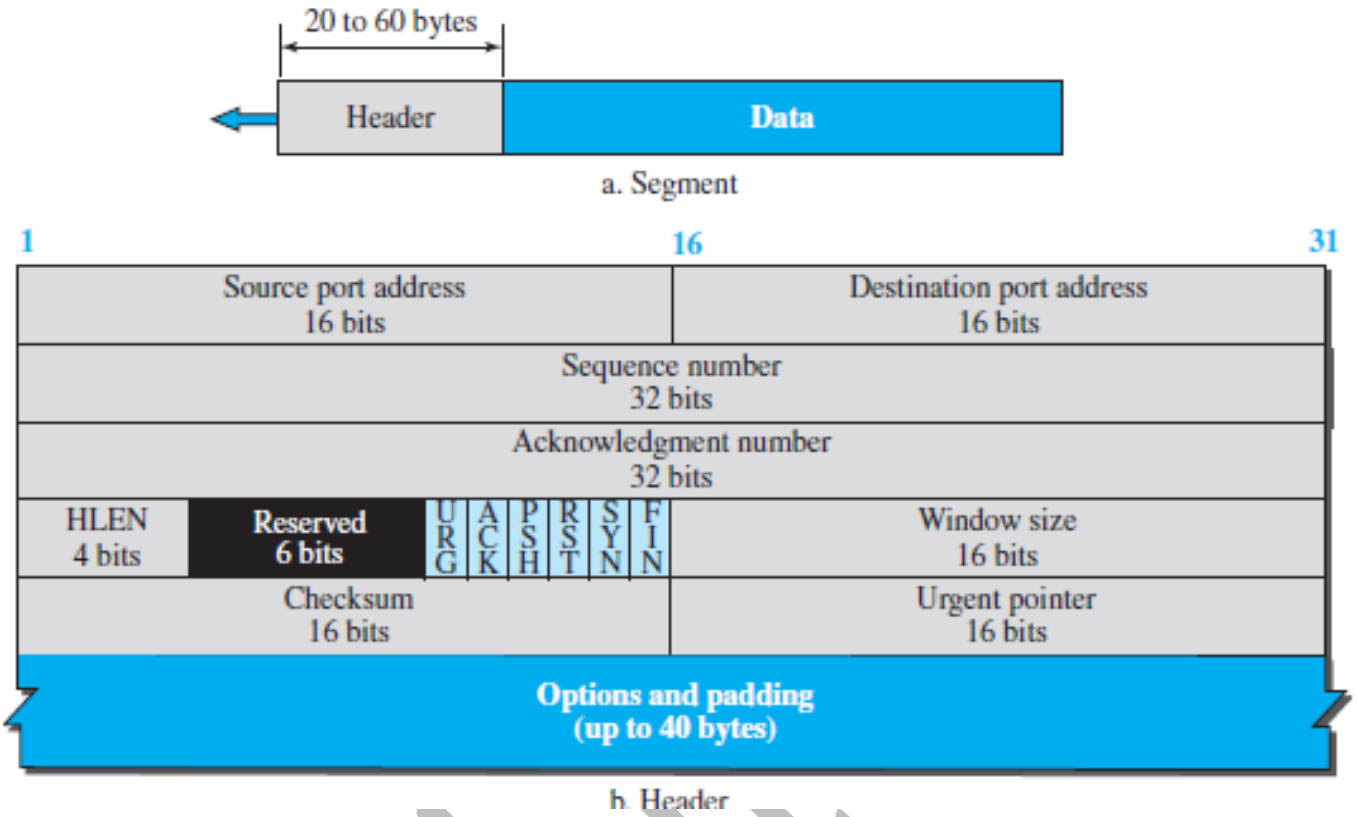
Q Consider a long-lived TCP session with an end-to-end bandwidth of 1 Gbps (= 10^9 bits/second). The session starts with a sequence number of 1234. The minimum time (in seconds, rounded to the closest integer) before this sequence number can be used again is _____. **(GATE-2018) (1 Marks)**

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 defines x + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- The acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.
- Acknowledgment number no can be calculated by subtracting header length of IP and TCP to get the total byte count of the TCP segment and then can find the ack no

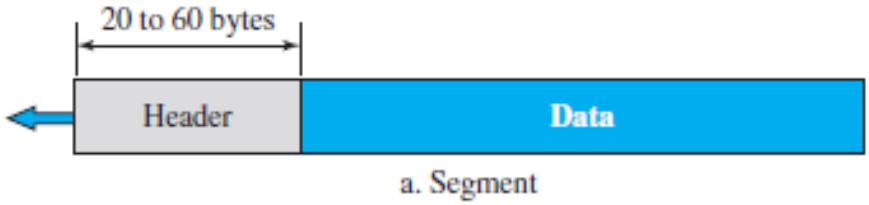
Header length



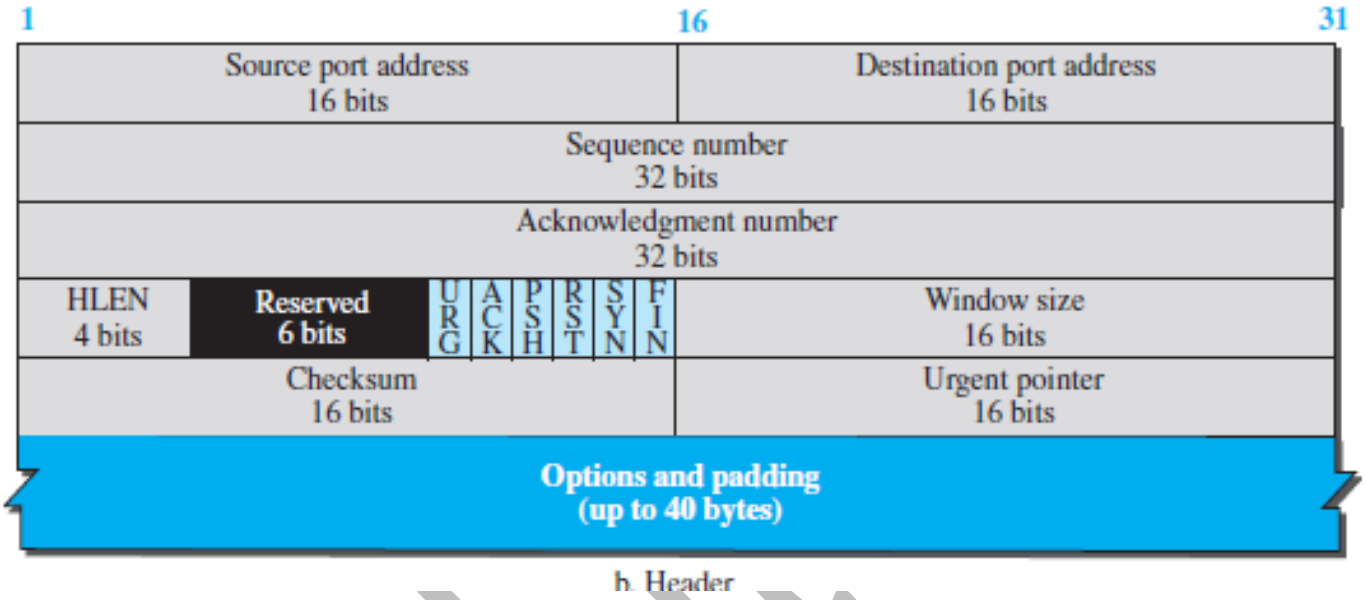
- Header length: This 4-bit field indicates the number of4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- Concept of Scaling Factor
    - *Header length = Header length field value x 4 bytes*

- Reserved. This is a 6-bit field reserved for future use.

Checksum

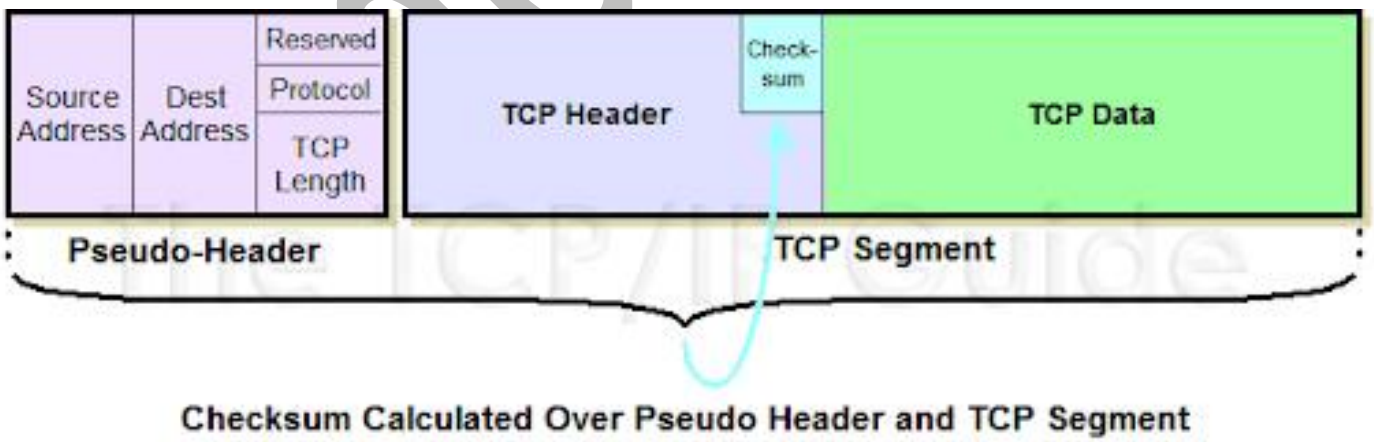


a. Segment



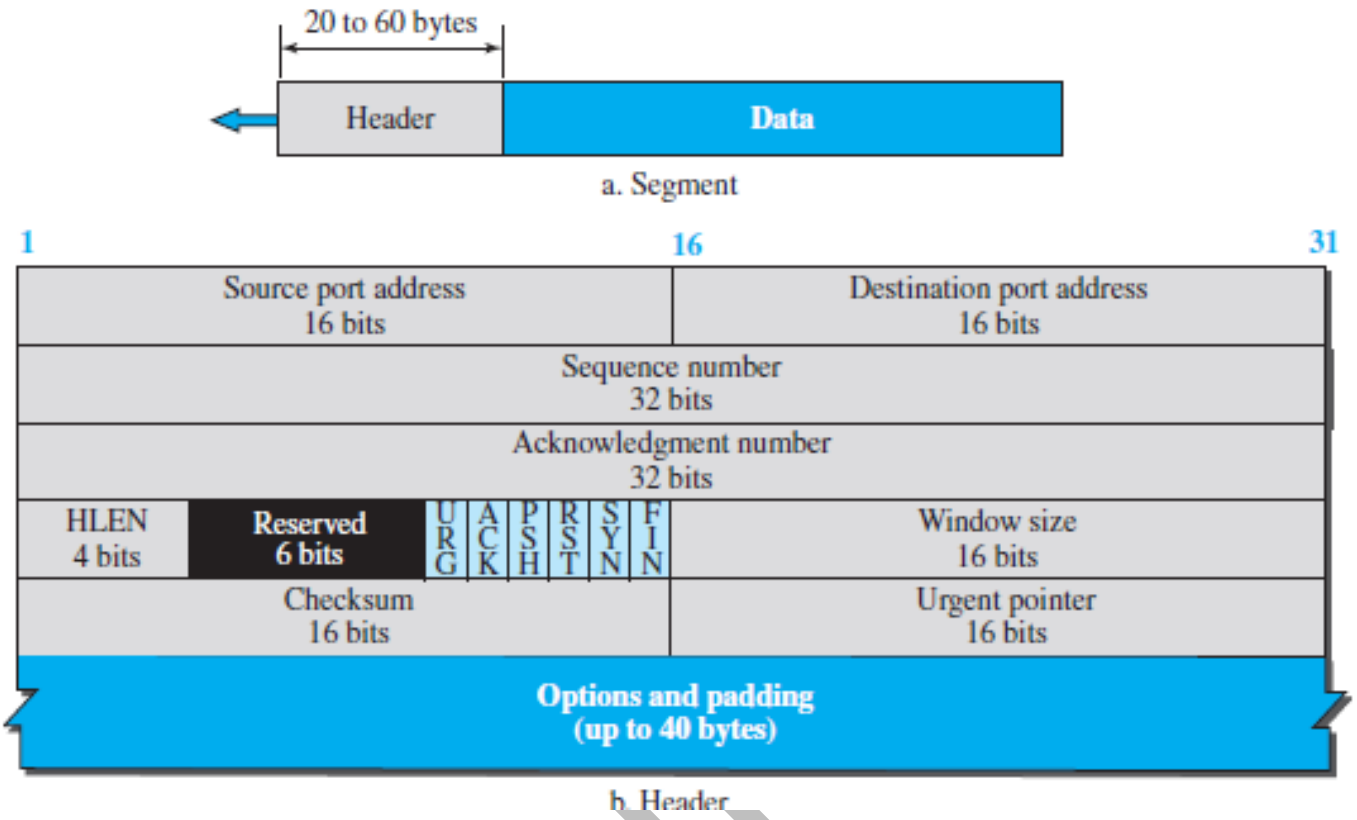
b. Header

- **Checksum:** This 16-bit field contains the checksum.
- While calculation of the checksum for TCP, Entire TCP segment and pseudo header (IP) is considered.
- For the TCP pseudo header, the value for the protocol field is 6.



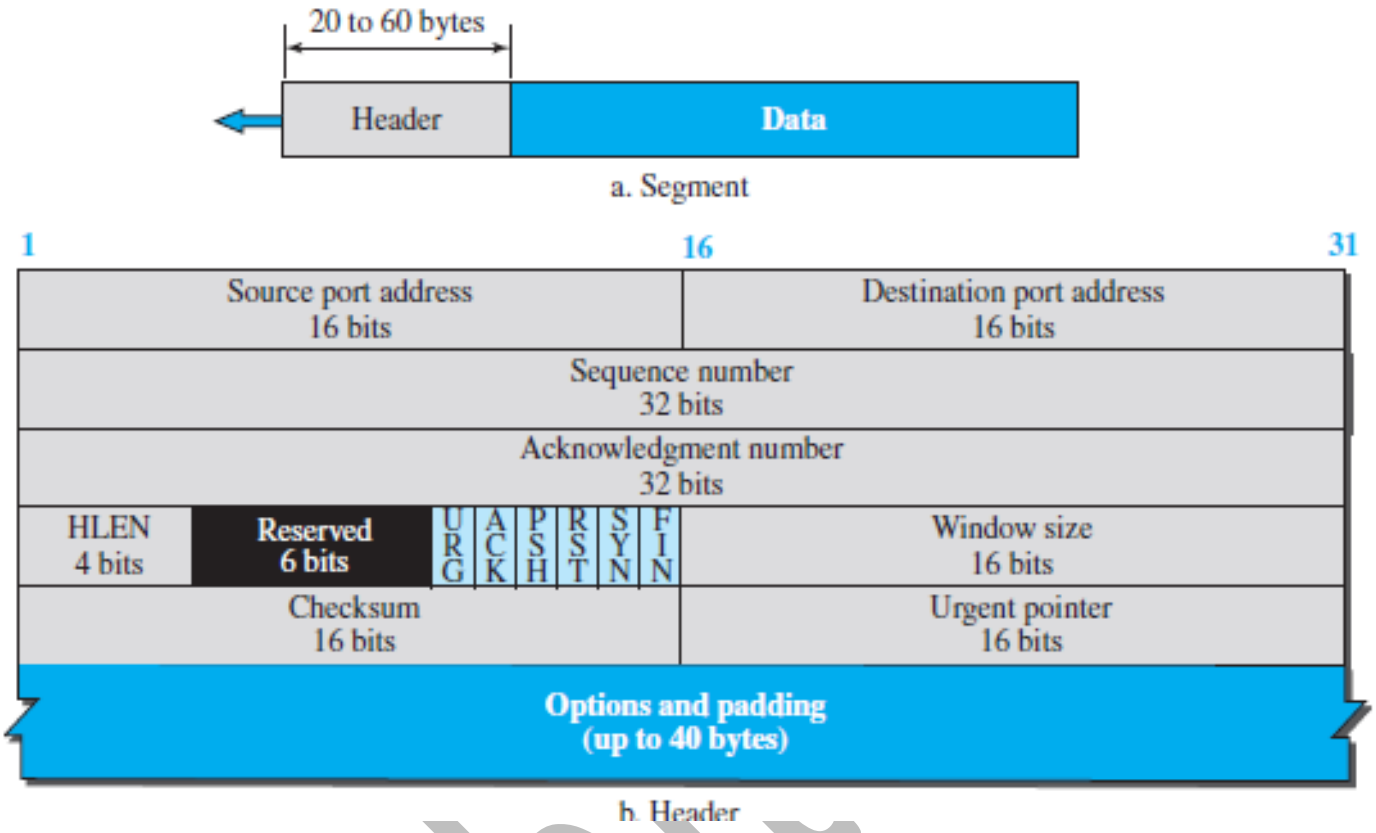
Checksum Calculated Over Pseudo Header and TCP Segment

Window Size



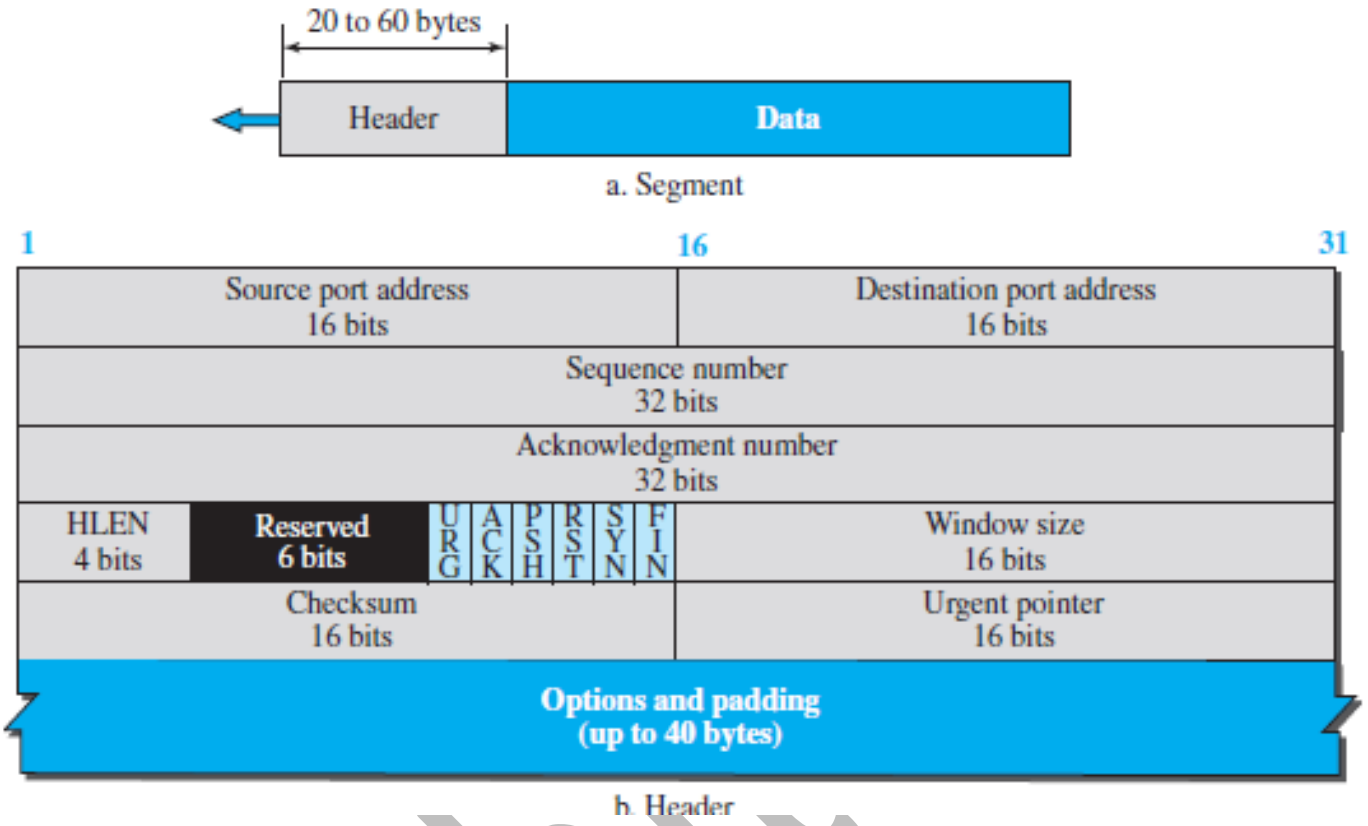
- ***Window size:*** Window size. This field defines the size of the window, in bytes, that the other party must maintain.
- Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

Urgent pointer



- **Urgent pointer:** This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

Control Flag



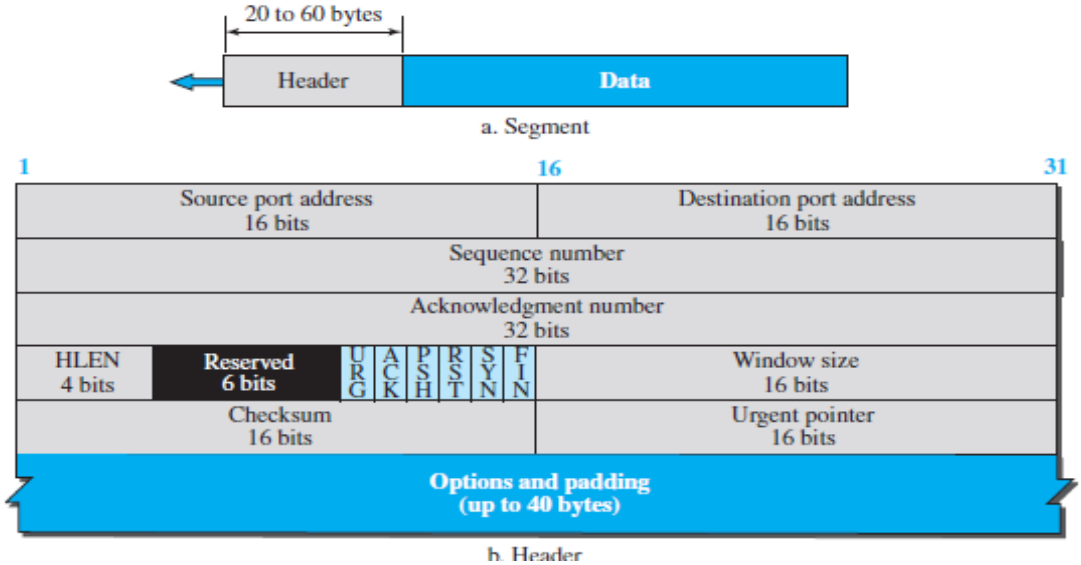
- **Control Flag:** This field defines 6 different control bits or flags. One or more of these bits can be set at a time.
- These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

PUSH Flag

- **Push (PSH) –** Transport layer by default waits for some time for application layer to send enough data equal to maximum segment size so that the number of packets transmitted on network minimizes which is not desirable by some application like interactive applications(chatting).
- Similarly transport layer at receiver end buffers packets and transmit to application layer if it meets certain criteria. This problem is solved by using PSH. Transport layer sets PSH = 1 and immediately sends the segment to network layer as soon as it receives signal from application layer.
- Receiver transport layer, on seeing PSH = 1 immediately forwards the data to application layer. In general, it tells the receiver to process these packets as they are received instead of buffering them.

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

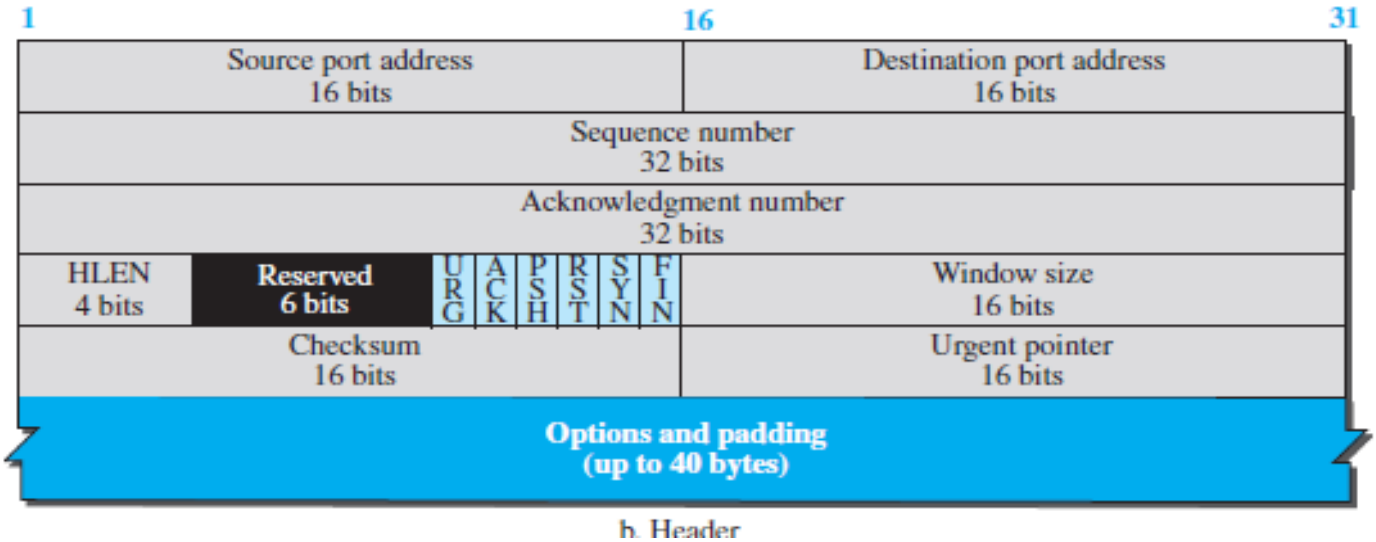


RST Flag

- **Reset (RST) –** It is used to terminate the connection if the sender or receiver feels something is wrong with the TCP connection or that the conversation should not exist.
- It can get send from receiver side when packet is sent to particular host that was not expecting it

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

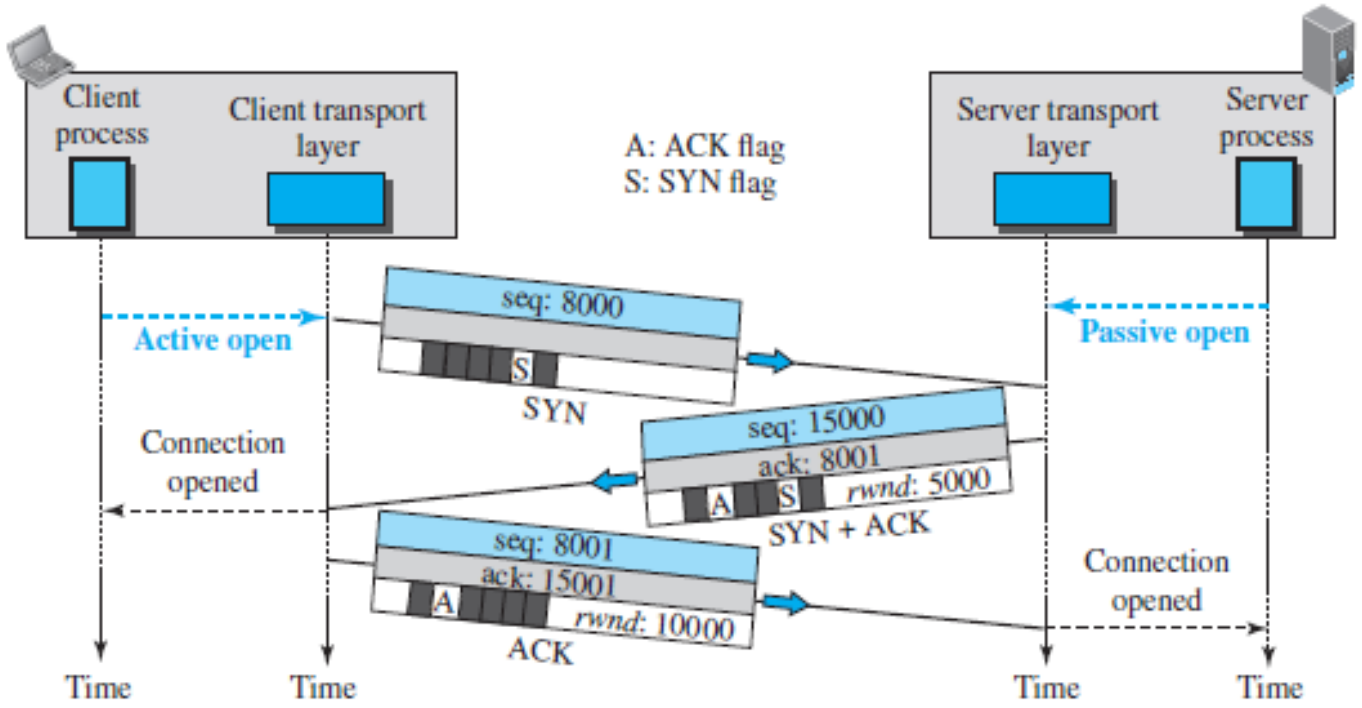
a. Segment



b. Header

A TCP Connection

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



- In the above figure, 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.

Connection Establishment (Three-way handshaking)

- The client sends the first segment, a SYN segment, in which only the SYN flag is set. This segment is for synchronization of sequence numbers.
    - Client sends the ***initial sequence number (ISN).***
    - This segment does not contain an acknowledgment number
    - SYN segment is a control segment and carries no data. However, it consumes one sequence number because it needs to be acknowledged.

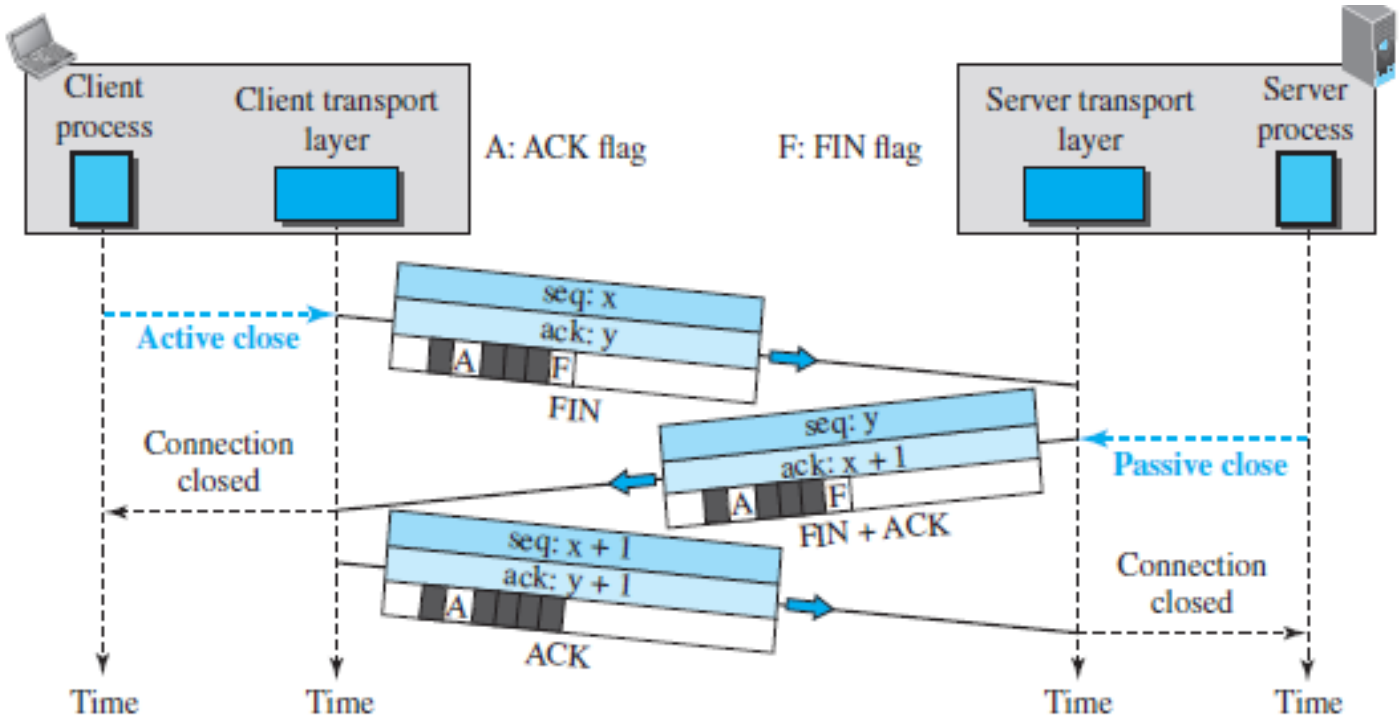
- The server sends the second segment, a SYN + ACK segment with two flag bits set as: SYN and ACK.
    - It is a SYN segment for communication in the other direction.
    - The server uses this segment to initialize a sequence number for numbering the bytes sent from the server to the client.
    - The 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.

- 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.

- ACK segment does not consume any sequence numbers if it does not carry data.
- After connection is established, bidirectional data transfer can take place.

Connection Termination (Three-way handshaking)

- Either of the two parties involved in exchanging data (client or server) can close the connection



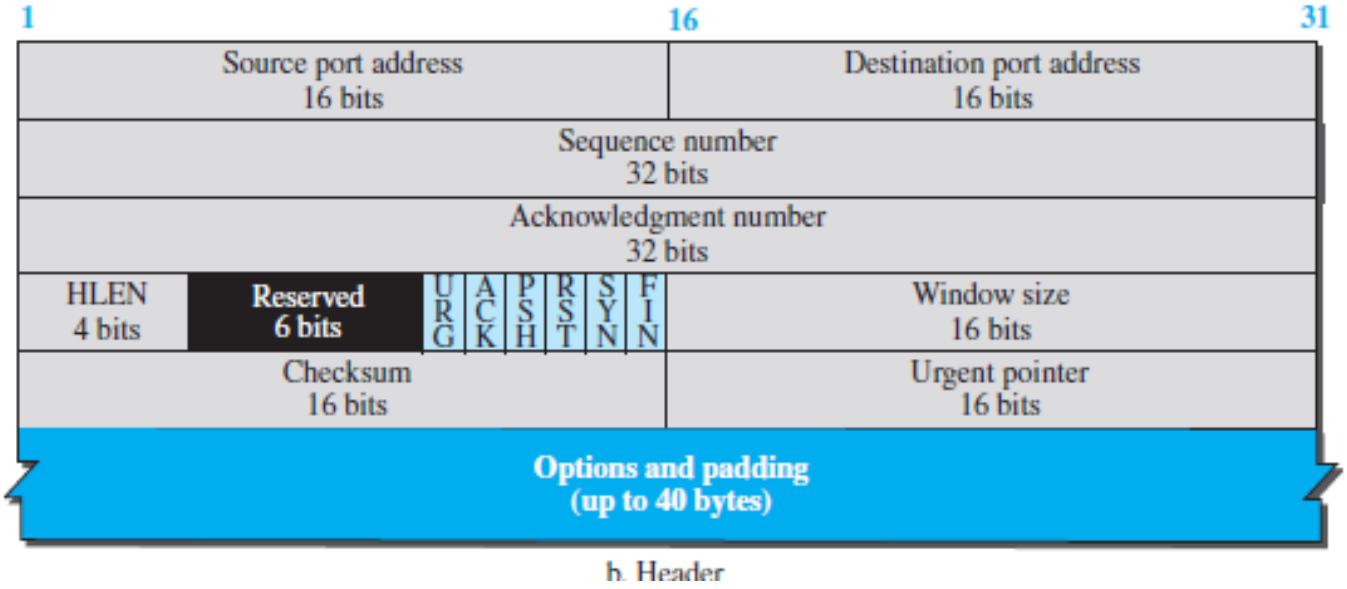
- In this situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.
    - The FIN segment consumes one sequence number if it does not carry data.
- The server TCP, after receiving the FIN segment, informs its 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.
    - If it does not carry data, it consumes only one sequence number because it needs to be acknowledged.
- The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.
    - This segment cannot carry data and consumes no sequence numbers.

States for TCP

State	Description
CLOSED	No connection exists
LISTEN	Passive open received; waiting for SYN
SYN-SENT	SYN sent; waiting for ACK
SYN-RCVD	SYN + ACK sent; waiting for ACK
ESTABLISHED	Connection established; data transfer in progress
FIN-WAIT-1	First FIN sent; waiting for ACK
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out
LAST-ACK	Second FIN sent; waiting for ACK
CLOSING	Both sides decided to close simultaneously

Options

a. Segment



b. Header

- There can be up to 40 bytes of optional information in the TCP header.

Q Suppose two hosts use a TCP connection to transfer a large file. Which of the following statements is/are **False** with respect to the TCP connection? **(Gate-2015) (1 Marks)**

1. If the sequence number of a segment is m, then the sequence number of the subsequent segment is always m+1.
2. If the estimated round-trip time at any given point of time is t sec, the value of the retransmission timeout is always set to greater than or equal to t sec.
3. The size of the advertised window never changes during the course of the TCP connection.
4. The number of unacknowledged bytes at the sender is always less than or equal to the advertised window

(A) 3 only **(B)** 1 and 3 only **(C)** 1 and 4 only **(D)** 2 and 4 only

Answer: (B)

Q While opening a TCP connection, the initial sequence number is to be derived using a time-of-day(ToD) clock that keeps running even when the host is down. The low order 32 bits of the counter of the ToD clock is to be used for the initial sequence numbers. The clock counter increments once per millisecond. The maximum packet lifetime is given to be 64s. Which one of the choices given below is closest to the minimum permissible rate at which sequence numbers used for packets of a connection can increase? **(Gate-2009) (2 Marks)**

(A) 0.015/s **(B)** 0.064/s **(C)** 0.135/s **(D)** 0.327/s

Answer: (A)

SYN Flooding Attack

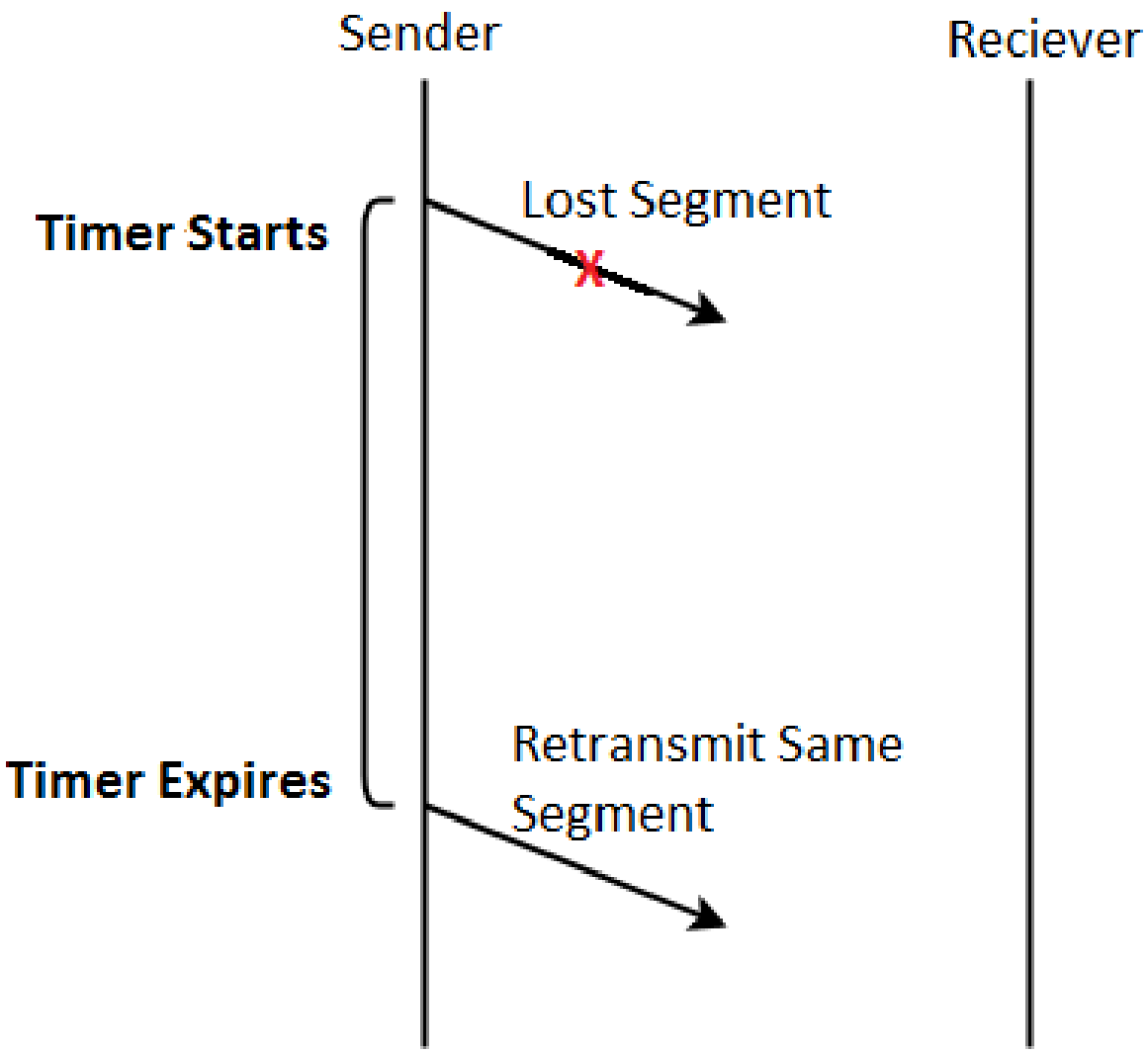
- When one or more malicious attackers send a large number of SYN segments to a server pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams.
- The server allocates the necessary resources, such as creating transfer control block (TCB) tables and setting timers.
- TCP server then sends the SYN + ACK segments to the fake clients, which are lost.
- The server waits for the third leg of the handshaking process and resources are allocated without being used.
- During this short period of time, if the number of SYN segments is large, the server eventually runs out of resources and may be unable to accept connection requests from valid clients.
- SYN flooding attack belongs to denial of service attack.

TCP Retransmission

- After establishing the connection, Sender starts transmitting TCP segments to the receiver. A TCP segment sent by the sender may get lost on the way before reaching the receiver.
- This causes the receiver to send the acknowledgement with same ACK number to the sender. As a result, sender retransmits the same segment to the receiver. This is called as **TCP retransmission**.
- Sender discovers that the TCP segment is lost when
    - Either Time Out Timer expire or it receives three duplicate acknowledgements

Retransmission after Time out Timer Expiry

- Each time sender transmits a TCP segment to the receiver, it starts a Time Out Timer. Following two cases are possible
    - Sender receives an acknowledgement for the sent segment before the timer goes off. In this case, sender stops the timer.
    - Sender does not receive any acknowledgement for the sent segment and the timer goes off. In this case, sender assumes that the sent segment is lost. Sender retransmits the same segment to the receiver and resets the timer.



Retransmission After Receiving 3 Duplicate Acknowledgements/ Early Retransmission

- Consider sender receives three duplicate acknowledgements for a TCP segment sent by it. Then, sender assumes that the corresponding segment is lost.
- So, sender retransmits the same segment without waiting for its time out timer to expire. This is known as ***early retransmission*** or Fast retransmission.

Example:

Consider Sender sends 5 TCP segments to the receiver. The second TCP segment gets lost before reaching the receiver. The sequence of steps that will take place are

- On receiving segment-1, receiver sends acknowledgement asking for segment-2 next. (Original ACK)
- On receiving segment-3, receiver sends acknowledgement asking for segment-2 next. (1st duplicate ACK)
- On receiving segment-4, receiver sends acknowledgement asking for segment-2 next. (2nd duplicate ACK)
- On receiving segment-5, receiver sends acknowledgement asking for segment-2 next. (3rd duplicate ACK)
- Now, Sender receives 3 duplicate acknowledgements for segment-2 in total. So, sender assumes that the segment-2 is lost. So, it retransmits segment-2 without waiting for its timer to go off.

Points to Note

- In case time out timer expires before receiving the acknowledgement for a TCP segment, then there is a strong possibility of congestion in the network.
- Retransmission on receiving 3 duplicate acknowledgements is a way to improve the performance over retransmission on time out.
- TCP uses SR (80%) and GBN (20%) both, as Ws =Wr (SR) out of order packets will be accepted and in GBN use cumulative acknowledgement
- Question is why only 3 duplicate ack, experimentally it is found out that this works best.

Q Consider a TCP connection in a state where there are no outstanding ACKs. The sender sends two segments back to back. The sequence numbers of the first and second segments are 230 and 290 respectively. The first segment was lost but the second segment was received correctly by the receiver.

Let X be the amount of data carried in the first segment (in bytes) and Y be the ACK number sent by the receiver. The values of X and Y are

Ans. Sequence number of 1st segment = 230 and Sequence number of 2nd segment = 290

Range of sequence numbers contained in the 1st segment = [230,289].

Total number of sequence numbers contained in the 1st segment = 289 – 230 + 1 = 60.

TCP assigns 1 sequence number to 1 byte of data, so total data is ***60 Bytes.***

On receiving the 2nd segment, Receiver sends the acknowledgement asking for the first segment only. This is because it expects the 1st segment first. Thus, ***Acknowledgement number = Sequence number of the 1st segment = 230.***

Q Consider a TCP client and a TCP server running on two different machines. After completing data transfer, the TCP client calls *close* to terminate the connection and a FIN segment is sent to the TCP server. Server-side TCP responds by sending an ACK, which is received by the client-side TCP. As per the TCP connection state diagram (RFC 793), in which state does the client-side TCP connection wait for the FIN from the server-side TCP? **(GATE-2017) (1 Marks)**

(a) LAST-ACK
(b) TIME-WAIT
(c) FIN-WAIT-1
(d) FIN-WAIT-2

Ans: d

Q Assume that the bandwidth for a TCP connection is 10,48,560 bits/sec. Let α be the value of RTT in milliseconds (rounded off to the nearest integer) after which the TCP window scale option is needed. Let β be the maximum possible window size with window scale option. Then the values of α and β are. **(Gate-2015) (2 Marks)**

(A) 63 milliseconds 65535×2^{14}
(B) 63 milliseconds 65535×2^{16}
(C) 500 milliseconds 65535×2^{14}
(D) 500 milliseconds 65535×2^{16}

Answer: (C)

Q Consider the following statements. **(Gate-2015) (1 Marks)**

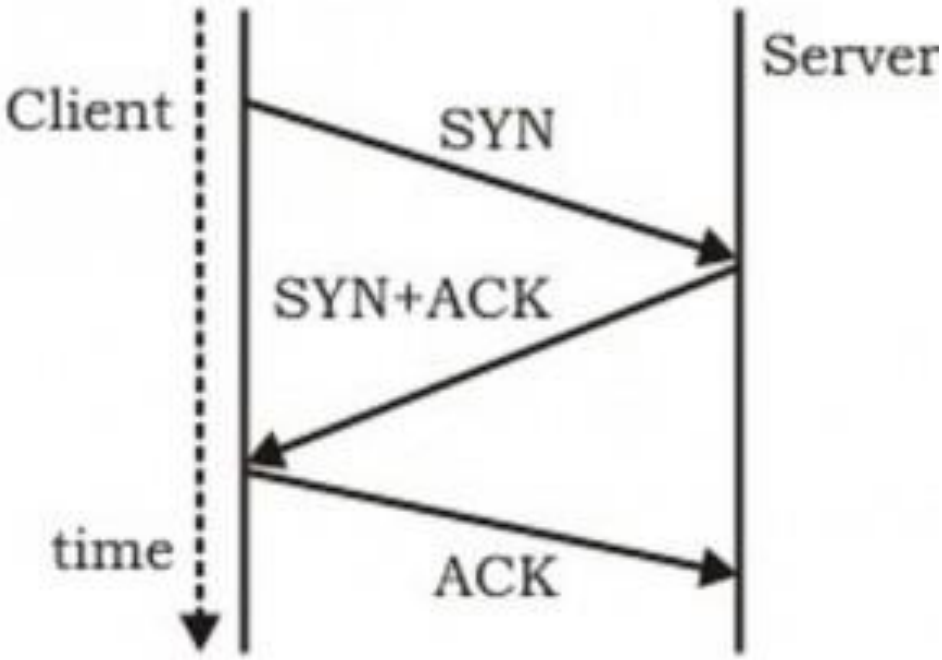
I. TCP connections are full duplex.

II. TCP has no option for selective acknowledgment

III. TCP connections are message streams.

(A) Only I is correct
(B) Only I and II are correct
(C) Only II and III are correct
(D) All of I, II and III are correct
Answer: (A)

Q The three-way handshake for TCP connection establishment is shown below.



Which of the following statements are TRUE?
(S1) Loss of SYN + ACK from the server will not establish a connection
(S2) Loss of ACK from the client cannot establish the connection
(S3) The server moves LISTEN → SYN_RCVD → SYN_SENT → ESTABLISHED in the state machine on no packet loss
(S4) The server moves LISTEN → SYN_RCVD → ESTABLISHED in the state machine on no packet loss. **(Gate-2008) (2 Marks)**
(A) S2 and S3 only **(B)** S1 and S4 **(C)** S1 and S3 **(D)** S2 and S4
Answer: (B)

Q Consider a TCP connection in a state where there are no outstanding ACKs. The sender sends two segments back to back. The sequence numbers of the first and second segments are 230 and 290 respectively. The first segment was lost, but the second segment was received correctly by the receiver. Let X be the amount of data carried in the first segment

(in bytes), and Y be the ACK number sent by the receiver. The values of X and Y (in that order) are **(Gate-2007) (1 Marks)**

(A) 60 and 290
(B) 230 and 291
(C) 60 and 231
(D) 60 and 230

Answer: (D)

Congestion Control

- **Congestion:** Congestion refers to a network state where, the message traffic becomes so heavy that it slows down the network response time.
- Congestion control refers to techniques and mechanisms that can: Either prevent congestion before it happens or remove congestion after it has happened
    - TCP reacts to Congestion by reducing the sender window size.
    - TCP uses a combination of GBN and SR protocols to provide reliability.

Windows in TCP

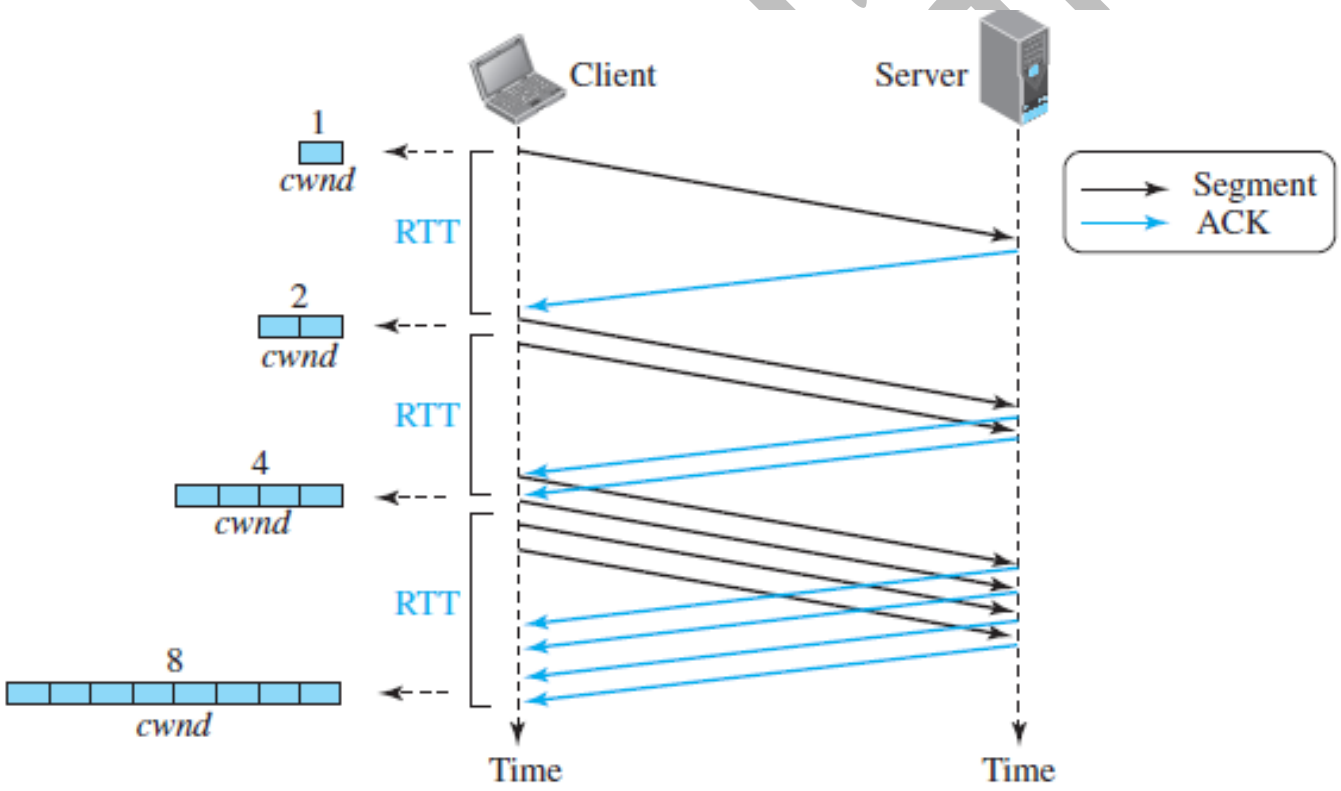
- TCP uses two windows (send window and receive window) for each direction of data transfer, i.e. four windows for a bidirectional communication.
- ***Send Window***
    - The size of the sender window is determined by the following two factors
    - **Receiver window size and Congestion window size.**
- ***Receive Window***
    - Sender should not send data greater than receiver window size. Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
    - So, sender should always send data less than or equal to receiver window size. Receiver dictates its window size to the sender through TCP Header.
- **Congestion Window**
    - Sender should not send data greater than congestion window size. Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
    - So, sender should always send data less than or equal to congestion window size.
    - Different variants of TCP use different approaches to calculate the size of congestion window. Congestion window is known only to the sender and is not sent over the links.
- **In general, Sender window size = Minimum (Receiver window size, Congestion window size)**

TCP Congestion Policy

- TCP's general policy for handling congestion consists of following three phases
  - Slow Start (Exponential Increase)
  - Congestion Avoidance (Additive Increase)
  - Congestion Detection

Slow Start Phase (Exponential Increase)

- Initially, sender sets congestion window size = Maximum Segment Size (1 MSS).
- After receiving each acknowledgment, the size of congestion window increases exponentially.



- After 1 round trip time, congestion window size = $(2)^1$ = 2 MSS
- After 2 round trip time, congestion window size = $(2)^2$ = 4 MSS
- After 3 round trip time, congestion window size = $(2)^3$ = 8 MSS and so on.
- This phase continues until the congestion window size reaches the slow start threshold.
- **Threshold = Maximum number of TCP segments that receiver window can accommodate / 2 = (Receiver window size / Maximum Segment Size) / 2**

Q Consider the following statements regarding the slow start phase of the TCP congestion control algorithm. Note that *cwnd* stands for the TCP congestion window and MSS denotes the Maximum Segment Size.

(i) The *cwnd* increase by 2 MSS on every successful acknowledgement.

(ii) The *cwnd* approximately doubles on every successful acknowledgement.

(iii) The *cwnd* increase by 1 MSS every round-trip time.

(iv) The *cwnd* approximately doubles every round-trip time.

Which one of the following is correct? **(GATE-2018) (1 Marks)**

(a) Only (ii) and (iii) are true

(b) Only (i) and (iii) are true

(c) Only (iv) is true

(d) Only(i) and (iv) is true

Q In the slow start phase of the TCP congestion algorithm, the size of the congestion window: **(Gate-2008) (2 Marks)**

a) does not increase

b) increase linearly

c) increases quadratically

d) increases exponentially

Congestion Avoidance Phase

- After reaching the threshold, Sender increases the congestion window size linearly to avoid the congestion.
- On receiving each acknowledgement, sender increments the congestion window size by 1.
- **Congestion window size = Congestion window size + 1,** This phase continues until the congestion window size becomes equal to the receiver window size.

Congestion Detection Phase

- When sender detects the loss of segments, it reacts in different ways depending on how the loss is detected
- **Detection On Time Out**
    - Time Out Timer expires before receiving the acknowledgement for a segment. It suggests the strong possibility of congestion in the network. There are chances that a segment has been dropped in the network.
    - **Reaction:** In this case, sender reacts by
        - Setting the slow start threshold to half of the current congestion window size.
        - Decreasing the congestion window size to 1 MSS.
        - Resuming the slow start phase.
- **Detection On Receiving 3 Duplicate Acknowledgements**
    - Sender receives 3 duplicate acknowledgements for a segment. This case suggests the weaker possibility of congestion in the network. There are chances that a segment has been dropped but few segments sent later may have reached.
    - **Reaction**
        - In this case, sender reacts by setting the slow start threshold to half of the current congestion window size.
        - Decreasing the congestion window size to slow start threshold.
        - Resuming the congestion avoidance phase.

Q Let the size of congestion window of a TCP connection be 32 KB when a timeout occurs. The round-trip time of the connection is 100 msec and the maximum segment size used is 2 KB. The time taken (in msec) by the TCP connection to get back to 32 KB congestion window is _____. **(Gate-2014) (2 Marks)**
ANSWER 1100 to 1300

Q Consider an instance of TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm where the window size at the start of the slow start phase is 2 MSS and the threshold at the start of the first transmission is 8 MSS. Assume that a timeout occurs during the fifth transmission. Find the congestion window size at the end of the tenth transmission. **(Gate-2012) (2 Marks)**
a) 8 MSS **b)** 14 MSS **c)** 7 MSS **d)** 12 MSS
ANSWER 3

Q On a TCP connection, current congestion window size is Congestion Window = 4 KB. The window size advertised by the receiver is Advertise Window = 6 KB. The last byte sent by the sender is LastByteSent = 10240 and the last byte acknowledged by the receiver is LastByteAcked = 8192. The current window size at the sender is **(Gate-2005) (2 Marks)**
(A) 2048 bytes **(B)** 4096 bytes **(C)** 6144 bytes **(D)** 8192 bytes
Answer: (B)

Q Suppose that the maximum transmit window size for a TCP connection is 12000 bytes. Each packet consists of 2000 bytes. At some point of time, the connection is in slow-start phase with a current transmit window of 4000 bytes. Subsequently, the transmitter receives two acknowledgements. Assume that no packets are lost and there are no time-outs. What is the maximum possible value of the current transmit window? **(Gate-2004) (2 Marks)**
(A) 4000 bytes **(B)** 8000 bytes **(C)** 10000 bytes **(D)** 12000 bytes
Answer: (B)

Q Which one of the following statements is FALSE? **(Gate-2004) (1 Marks)**
(A) TCP guarantees a minimum communication rate
(B) TCP ensures in-order delivery
(C) TCP reacts to congestion by reducing sender window size
(D) TCP employs retransmission to compensate for packet loss
Answer: (A)

Timer

- Time-wait timer (Take care of late packets)
    - never close connection immediately, other wise the port no will be available for some other process, generally we wait for 2*LT. If some packet arrives late then there will a problem.
- Keep-alive timer
    - Sever periodically check connection and close them.
    - after keep-alive timer sends 10 probe messages with a gap of 75 seconds and in case of no reply, will close the connection.
- Persistent timer
    - Window size zero advertise
- Acknowledgement time
    - ack timer is used to generate cumulative ack and piggybacking ack
- Time-out timer
    - will be discussed in detail in next section.

Network Traffic And Time Out Timer

- TCP uses a time out timer for retransmission of lost segments.
- The value of time out timer is dynamic and changes with the amount of traffic in the network.
- Consider Receiver has sent the ACK to the sender and the ACK is on its way through the network. Now, following two cases are possible
    - **High traffic:** If there is high traffic in the network, the time taken by the ACK to reach the sender will be more. So, as per the high traffic, the value of time out timer should be kept large.
        1. ***If the value is kept small***, then timer will time out soon. It causes the sender to assume that the segment is lost before reaching the receiver. However, in actual the ACK is delayed due to high traffic. Sender keeps retransmitting the same segment. This overburdens the network and might lead to congestion.
    - **Low traffic**: If there is low traffic in the network, the time taken by the ACK to reach the sender will be less. So, as per the low traffic, the value of time out timer should be kept small.
        1. ***If the value is kept large***, Timer will not time out soon. Sender keeps waiting for the ACK even when it is actually lost. This causes excessive delay.

- **Conclusion:** The setting of the time-out timer is very important if we want to use the network efficiently, and the value of the timer must change based on the change in the network scenario.

Algorithms for Computing Time Out Timer Value

- The algorithms used for computing the value of time out timer dynamically are-
  1. Basic Algorithm
  2. Jacobson's Algorithm
  3. Karn's modification

General Rules for Algorithms (Basic algorithm)

- **Rule-01**
  - The value of time out timer for the next segment is increased when Actual round-trip time for the previous segment is found to be increased indicating there is high traffic in the network.
- **Rule-02**
  - The value of time out timer for the next segment is decreased when Actual round-trip time for the previous segment is found to be decreased indicating there is low traffic in the network.

- **Basic Algorithm** The steps followed under Basic Algorithm are-
- **Step-01: Sending 1st Segment**
  - Sender assumes any random value of initial RTT say $IRTT_1$.
  - So, after sending the 1st segment, sender expects its ACK to arrive in time $IRTT_1$.
  - Sender sets time out timer value (TOT) for the 1st segment to be- $\mathbf{TOT_1 = 2\ X\ IRTT_1}$
  - Suppose ACK for the 1st segment arrives in time $ARTT_1$. Here, $ARTT_1$ = Actual Round-Trip Time for the 1st segment.
- **Step-02: Sending 2nd Segment**
  - Sender computes the value of initial RTT for the 2nd segment using the relation $\mathbf{IRTT_{n+1} = \alpha\ IRTT_n + (1 - \alpha)\ ARTT_n}$
  - Here, α is called smoothing factor where 0 <= α <= 1 (Its value will be given in questions)
  - Substituting n=1, sender gets $IRTT_2 = \alpha\ IRTT_1 + (1 - \alpha)\ ARTT_1$.

- So, after sending the 2nd segment, sender expects its ACK to arrive in time $IRTT_2$.
- Sender sets time out timer value (TOT) for the 2nd segment to be- **$TOT_2 = 2 \times IRTT_2$**
- Suppose ACK for the 2nd segment arrives in time $ARTT_2$.
- Here, $ARTT_2$ = Actual Round-Trip Time for the 2nd segment.
- In the similar manner, algorithm computes the time out timer value for all the further segments.

Advantages

- Time out timer value is flexible to dynamic round trip time.
- It takes into consideration all the previously sent segments to derive the initial RTT for the current segment.

Disadvantage

- It always considers Time out timer value = 2 x Initial round trip time.
- There is no logic behind using the number 2.

Jacobson's Algorithm

- Jacobson's Algorithm is a modified version of the basic algorithm.
- It gives better performance than Basic Algorithm.
- The steps involved in Jacobson's Algorithm are

- **Step-01: Sending 1st Segment-**
  - Sender assumes any random value of initial RTT say $IRTT_1$.
  - So, after sending the 1st segment, sender expects its ACK to arrive in time $IRTT_1$.
  - Sender assumes any random value of initial deviation say ID_1.
  - So, after sending the 1st segment, sender expects there will be a deviation of ID_1 time from $IRTT_1$.
  - Sender sets time out timer value (TOT) for the 1st segment to be-
    1. **$TOT_1 = 4 \times ID_1 + IRTT_1$**
  - Suppose ACK for the 1st segment arrives in time $ARTT_1$. Here, $ARTT_1$ = Actual Round-Trip Time for the 1st segment.
  - Then, Actual deviation from $IRTT_1$ is given by-
  - $AD_1 = | IRTT_1 - ARTT_1 |$

- **Step-02: Sending 2nd Segment-**
  - Sender computes the value of initial RTT for the 2nd segment using the relation-
    1. **$IRTT_{n+1} = \alpha\ IRTT_n + (1 - \alpha)\ ARTT_n$**
    2. Here, α is called smoothing factor where 0 <= α <= 1 (Its value will be given in questions)
  - Sender computes the value of initial deviation for the 2nd segment using the relation
    1. **$ID_{n+1} = \alpha\ ID_n + (1 - \alpha)\ AD_n$**
    2. Here, α is called smoothing factor where 0 <= α <= 1 (Its value will be given in questions)
  - Substituting n=1, sender gets
    1. $IRTT_2 = \alpha\ IRTT_1 + (1 - \alpha)\ ARTT_1$
    2. $ID_2 = \alpha\ ID_1 + (1 - \alpha)\ AD_1$
  - So after sending the 2nd segment, sender expects its ACK to arrive in time $IRTT_2$ with deviation of ID_2 time.
  - Sender sets time out timer value (TOT) for the 2nd segment to be-
    1. **$TOT_2 = 4 \times ID_2 + IRTT_2$**
  - Suppose ACK for the 2nd segment arrives in time $ARTT_2$. Here, $ARTT_2$ = Actual Round Trip Time for the 2nd segment.
  - Then, Actual deviation from $IRTT_2$ is given by-
    1. $AD_2 = | IRTT_2 - ARTT_2 |$
  - In the similar manner, algorithm computes the time out timer value for all the further segments.

Problems with Basic Algorithm and Jacobson's Algorithm

- To calculate initial round trip time, both the algorithms depend on the actual round-trip time of the previous segment through the relation
  1. **$IRTT_{n+1} = \alpha\ IRTT_n + (1 - \alpha)\ ARTT_n$**

- Consider ACK of some segment arrives to the sender after its initial time out timer goes off. Then, sender will have to re transmit the segment.
- Now for the segment being re transmitted, what should be the initial time out timer value is the concern.
- This is because the ACK is delayed and will arrive after time out. So, ARTT is not available.
- This problem is resolved by Karn's modification.

Karn's Modification

- Karn's modification states:
  - Whenever a segment has to be re transmitted, do not apply either of Basic or Jacobson's algorithm since actual RTT is not available.
  - Instead, double the time out timer (TOT) whenever the timer times out and make a retransmission.

Silly Window Syndrome

- Silly Window Syndrome is a problem that arises due to the poor implementation of TCP.
- It degrades the TCP performance and makes the data transmission extremely inefficient.
- The problem is called so because
    1. It causes the sender window size to shrink to a silly value.
    2. The window size shrinks to such an extent where the data being transmitted is smaller than TCP Header.

- **The problem arises due to following causes**
    1. Sender transmitting data in small segments repeatedly
    2. Receiver accepting only few bytes at a time repeatedly

- This problem is solved using Nagle's Algorithm.

Nagle's Algorithm

- Nagle's Algorithm tries to solve the problem caused by the sender delivering 1 data byte at a time. Nagle's algorithm suggests
    - Sender should send only the first byte on receiving one-byte data from the application.
    - Sender should buffer all the rest bytes until the outstanding byte gets acknowledged. In other words, sender should wait for 1 RTT.
    - After receiving the acknowledgement, sender should send the buffered data in one TCP segment.
    - Then, sender should buffer the data again until the previously sent data gets acknowledged.

Clark's Solution

- **Receiver Accepting Only Few Bytes Repeatedly**
  - Consider the receiver continues to be unable to process all the incoming data.
  - In such a case, its window size becomes smaller and smaller.
  - A stage arrives when it repeatedly sends the window size of 1 byte to the sender.
  - **This problem is solved using Clark's Solution.**

- Clark's Solution tries to solve the problem caused by the receiver sucking up one data byte at a time. **Clark's solution suggests-**
  - Receiver should not send a window update for 1 byte.
  - Receiver should wait until it has a decent amount of space available.
  - Receiver should then advertise that window size to the sender.
  - **Specifically, the receiver should not send a window update**
    - Until it can handle the MSS it advertised during Three Way Handshake
    - Or until its buffer is half empty, whichever is smaller.

- **Important Notes**
  - **Nagle's algorithm is turned off for the applications that require data to be sent immediately.** This is because Nagle's algorithm sends only one segment per round trip time. This impacts the latency by introducing a delay.
  - **Nagle's algorithm and Clark's solution are complementary.** Both Nagle's solution and Clark's solution can work together. The ultimate goal is sender should not send the small segments and receiver should not ask for them.

Q Consider the following statements about the timeout value used in TCP.
i. The timeout value is set to the RTT (Round Trip Time) measured during TCP connection establishment for the entire duration of the connection.
ii. Appropriate RTT estimation algorithm is used to set the timeout value of a TCP connection.
iii. Timeout value is set to twice the propagation delay from the sender to the receiver.
Which of the following choices hold? **(Gate-2007) (1 Marks)**
(A) (i) is false, but (ii) and (iii) are true
(B) (i) and (iii) are false, but (ii) is true
(C) (i) and (ii) are false, but (iii) is true
(D) (i), (ii) and (iii) are false
Answer: (B)

Q Identify the correct order in which a server process must invoke the function calls accept, bind, listen, and recv according to UNIX socket API. **(Gate-2015) (1 Marks)**

(A) listen, accept, bind recv
(B) bind, listen, accept, recv
(C) bind, accept, listen, recv
(D) accept, listen, bind, recv

Answer: (B)

Q Which one of the following socket API functions converts an unconnected active TCP socket into a passive socket? **(Gate-2014) (1 Marks)**

(a) CONNECT **(b)** BIND **(c)** LISTEN **(d)** ACCEPT

ANSWER C

Q A client process P needs to make a TCP connection to a server process S. Consider the following situation: the server process S executes a socket (), a bind () and a listen () system call in that order, following which it is pre-empted. Subsequently, the client process P executes a socket () system call followed by connect () system call to connect to the server process S. The server process has not executed any accept () system call. Which one of the following events could take place? **(Gate-2008) (2 Marks)**

(A) connect () system call returns successfully
(B) connect () system call blocks
(C) connect () system call returns an error
(D) connect () system call results in a core dump

Answer: (C)

Q Which of the following system calls results in the sending of SYN packets? **(Gate-2008) (1 Marks)**

(A) socket **(B)** bind **(C)** listen **(D)** connect

Answer (D)

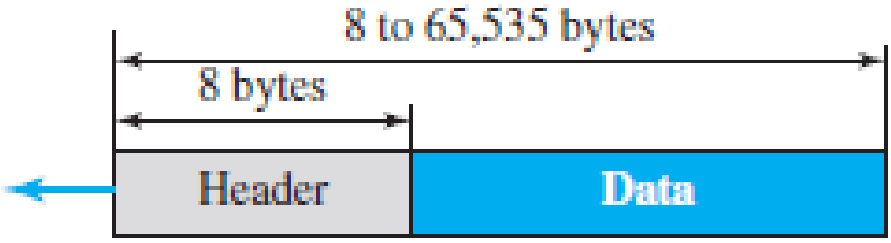
USER DATAGRAM PROTOCOL (UDP)

- The **User Datagram Protocol (UDP)** is a connectionless, unreliable transport protocol.
- It does not add anything to the services of IP except for providing process-to-process communication instead of host-to-host communication.

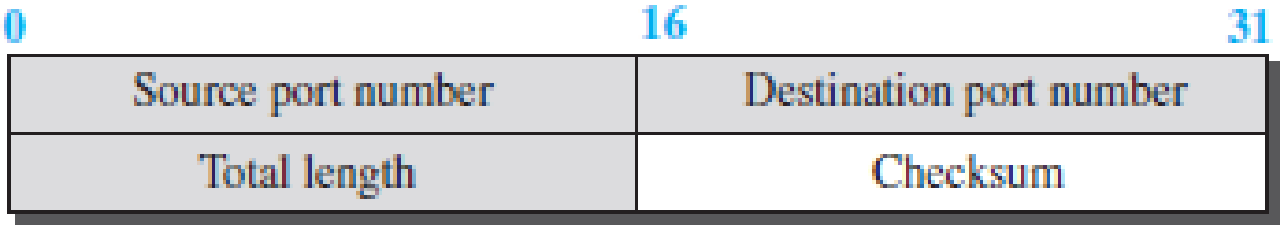
- **Why to use UDP**
  - UDP is a very simple protocol using a minimum of overhead.
  - If a process wants to send a small message and does not care much about reliability, it can use UDP.
  - Sending a small message using UDP takes much less interaction between the sender and receiver than using TCP.

User Datagram

- UDP packets, called ***user datagrams,*** have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits).



a. UDP user datagram



b. Header format

- 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.
- The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes.
- The last field can carry the optional checksum

Example: The following is the content of a UDP header in hexadecimal format.

CB84000D001C001C

Identify: Source port number, destination port number, total length of the user datagram, length of the data.

- The source port number is the first four hexadecimal digits $(CB84)_{16}$, that is the source port number is 52100. (as 1 digit of hexa defines 4 binary digits)
- The destination port number is the second four hexadecimal digits $(000D)_{16}$, that is the destination port number is 13.
- The third four hexadecimal digits $(001C)_{16}$, define the length of the whole UDP packet as 28 bytes.
- The length of the data is the length of the whole packet minus the length of the header, 28 − 8 = 20 bytes.

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***
  - Each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams.
  - The user datagrams are not numbered.
  - There is no connection establishment and no connection termination unlike TCP.
- Only those processes sending short messages, messages less than 65,507 bytes (65,535 minus 8 bytes for the UDP header and minus 20 bytes for the IP header), can use UDP.

Flow Control

- There is no *flow control,* and hence no window mechanism.

Error Control

- There is no *error control* mechanism in UDP except for the checksum.

Congestion Control

- It does not provide congestion control.

UDP Applications

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as ***FTP*** that needs to send bulk data.

- UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the ***Trivial File Transfer Protocol (TFTP)*** process includes flow and error control. It can easily use UDP.

- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.

- UDP is used for management processes such as SNMP

- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)

- UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message.

Q Match the following: **(GATE-2018) (1 Marks)**

Field	Length in bits
P. UDP Header's Port Number	I. 48
Q. Ethernet MAC Address	II. 8
R. IPv6 Next Header	III.32
S. TCP Header's Sequence Number	IV. 16

(a) P-III, Q-IV, R-II, S-I
(b) P-II, Q-I, R-IV, S-III
(c) P-IV, Q-I, R-II, S-III
(d) P-IV, Q-I, R-III, S-II
Ans: c

Q Consider socket API on a Linux machine that supports connected UDP sockets. A connected UDP socket is a UDP socket on which **connect** function has already been called. Which of the following statement is/are CORRECT? **(Gate-2017) (1 Marks)**
I. A connected UDP socket can be used to communicate with multiple peers simultaneously.
II. A process can successfully call **connect** function again for an already connected UDP socket
(a) I only **(b)** II only **(c)** Both I and II **(d)** Neither I nor II
Ans: b

Q Which of the following statements are TRUE? **(Gate-2008) (2 Marks)**
(S1) TCP handles both congestion and flow control
(S2) UDP handles congestion but not flow control
(S3) Fast retransmit deals with congestion but not flow control
(S4) Slow start mechanism deals with both congestion and flow control
(A) S1, S2 and S3 only
(B) S1 and S3 only
(C) S3 and S4 only
(D) S1, S3 and S4 only
Answer: (B)

Q A program on machine X attempts to open a UDP connection to port 5376 on a machine Y, and a TCP connection to port 8632 on machine Z. However, there are no applications listening at the corresponding ports on Y and Z. An ICMP Port Unreachable error will be generated by **(Gate-2006) (2 Marks)**
(A) Y but not Z
(B) Z but not Y
(C) Neither Y nor Z
(D) Both Y and Z
Answer: (D)

Q Packets of the same session may be routed through different paths in: **(Gate-2005) (1 Marks)**
(a) TCP, but not UDP
(b) TCP and UDP
(c) UDP, but not TCP
(d) Neither TCP nor UDP

Answer (b)

Q A firewall is to be configured to allow hosts in a private network to freely open TCP connections and send packets on open connections. However, it will only allow external hosts to send packets on existing open TCP connections or connections that are being opened (by internal hosts) but not allow them to open TCP connections to hosts in the private network. To achieve this the minimum capability of the firewall should be that of **(GATE-2007) (1 Marks)**

(A) A combinational circuit

(B) A finite automaton

(C) A pushdown automaton with one stack

(D) A pushdown automaton with two stacks

Answer: (D)

Q Which one of the following statements is FALSE? **(Gate-2004) (1 Marks)**

(A) TCP guarantees a minimum communication rate

(B) TCP ensures in-order delivery

(C) TCP reacts to congestion by reducing sender window size

(D) TCP employs retransmission to compensate for packet loss

Answer: (A)