Unit IV

4 Transport Layer

4.1 : Process to Process Delivery

Q.1 What are the duties of transport layer ? Explain how QoS is improved. [SPPU : May-15, Dec-15, Marks 6]

Ans. : Duties/ functions of transport layer :

1. This layer breaks messages into packets.
2. It performs error recovery if the lower layer are not adequately erro free.
3. Function of flow control if not done adequately at the network layer.
4. Functions of multiplexing and demultiplexing sessions together.
5. This layer can be responsible for setting up and releasing connection across the network.

- A transport layer protocol provides for logical communication betwee application processes running on different hosts. The logica communication means that the communicating application processes ar not physically connected to each other from the applications' viewpoint.

Quality of service

- The transport protocol entity should allow the transport service user t specify the quality of transmission service to be provided. Following ar the transport layer quality of service parameters

1. Error and loss levels
2. Desired average and maximum delay
3. Throughput
4. Priority level
5. Resilience.

- The error and loss level measures the number of lost or garble messages as a fraction of the total sent.

- The desired average and maximum delay measures the time between a message being sent by the transport user on the source machine and its being received by the transport user on the destination machine.
- The throughput parameter measures the number of bytes of user data transferred per second, measured over some time interval.
- The priority level parameter provides a way for a transport user to indicate that some of its connection are more important than other ones. The high priority connections get serviced before the low priority ones.

Q.2 Explain process to process delivery.

Ans. : • The transport layer is the fourth layer in the OSI layered architecture. The transport layer is responsible for reliable data delivery. The upper-layer protocols depends heavily on the transport layer protocol. A high level of error recovery is also provided in this layer. This layer ensures, that packets are delivered error free, in sequence and with no losses or duplications.

- Data link layer is responsible for delivery of frames between two neighboring nodes over a link. So this is called node-to-node delivery. Network layer is responsible for host-to-host delivery i.e. delivery of datagrams between two hosts.
- Transport layer is responsible for process to process delivery i.e. the delivery of a packet, part of a message from one process to another.
- Client server paradigm is used for process to process communication. A process on the local machine (hot) called a client needs services from a process usually on the remote host called server. Fig. Q.2.1 shows types of data deliveries.
- Nowadays, operating system support multiuser and multiprogramming environments. A remote computer can run several server programs at the same time.
- The services that a transport protocol can provide are often constrained by the service model of the underlying network layer protocol. If network layer protocol cannot provide delay or bandwidth guarantee for 4 PDU's sent between hosts, then the transport layer protocol cannot

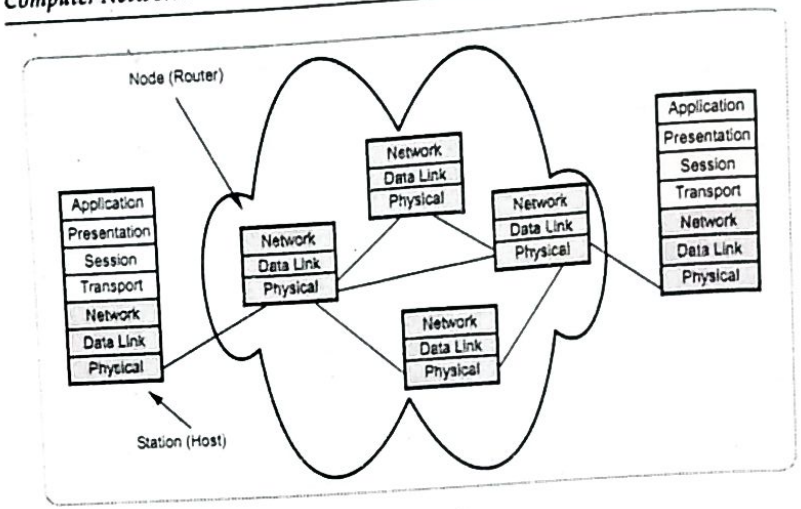


Fig. Q.2.1

provide delay or bandwidth guarantees for messages sent between processes.

- Some of the services offered by the transport protocol even when the underlying network protocol does not offer the corresponding service at the network layer.
- A transport protocol can offer reliable data transfer service to an application even when the underlying network protocol is unreliable, even when the network protocol loses, garbles and duplicate packets.

4.2 : Transport Service

Q.3 What are various transport service primitives ? Explain in brief.

☞ [SPPU : Dec.-13, Marks 4]

Ans. : Transport service primitives :

- To allow users to access the transport service, the transport layer must provide some operations to application programs. Real networks can lose packets, so the network service is generally unreliable. The transport service is reliable. Network service is used only by the transport entities. The transport service must be convenient and easy to use.

- Following are the primitives used for a simple transport service.

Primitive	Packet sent	Meaning
LISTEN	(None)	Block until some process tries to connect.
CONNECT	CONNECTION REQ	Actively attempt to establish a connection.
SEND	DATA	Send information.
RECEIVE	(None)	Block until a DATA packet arrives.
DISCONNECT	DISCONNECTION REQ	This side wants to release the connection.

- At the transport layer, even a simple unidirectional data exchange is more complicated than at the network layer. Every data packet sent will also be acknowledged. These acknowledgement are managed by the transport entities using the network layer protocol and are not visible to transport user.
- Transport Protocol Data Unit (TPDU) for messages sent from transport entity to transport entity. TPDU are contained in the packets. Packets are contained in frames. When a frame arrives, the data link layer processes the frame header and passes the contents of the frame payload field up to the network entity. The network entity processes the packet header and passes the contents of the packet payload upto the transport entity.
- Fig. Q.3.1 shows the nesting of TPDU with packets.
- The client's CONNECT call causes a CONNECTION REQUEST TPDU to be sent to the server. When it arrives, the transport entity checks to see that the server is blocked on a LISTEN. It then unblocks the server and sends a CONNECTION ACCEPTED TPDU back to the client.

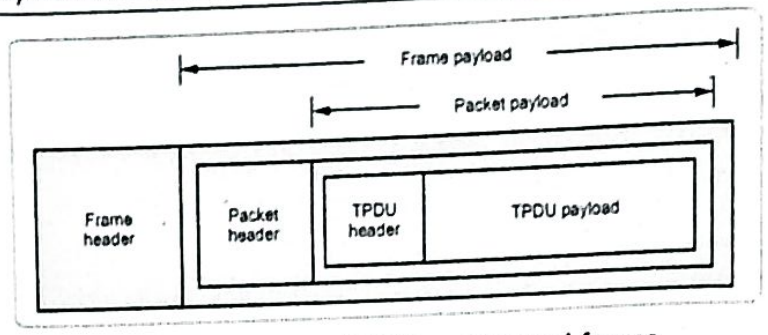


Fig. Q.3.1 Nesting of TPDU, packets and frames

When this TPDU arrives, the client is unblocked and the connection is established.

- Data can be exchanged using the SEND and RECEIVE primitives. When the TPDU arrives, the receiver is unblocked. It can then process the TPDU and send a reply.
- When a connection is no longer needed, it must be released to free up table space within the two transport entities. Disconnection has two types : Asymmetric and symmetric.

Q.4 Explain transport layer design issues.

Ans. : The transport layer delivers the message from one process to another process running on two different hosts. Thus, it has to perform number of functions to ensure the accurate delivery of message. The various functions of transport layer are :

1. Establishing, maintaining and releasing connection
   - The transport layer establishes, maintains and releases end-to-end transport connection on the request of upper layers.
   - Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc.
   - A connection is released at the request of upper layer.
2. Addressing
   - In order to deliver the message from one process to another, an addressing scheme is required. Several processes may be running on a system at a time.

   - In order to identify the correct process out of the various running processes, transport layer uses an addressing scheme called port number. Each process has a specific port number.
3. Data transfer
   - Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU.
   - The TPDU is handed over to the network layer for its delivery to destination. The TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.
4. Flow control
   - Like data link layer, transport layer also performs flow control. However, flow control at transport layer is performed end-to-end rather than node-to-node.
   - Transport layer uses a sliding window protocol to perform flow control.
5. Error Control : Transport layer also provides end-to-end error control facility. It also deals with several different types of errors like damaged bits, non delivery and duplicate delivery of TPDUs.
6. Congestion Control : Transport layer also handles congestion in the networks. Several different congestion control algorithms are used to avoid congestion.

4.3 : Socket Programming

Q.5 What are the types of socket ? Explain various socket primitives used in connection oriented client server approach.

☞ [SPPU : Dec-17,19, May-18, End Sem, Marks 6]

OR What are three different types of sockets ? Explain various socket primitives used in connection oriented client server approach.

☞ [SPPU : May-19, End Sem, Marks 8]

Ans. : • Sockets allow communication between two different processes on the same or different machines. The socket is the combination of IP address and software port number used for communication between multiple processes.

- Types of socket are as follows :
  1. **Stream socket :** It uses TCP protocol. The stream socket (SOCK_STREAM) interface defines a reliable connection-oriented service. Data is sent without errors or duplication and is received in the same order as it is sent.
  2. **Datagram socket :** It uses UDP protocol. The datagram socket (SOCK_DGRAM) interface defines a connectionless service for datagrams or messages. Datagrams are sent as independent packets. The reliability is not guaranteed, data can be lost or duplicated and datagrams can arrive out of order.
  3. **Raw socket :** It uses IP and ICMP protocol. The raw socket (SOCK_RAW) interface allows direct access to lower-layer protocols such as Internet Protocol (IP).

- Socket primitive are as follows :
  - Before an application program can transfer any data, it must first create an end point for communication by calling socket. Socket facilities are provided in C language.
  - To use these facilities, the header files <types.h> and <socket.h> must be included in the program. Its prototype is

    int socket (int family, int type, int protocol) ;

    where the family identifies the family by address or protocol. The address family identifies a collection of protocol with the same address format, while the protocol family identifies a collection of protocols having the same architecture. The type identifies the semantics of communication.
  - After a socket is created, the bind system call can be used to assign an address to the socket. Its prototype is

    int bind (int sd, struct sockaddr *name, int namelen) ;

    where sd is the socket descriptor returned by the socket call, name is a pointer to an address structure that contains the local IP address and port number and nameless is the size of the address structure in bytes. The bind system call returns 0 on success and – 1 on failure.

  - A connection-oriented server indicates its response to receive connection request by calling listen the prototype is

    int listen (int sd, int backlog) ;

    where sd is socket descriptor returned by the socket call and backlog specifies the maximum number of connection requests that the system should queue while it waits for the server to accepts them.
  - Server can accept the connection request after listen call. The prototype for accept is :

    int accept (int sd, struct sockaddr *addr, int *addrlen) ;
  - After this client and server transmit data using write and read system calls. If the socket is not used for long time, socket is closed by using close system call. The prototype used for close call is :

    int close (int sd) ;

    If socket is closed successfully, it returns 0 otherwise – 1 for failure.

4.4 : Elements of Transport Layer Protocols

Q.6 Explain upward and downward multiplexing in transport layer.

Ans. : Multiplexing : • Many virtual circuits open for long periods of time is to make multiplexing of different transport connections onto the same network connection attractive. This form of multiplexing called **upward multiplexing.**

- Four distinct transport connections all use the same network connection to the remote host. The transport layer forms the group of transport connection according to their destination and map each group onto the minimum number of network connections.

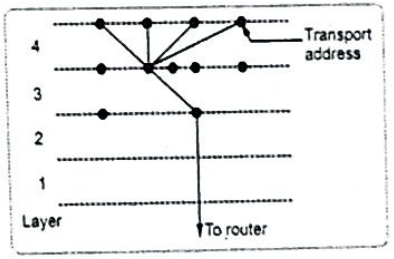


Fig. Q.6.1 (a) Upward multiplexing

- Many transport connections are mapped onto one network connection, the performance will be poor, because users will have to wait their turn

to send one message. If too few transport connections are mapped onto one network connection, the service will be expensive.

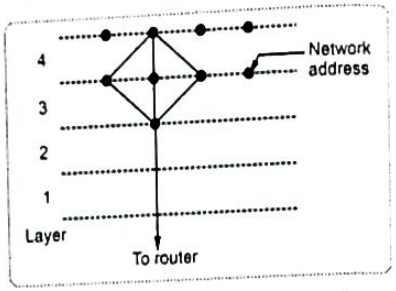


Fig. Q.6.1 (b) Downward multiplexing

- The transport layer opens multiple network connections and distributes the traffic among them on a round-robin basis. This is called **downward multiplexing.**
- With k network connections open, the effective bandwidth is increased by a factor of k. If multiple output lines are available, downward multiplexing can also be used to increase the performance even more.
- Fig. Q.6.1 shows the multiplexing.

Q.7 Explain transport layer crash recovery.

Ans. : Crash Recovery :

- If the host computer (server) and routers are subject to crashes, the recovery from these crashes makes some problem. If the network layer provides connection-oriented service, then the lost of data or virtual circuit is handled by establishing a new connection. Then retransmit the TPDUs, which has not received. If the host server crashes, then the client must quickly reboot.
- When the server crash while receiving data from client, the outstanding TPDU is lost. To recover the data, when the server comes back up, its tables are reinitialized, so it no longer knows precisely where it was.
- The server might send a broadcast TPDU to all other host, announcing that it had just crashed and requesting that its clients inform it for the status of all open connections. Client can be in one of two states : TPDU outstanding or no TPDU outstanding. Based on only this state information the client must decide whether or not to retransmit the most recent TPDU.

- To avoid the duplicate of data the client should retransmit only if it has an unacknowledged TPDU outstanding.
- There are many situation for crash recovery. If the server send acknowledge and crash the server before writing the data. The writing (saving data) data and sending acknowledgement, both are different process. So the server and client programmed in any way, the lost of data and duplicate of data may occurs.
- If both sides are programmed considering all situation, upto some limit it is possible to recover the lost data and possible to avoid retransmitting.

Q.8 Explain connection establishment and connection release with respect to the transport layer.

Ans. : i) Connection establishing : The connection establishment serves three main purposes.

1. It allows each end to assure that the other exists.
2. It allows negotiation of optional parameter like maximum segment size, maximum window size and quality of service.
3. It triggers allocation of transport entity resources like buffer space.

- There are three phases in any virtual connection. These are the connection establishment, data transfer and connection termination phases. In order for two hosts to communicate using TCP they must first establish a connection by exchanging messages in what is known as the three-way handshake.
- Fig. Q.8.1 shows the process of the three-way handshake.
- There are three TCP segments exchanged between two hosts, host A and host B. To start, host A initiates the connection by sending a TCP segment with the SYN control bit set and an Initial Sequence Number (ISN) here it is represent as the variable x in the sequence number field.
- At some moment later in time, host B receives this SYN segment, process it and responds with a TCP segment of its own. The response from host B contains the SYN control bit set and its own ISN represented as a variable y.
- Host B also sets the ACK control bit to indicate the next expected byte from host A should contain data starting with sequence number x+1.

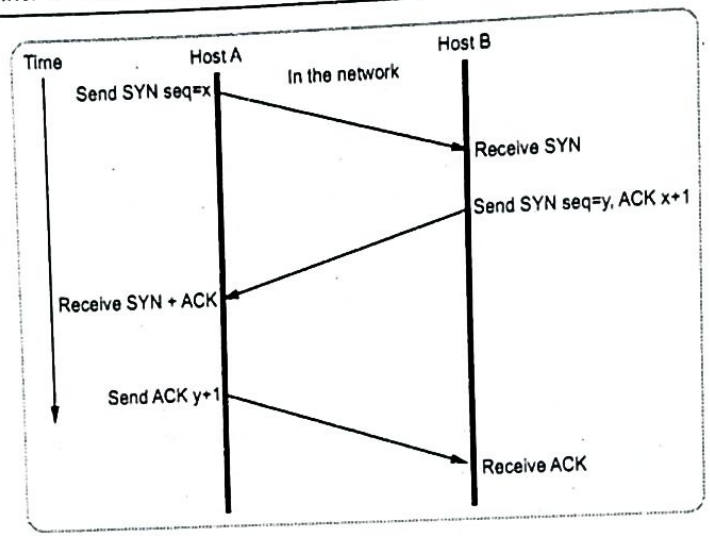


Fig. Q.8.1 TCP connection establishment

- When host A receives host B's ISN and ACK, it finishes the connection establishment phase by sending a final acknowledgement segment to host B. In this case, host A sets the ACK control bit and indicates the next expected byte from host B by placing acknowledgement number y+1 in the acknowledgement field.
- Once ISNs have been exchanged, communicating applications can transmit data between each other.

ii) **Connection Release :** In order for a connection to be released, four segments are required to completely close a connection. Four segments are necessary due to the fact that TCP is a full-duplex protocol, meaning that each end must shut down independently. The connection termination phase is shown in Fig. Q.8.2.

- Notice that instead of SYN control bit fields, the connection termination phase uses the FIN control bit fields to signal the close of a connection.
- To terminate the connection, the application running on host A signals TCP to close the connection. This generates the first FIN segment from host A to host B.
- When host B receives the initial FIN segment, it immediately acknowledges the segment and notifies its destination application of the

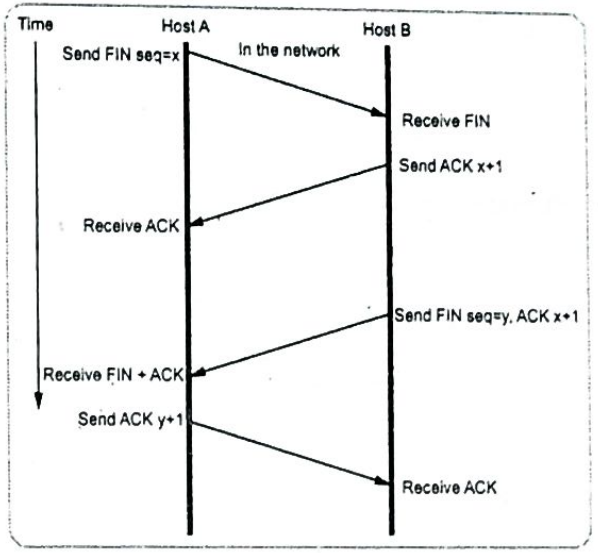


Fig. Q.8.2 Connection termination

termination request. Once the application on host B also decides to shut down the connection, it then sends its own FIN segment, which host A will process and respond with an acknowledgement.

Q.9 Write short note on flow control and buffering.

Ans. : • Flow control is implemented using modified form of sliding window protocol. The window size is variable and is controlled by the receiver. The receiver sends a credit allocation to the sender. The credit allocation indicates how many TDPU the receiver is ready to receive if the network service is unreliable, the sender must buffer all TDPUs sent. With reliable network service, if the sender knows that the receiver always has buffer space, it need not retain copies of the TPDU it sends.

- If the receiver cannot guarantee that every incoming TPDU will be accepted, the sender will have to buffer anyway. Even if the receiver has agreed to do the buffering, then the problem comes with buffer size.
- If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identical size buffers, with one TPDU per buffer. If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives. If the buffer size is less than

the maximum TPDU size, multiple buffers will be needed for long TPDUs, with the attendent complexity.

4.5 : Congestion Control

Q.10 What is congestion and congestion control ?

Ans. : Congestion :

When too many packets rushing to a node or a part of network, the network performance degrades, and this situation is called as **congestion**. When the number of packets dumped into the subnet and as traffic increases the network is no longer able to cope and design losing packets at very high traffic, performance collapses completely and almost no packets are delivered.

Congestion control :

Congestion control is a process of maintaining the number of packets in a network below a certain level at which performance falls off. Congestion control makes sure that subnet is able to carry the offered traffic. So congestion control is different process than flow control.

Q.11 Explain Additive Increase, Multiplicative Decrease control (AIMD) TCP congestion control method.

Ans. : • TCP maintains a new state variable for each connection, called congestion window, which is used by the source to limit how much data it is allowed to have in transit at a given time. The congestion window represents the amount of data, in bytes.

- AIMD performs a slow increase in the congestion window size when the congestion in the network decreases and a fast drop in the window size when congestion increases.
- Let W_m be the maximum window size, in bytes, representing the maximum amount of unacknowledged data that a sender is allowed to send.
- Let W_a be the advertised window sent by the receiver, based on its buffer size.
- TCP's effective window is revised as follows :

  Max window = MIN (Congestion window, Advertised window)

Effective window = Max window – (Last byte sent – Last byte ACKed)

- **Max window** replaces **Advertised window** in the calculation of **Effective window.**
- Fig. Q.11.1 shows the additive increase control for TCP congestion control.

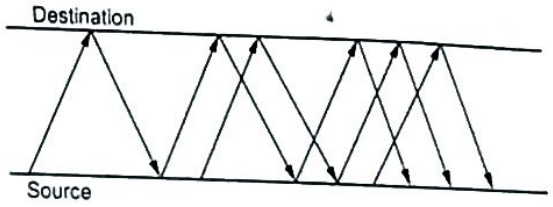


Fig. Q.11.1 AIDM

- The challenge in TCP congestion control is for the source node to find a right value for the congestion window. The congestion window size varies, based on the traffic conditions in the network. TCP watches for timeout as a sign of congestion.
- TCP technique requires that the timeout values be set properly. Two important factors in setting timeouts follow.
  1. Average round trip times (RTTs) and RTT standard deviation based to set timeouts.
  2. RTTs are sampled once every RTT is completed.

Q.12 Explain slow start method.

Ans. : • Slow start method increases the congestion window size nonlinearly and in most cases exponentially, as compared to the linear increase in additive increase.

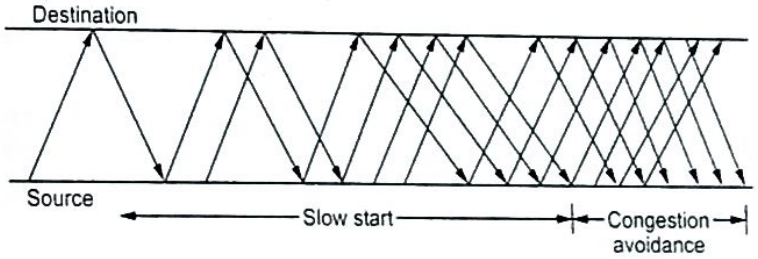


Fig. Q.12.1 Slow start

- Fig. Q.12.2 shows the slow start method. In this method, the congestion window is again interpreted in packets instead of bytes.
- Source initially sets the congestion window to one packet. When its corresponding acknowledgement arrives, the source sets the congestion window to two packets. Now, the source sends two packets. On receiving the two corresponding acknowledgements, TCP sets the congestion window size to 4. Thus, the number of packets in transit doubles for each round-trip time.
- The slow start method is normally used,
  1. Just after a TCP connection is set up.
  2. When a source is blocked, waiting for a timeout.

4.6 : Transport Layer Protocols : TCP

Q.13 Explain TCP header in detail.

☞ [SPPU : May-18,19, End Sem, Marks 6]

Ans. : • Fig. Q.13.1 shows the format of the TCP header.

- Description of field in the TCP header as follows :
  1. **Source port :** It specifies the application sending the segment. This is different from the IP address, which specifies an interne address.
  2. **Destination port :** It identifies the receiving application por numbers below 256 called well-known ports and are assigned t commonly used applications. For examples, port 23 corresponds t a Telnet function. Port 53 for DNS name server and port 2 assigned for FTP.
  3. **Sequence number :** Each byte in the stream that TCP sends i numbered. The sequence number wraps back to 0 after $2^{32} - 1$.
  4. **Acknowledgement number :** This field identifies the sequenc number of the next data by the that the sender expects to receive i the ACK bit is set. If the ACK bit is not set, this field has n effect.

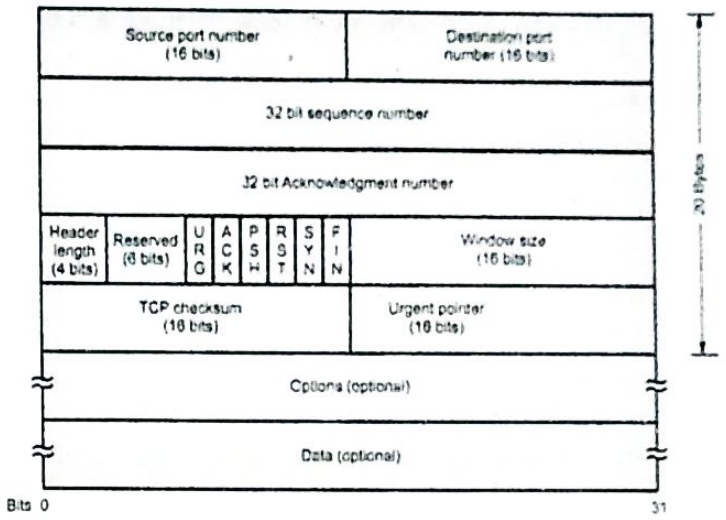


Fig. Q.13.1 TCP header format

5. **Header length :** It specifies the length of the TCP header in 32-bit words. Because of option field, header length is used.
6. **Reserved :** This field is reserved for future use and must be set to 0 (zero).
7. TCP header contains six flag bits. One or more than one can be turned on at the same time. The function of each flag is as follows.
   a. **URG :** The Urgent pointer is valid if it set to 1.
   b. **ACK :** ACK bit is set to 1 to indicate that the acknowledgement number is valid.
   c. **PSH :** The receiver should pass this data to the application as soon as possible.
   d. **RST :** This flag is used to reset the connection. It is also used to reject an invalid segment.
   e. **SYN :** Synchronize sequence number to initiate a connection. The connection request has SYN = 1 and ACK = 0 to indicate that the piggyback acknowledgement field is not in use.

- A typical value for MSS is 1460.
- TCP connections are full duplex. The steps required establishing and release connections can be represented in a finite state machine.

Q.15 Explain four steps of connection termination in TCP.

Ans. : • Any of the two parties involved in exchanging data can close the connection. When connection in one direction is terminated, the other party can continue sending data in the other direction.

- Four steps are required to close the connection in both direction. Fig. Q.15.1 shows four step connection termination.

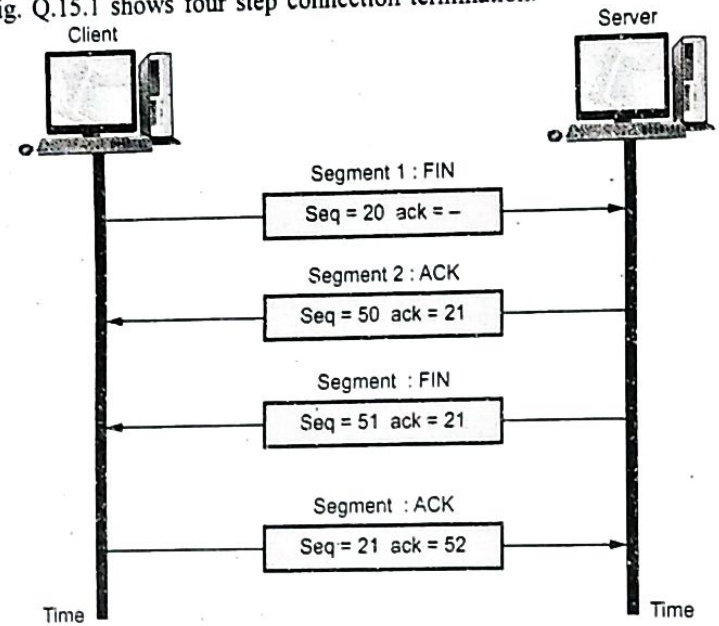


Fig. Q.15.1 Four steps connection termination

- **Steps are as follows :**

1. The client TCP sends the first segment, a FIN segment.
2. The server TCP sends the second segment, an ACK segment, to confirm the receipt of the FIN segment from the client.
3. The server TCP can continue sending data in the server client direction. When it does not have any more data to send, it sends the third segment.

4. The client TCP sends the fourth segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.

Q.16 What causes silly window syndrome ? How it is solved ? Explain with an example. ☞ [SPPU : April-17 (In Sem), Marks 5]

Ans. : Silly window syndrome :

- When large block of data is passed from sender but the receiver reads data one byte at a time. Receiving side, the TCP buffer is full and the sender know the condition. The interactive application reads one character from the TCP stream.
- Receiving TCP tells to the sender to send the only 1 byte. Sender send 1 byte. Now buffer is full and receiver send acknowledgement the 1-byte segment and set the window 0. This operation is continuous. Fig. Q.16.1 shows these steps.

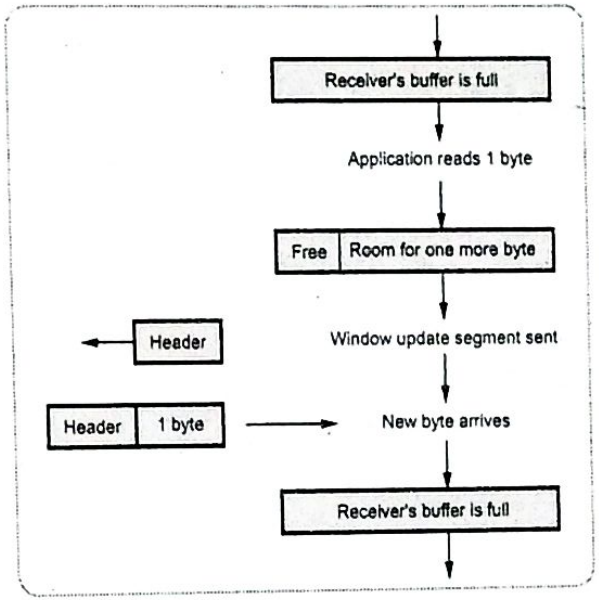


Fig. Q.16.1 Silly window syndrome

- Nagle algorithm and Clark's solution to the silly window syndrome are complementary. Clark solution is to prevent the receiver from sending a

window update for 1 byte. Instead it is forced to wait until it has a decent amount of space available.

Q.17 Differentiate between TCP and UDP.

☞ [SPPU : Dec.-18, 19, May-19, End Sem, Marks 6]

Ans. :

Sr. No.	TCP	UDP
1.	TCP is connection oriented.	UDP is connectionless.
2.	TCP connection is byte stream.	UDP connection is message stream.
3.	TCP does not support multicasting and broadcasting.	UDP support broadcasting.
4.	It provides error control and flow control.	It does not provide flow control and error control.
5.	TCP supports full duplex transmission.	UDP does not support full duplex transmission.
6.	TCP is reliable.	UDP is unreliable.
7.	TCP packet is called segment.	UDP packet is called user datagram.

Q.18 Explain TCP timers.

Ans. : • TCP manages four different timers for each connection.

a) A **retransmission timer** is used when excepting an acknowledgement from the other end.

b) A **persist timer** keeps window size information flowing even if the other end closes its receiver window.

c) A **keep alive timer** detects when the other end on an otherwise idle connection crashes.

d) A 2 **maximum segment lifetime** (2 MSL) timer measures the time a connection has been in the TIME_WAIT state.

Q.19 For each of the following applications, determine whether TCP or UDP is used as the transport layer protocol and explain the reasons(s) for your choice

i) Watching a real time streamed video.

ii) Web browsing.

iii) A Voice over IP (VoIP) telephone conversation.

iv) YouTube video.

☞ [SPPU : Dec.-17, End Sem, Marks 8]

Ans. : i) Watching a real time streamed video : This should be UDP. The reason is that when watching a movie, delay is critical and therefore there simply is not any time to seek the retransmission of any errors. The simplicity of UDP is therefore required.

ii) Web browsing : This should be TCP. The reason is that web pages need to be delivered without error so that all content is properly formatted and presented. Therefore the error detection and correction properties of TCP are needed.

iii) A Voice over IP (VoiP) telephone conversation : This should be UDP. The reason is that a telephone conversation has strict timing requirements for the transfer of data and seeking the retransmission of any errors would introduce too much delay. Therefore the simplicity of UDP is needed.

iv) YouTube video : This should be TCP. Video streaming adopts pre-fetching and buffering to achieve smooth play-out. TCP provides such (network} buffer, as well as the reliable transmission guarantee for no loss of frame.

Q.20 Explain state transition diagram of TCP.

☞ [SPPU : Dec-18, 19, End Sem, Marks 6]

Ans. : TCP Finite State Machine :

• Fig. Q.20.1 shows finite state machine.

• The lightface lines are unusual event sequences. Each line in Fig. Q.20.1 is marked by an event/action pair. The event can either be a user-initiated system call (CONNECT LISTEN, SEND or CLOSE), a segment arrival (SYN, FIN, ACK or RST) or in one case, a timeout of twice the maximum packet lifetime. The action is the sending of a control segment (SYN, FIN, or RST) or nothing, indicated by ---. Comments are shown in parentheses.

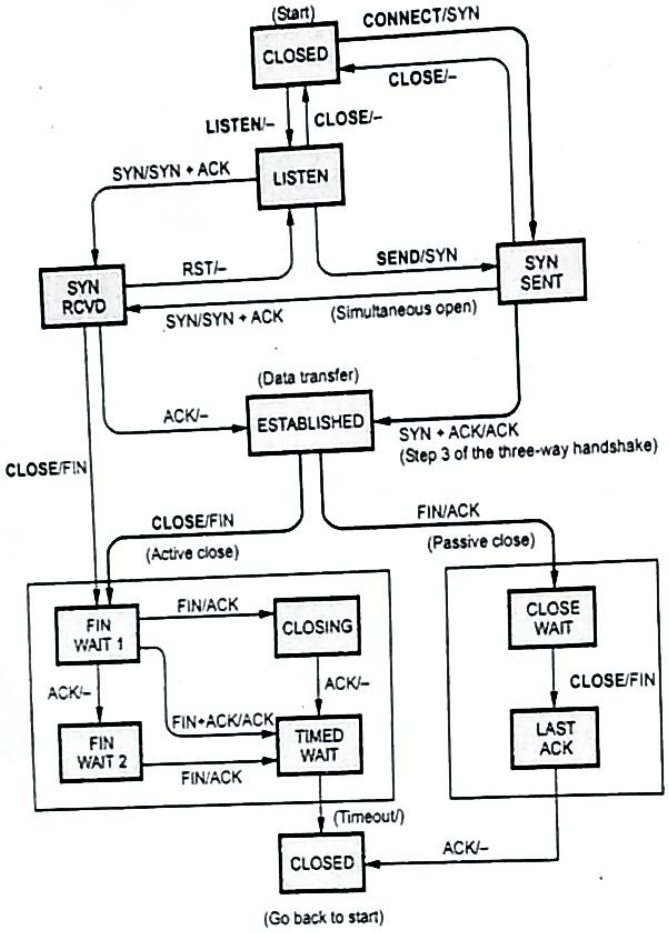


Fig. Q.20.1 TCP finite state machine

- The states used in the TCP connection management finite state machine are as follows :

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call

SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

- The diagram can best be understood by first following the path of a client (the heavy solid line) then later the path of a server (the heavy dashed line). When an application on the client machine issues a CONNECT request, the local TCP entity creates a connection record, marks it as being in the SYN SENT state, and sends a SYN segment.
- Note that many connections may be open (or being opened) at the same time on behalf of multiple applications, so the state is per connection and recorded in the connection record. When the SYN + ACK arrives. TCP sends the final ACK of the three-way handshake and switches into the ESTABLISHED state data can now be sent and received.
- When an application is finished, it executes a CLOSE primitive, which causes the local TCP entity to send a FIN segment and wait for the corresponding ACK (dashed box marked active close).
- When the ACK arrives, a transition is made to state FIN WAIT 2 and one direction of the connection is now closed. When the other side closes, too, a FIN comes in, which is acknowledged. Now both sides

are closed, but TCP waits a time equal to the maximum packet lifetime to guarantee that all packets from the connection have died off, just in case the acknowledgement was lost. When the timer goes off, TCP deletes the connection record.

- Connection management from server view point, sever does a LISTEN and settles down to see who turns up. When a SYN comes in, it is acknowledged and the server goes to the SYN ACK state. When the server's SYN is itself acknowledged, the three way handshake is complete and the server goes to the ESTABLISHED state. Data transfer can now occur.

4.7 : Transport Layer Protocols : UDP, SCTP, RTP

Q.21 Explain UDP protocol header format.

Ans. : • Fig. Q.21.1 shows the format of the UDP header. The port number identify the sending process and the receiving process.

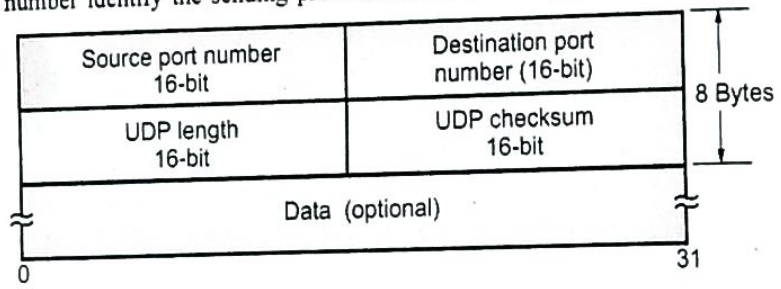


Fig. Q.21.1 UDP header

- The UDP datagram contains a source port number and destination port number. Source port number identifies the port of the sending application process. The destination port number identifies the receiving process on the destination host machine.
- The UDP length field is the length of the UDP header and the UDP data in bytes. The minimum value for this field is 8 bytes.
- UDP checksum covers the UDP header and the UDP data. Both UDP and TCP include a 12 byte pseudo-header with the UDP datagram just for the checksum computation. This pseudo_header includes certain

fields from the IP header. The purpose is to let UDP double check that the data has arrived at the correct destination.

- UDP checksum is end-to-end checksum. It is calculated by the sender, and then verified by receiver. It is designed to catch any modification of the UDP header or data anywhere between sender and receiver.

Q.22 Explain RTP protocol in detail.

[SPPU : May-19, Dec-19, End Sem, Marks 8]

Ans. : • Fig. Q.22.1 shows the RTP header. RTP header size is 32 bits. Fields in the headers are version, P, X, CC, M, payload type, sequence number, timestamp, synchronization source identifier and contributing source identifier.

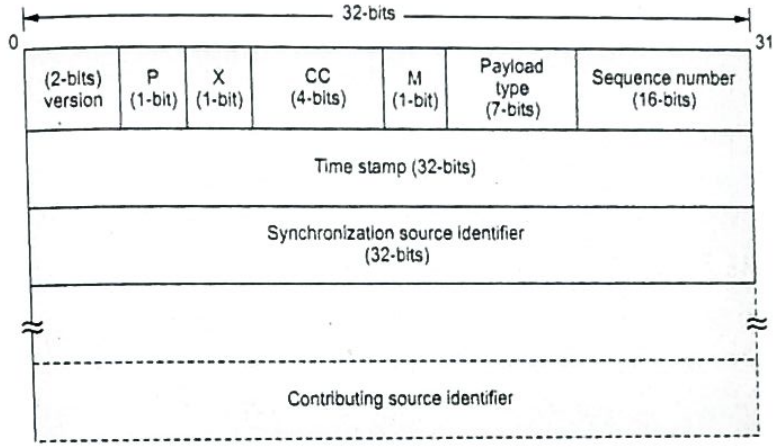


Fig. Q.22.1 RTP header

1. **Version :** Size of version field is 2-bits. It indicates version number. The current version is 2.
2. **P bit :** Size is 1-bit. P bit indicates that the packet has been **padded** to a multiple of 4 bytes.
3. **X-bit :** Size is again 1-bit and it indicates that the **extension** header is present.
4. **CC field :** Size of CC field is 4-bits. CC field is used for indicating number of source present. The range is from 0 to 15.

5. **m bit :** Marker bit is of 1-bit size. This bit is used to indicate start of the frame. It may be video frame, start of a word in an audio channel.
6. **Payload type :** Size of the payload type field is 7-bits. This field is used for indicating encoding algorithm has been used. It determines its interpretation by the application.
7. **Sequence number :** This 16-bit field is incremented by one each time an RTP packet is sent. The number can be used by the receiver to detect packet loss and to recover packet sequence. The initial value is selected at random.
8. **Time stamp :** It is 32-bits number specifies the sampling instant of the first byte in the RTP data packet. This value can help to reduce jitter at the receiver by decoupling the playback from the packet arrival time. The initial value is selected at random.
9. **Synchronization source identifier :** This field tells which stream the packet belongs to. It is the method used to multiplex and demultiplex multiple data streams onto a single stream of UDP packets.
10. **Contributing source identifier :** This list of 0 to 15 thirty-two bit items specifies the contributing sources for the payload contained in the packet. This field is used when mixers are present in the studio.

Q.23 Explain UDP header ? Below is an hexadecimal dump of an UDP datagram captured.

e2 a7 00 0D 00 20 74 9e 0e ff 00 00 00 01 00 00 00 00 00 00 06 69 73 61 74 61 70 00 00 01 00 01

i) What is source port number ?

ii) What is destination port number ?

iii) What is total length of the user datagram ?

iv) What is the length of the data ?

v) Is packet directed from a client to server or vice versa ?

☞ [SPPU : Dec.-17, End Sem, Marks 8]

Ans. : UDP header : Refer Q.21.

i) The source port number is the first four hexadecimal digits {e2a7), which means thc1t the source port number is 58023.

ii) The destination port number is the second four hexadecimal digits (OOOD), which means that the destination port number is 13.

iii) The third four hexadecimal digits (0020) define the length of the whole UDP packet as 32 bytes.

iv) The length of the data is the length of the whole packet minus the length of the header, or 32-8 = 24 bytes.

v) Since the destination port number Is 13 {well known port), the packet is from the client to the server.

Q.24 Below is an hexadecimal dump of an UDP datagram captured.

06 32 00 0D 00 1C E2 17

i) What is source port number ?

ii) What is destination port number ?

iii) What is the length of the data ?

iv) Is packet directed from a client to server or vice versa ?

☞ [SPPU : May-18, Dec-18, End Sem, Marks 4]

Ans. : i) Source port number = $(0632)^{16}$ = 1586

ii) Destination port number = $(000D)^{16}$ = 13

iii) Total length = $(001C)^{16}$= 28 bytes

iv) Since the header is 8 bytes the data length is 28 – 8 = 20 bytes.

v) From a client to a server

vi) Daytime

Q.25 Host A sends a UDP datagram containing 8880 bytes of user data to host B over an Ethernet LAN. Ethernet frames may carry data up to 1500 bytes (i.e. MTU = 1500 bytes). Size of UDP header is 8 bytes and size of IP header is 20 bytes. There is no option field in IP header. How may total number of IP fragments will be transmitted and what will be the contents of offset field in the last fragment ?

☞ [SPPU : Dec.-18, End Sem, Marks 6]

Ans. :

User data = 8880 bytes

MTU = 1500 bytes

8880 headers = 8 bytes

IP headers = 20 bytes

Header length = 20 + 8 = 28 bytes

Actual data in each fragments = 1500 – 28 = 1472 bytes

Fragments	Data	Headers	Offsets
1st	1472	28	0/8 = 0
2nd	1472	28	1472/8 = 184
3rd	1472	28	1472 × 2 / 8 = 368
4th	1472	28	1472 × 3 /8 = 552
5th	1472	28	1472 × 4 / 8 = 736
6th	1472	28	1472 × 5 / 8 = 920
7th	48	28	1472 × 6 / 8 = 1104

Number of IP fragments will be 7

Offset field of last fragments is 1104

So the correct options is "C"

Q.26 Write short note on SCTP.

Ans. : • SCTP is a reliable transport protocol operating on top of a potentially unreliable connectionless packet service such as IP. It offers acknowledged error-free non-duplicated transfer of datagrams (messages). Detection of data corruption, loss of data and duplication of data is

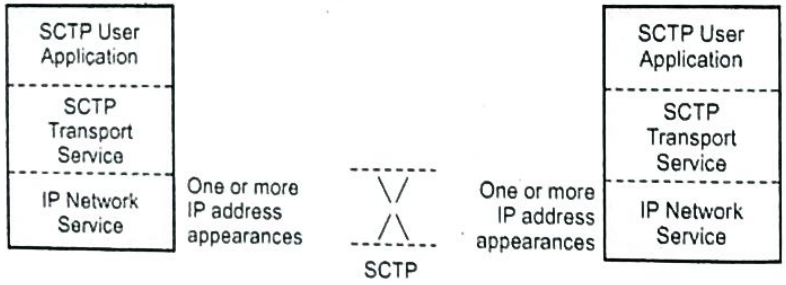


Fig. Q.26.1 Diagram showing the concept of SCTP association

achieved by using checksums and sequence numbers. A selective retransmission mechanism is applied to correct loss or corruption of data.

- **Stream Control Transmission Protocol** (SCTP) is a Ttransport Layer Protocol, serving in a similar role as the popular protocols Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). It provides some of the same service features of both, ensuring reliable, in-sequence transport of messages with congestion control.
- An SCTP association generally looks like this, so the services of SCTP are naturally at the same layer as TCP or UDP services :

Q.27 Explain SCTP services.

Ans. : • Similar to TCP, SCTP provides a reliable and in-order data transfer service to HTTP. Additionally, SCTP provides other services unavailable in TCP. These services are summarized below.

1. Multistreaming.
2. Process to process communication.
3. Four-way handshake during association establishment.
4. No Maximum Segment Lifetime (MSL) during association termination.
5. Multihoming for improved fault tolerance.
6. Preserving application message boundaries.
7. Reliable services.
8. Connection oriented service.
9. Sequenced delivery of user datagrams within a stream.

4.8 : Congestion Control and Quality of Service (QoS), Differentiated Services

Q.28 Write short note on RSVP.

Ans. : • RSVP is a signalling protocol used to reserve resources in the Internet. RSVP is a bandwidth reservation protocol.

- RSVP protocol allows applications to reserve bandwidth for their data flows.

Characteristics of RSVP

1. It provides reservations for bandwidth in multicast trees.
2. RSVP is receiver-oriented i.e. receiver intitiates this protocol for resource reservation.

- To get better reception and eliminate congestion any of the receivers in a group can send a reservation message up the tree to the sender. The message is propagated using the reverse path forwarding algorithm. Example of such a reservation is shown in the Fig. Q.28.1.

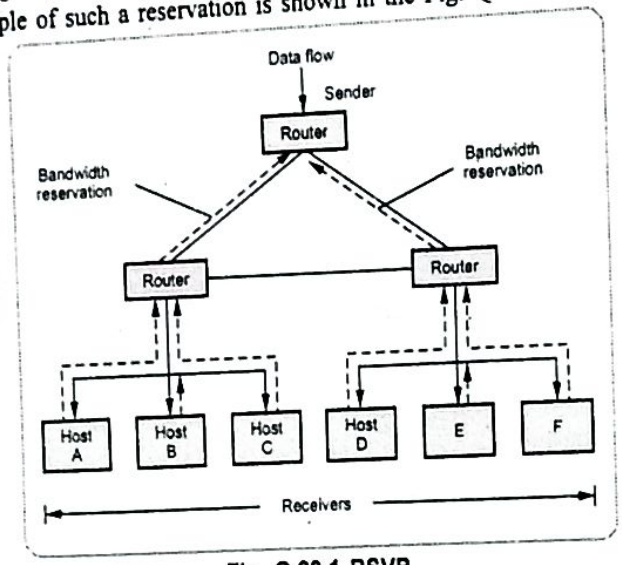


Fig. Q.28.1 RSVP

- Fig. Q.28.1 shows multicast spanning tree with data flowing from the top of the tree to hosts at the bottom of tree. The data originates from the sender and reservation message is originates from the receiver. The upstream reservation messages from host can be merged with other reservation message.

Q.29 List and explain fundamental elements of differentiated services.

Ans. : • The Differentiated Services (DiffServes) architecture consists of two sets of functioal elements :

1. Edge functions 2. Core functions

1. Edge functions :

- The packets arriving at the edge of network are marked. The mark of the packet defines the class of traffic to which it belongs. Depending on the mark, the packet may be immediately forwarded into the network, delayed or discarded.
- Edge functions are also called packet classification and traffic conditioning.

2. Core functions :

- On forwarding the packet by router it is then put on for next hop according to per hop behavior. The per hop behavior influences how router's buffer and BW are shared. It is a forwarding function of differentiated services (diffserv).
- Fig. Q.29.1 shows a logical view of classificaton and marking function within the edge7 router.

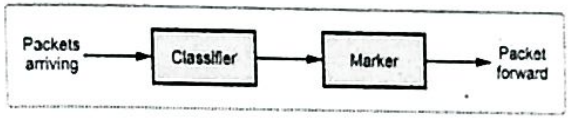


Fig. Q.29.1 Packet classification and marking

Q.30 What is policing ? Explain criteria for policing.

Ans. : Policing is the regulation of the rate at which packet flow is injected into the network.

Criteria for policing

- Three important policing criteria are identified, these are :

1. Average rate 2. Peak rate 3. Burst size

1. Average rate : Average rate is defined as packets per time interval. The average rate of packets in a network can be limited as a policy. This limits the traffic in the network for a long period of time.

2. Peak rate : Peak rate is defined as maximum number of packets that can be sent over a short period of time over a network.

3. Burst size : Burst size is the maximum number of packets that can be sent into the network over a extremely short interval of time.

Q.31 What are the techniques to improve Quality of Service (QoS) ?
☞ [SPPU : Dec.-19, End Sem, Marks 6]

Ans. : • There are four techniques to improve quality of service : Scheduling, traffic shaping, resource reservation and admission control.

- An admission control, which is a quality of service mechanism, can also prevent congestion in virutal circuit networks. Admission control in ATM operates at the connection level and is therefore called connection admission control.
- Switches in a flow first check the resource requirement of a flow before admitting it to the network.
- A router can deny establishing a virtual circuit connection if there is congestion in the network.
- A source initiating a new flow must first obtain permission from an admission control entity that decides whether the flow should be accepted or rejected.
- The QoS may be expressed interms of maximum delay, loss probability, delay variance, or other performance parameters. If the quality of service of the new flow can be satisfied without violating QoS of existing flows, the flow is accepted; otherwise, the flow is rejected.

Q.32 Explain token bucket and leaky bucket algorithm with diagram.
☞ [SPPU : Dec.-15, Marks 8]

Ans. : Traffic shaping is an open loop method of congestion control. Two types of algorithm are used for traffic shaping.
1. Leaky bucket algorithm 2. Token bucket algorithm

Leaky Bucket Algorithm

- Leaky bucket i.e. a bucket with a small hole in the bottom is used to store the water. The outflow from hole is at constant rate and irrespective of rate of entering water. Once the bucket is full, any additional water entering it spills over the sides and is lost.
- The same idea can be applied to packets. This is similar to a single server queueing system with constant service time.

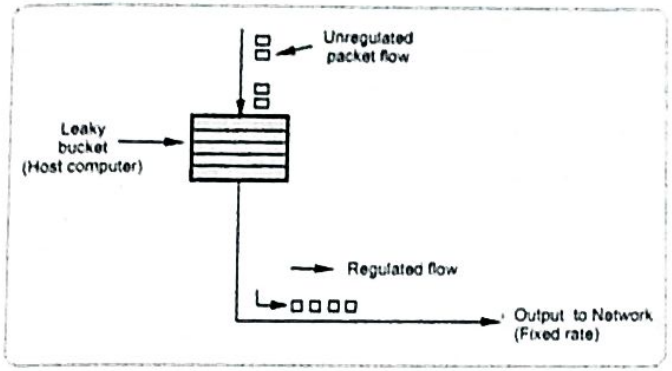


Fig. Q.32.1 Leaky bucket regulator

- Each host is connected to network with a finite internal queue. The host is allowed to put one packet per second on to the network. If a packet

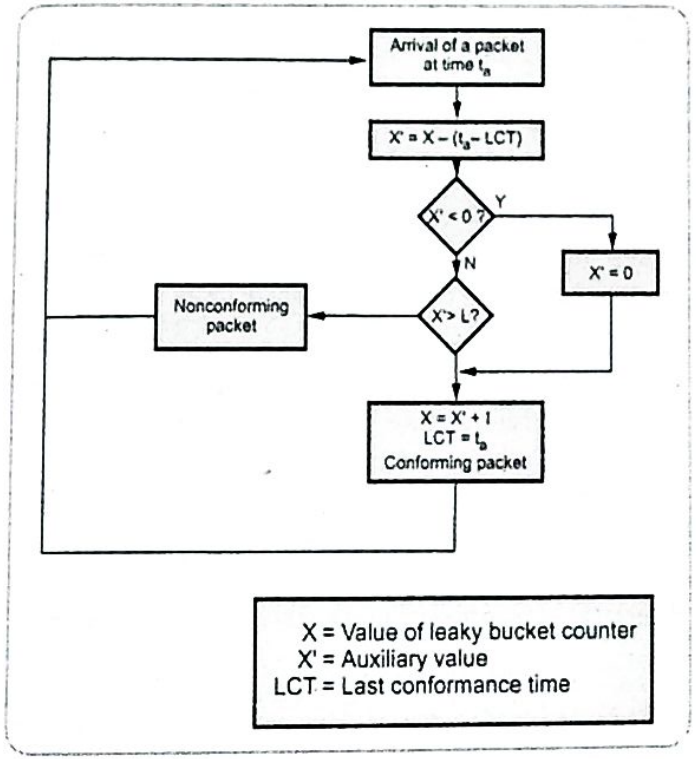


Fig. Q.32.2 Leaky bucket algorithm

arrives at the queue when it is full, the packet is discarded. This mechanism turn an unregulated traffic of the host regulated traffic on the network. Thus bursty traffic is smoothening and chances of congestion are reduced. Fig. Q.32.2 illustrates this algorithm.

- A leaky bucket regulator allows controlling the average rate, largest burst from a source. A leaky bucket regulator has both a packet bucket and a data buffer. Packets that arrive to the regulator that cannot be sent immediately are delayed in the data buffer.

Leaky Bucket Algorithm

- Fig Q.32.2 shows leaky bucket algorithm.

  X = Value of leaky bucket counter

  X' = Auxiliary value

  LCT = Last conformance time

- At the arrival of the first packet, the content of the bucket X is set to zero and the last conforming time is set to the arrival time of the first packet. The depth of the bucket is L+I where L typically depends on the traffic burstiness.

Token Bucket Algorithm

- The leaky bucket holds tokens. These tokens are generated by a clock at the rate of one token for every T sec. In token bucket bursts of up to n packets can be sent at once, which gives faster response to sudden bursts of input.
- The regulator collects tokens in a bucket, which fills-up at steady drip rate by packets. When a packet arrives at the regulator, the regulator sends the packet if the bucket has enough tokens. Otherwise, the packet waits either until the buckets has enough tokens. If the bucket is already fill of tokens, incoming tokens overflow and are not available to future packets. Thus, at any time, the largest burst a source can send into the network is roughly proportional to the size of the leaky bucket.
- Fig. Q.32.3 shows token bucket regulator.
- The regulator delays a packet if does not have sufficient number of tokens for transmission. A counter keeps track of tokens, the counter is

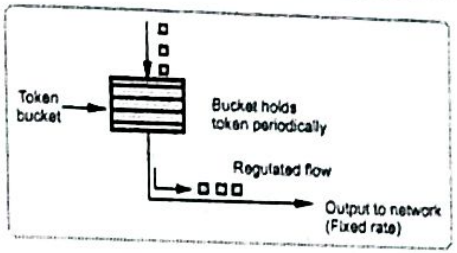


Fig. Q.32.3 Token bucket regulator

incremented by one every T and decremented by one whenever a packet is sent. When the counter hits zero, no packets may be sent. Smoother traffic can be obtained by putting a leaky bucket after the token bucket.

Q.33 Compare leaky bucket and token bucket.

Ans. : Comparison between leaky bucket and token bucket

Sr. No.	Leaky Bucket (LB)	Token Bucket (TB)
1.	Leaky bucket discards packets.	Token bucket discards tokens.
2.	With LB, a packet can be transmitted if the bucket is not full.	With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.
3.	LB sends the packets at an average rate.	TB allows for large bursts to be sent faster by speeding up the output.
4.	LB does not allow saving, a constant rate is maintained.	TB allows saving up tokens (permissions) to send large bursts.

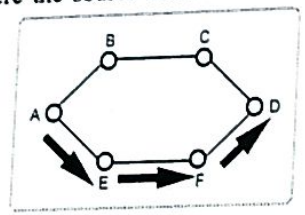
Q.34 Write short note on choke packet.

Ans. : • Another mechanism for congestion control is by using choke packets. This choke packet will have the effect of stopping or slowing down the rate of transmission from sources and hence limit the total number of packets in the networks. This approach requires additional

traffic on the network during a period of congestion. This can be applicable to both virtual circuit and datagram subnets.

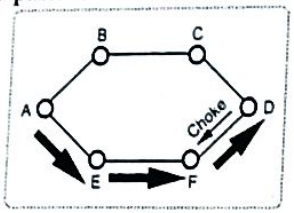
- When line utilization increases above some specific value called threshold, the line enters a 'alarming' situation. Each newly arriving packet is checked to see if its output line is in alarming state. If so, the router sends the said choke packet back to the source. This choke packet contains the destination address, so the source will not generate any more packets along the path.
- The traffic is reduced by adjusting parameters window size or leaky bucket output rate. Typically, the first choke packet causes the data rate to 50 % of its previous value the choke packet reduces the traffic to 25 % and so on.
- Congestion control using choke packets can be done by two ways. In first type the choke packet affects only source and in the second type the choke packet affects each hop it passes through. Fig. Q.34.1 shows choke packet that affects only the source.
- Fig. Q.34.1 shows a choke packet that affects each hop it passes through.

i) A subnet with six nodes A, B, C, D, E and F is shown in Fig. Q.34.1 (a). Here the source node is A and destination node is D.



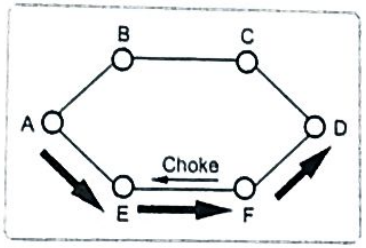
(a)

ii) When link utilization increased above its threshold, destination node D starts sending choke packets towards source node A.



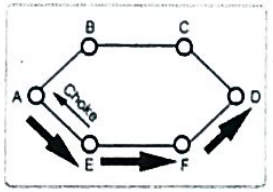
(b)

iii) The choke packets travels through the shortest or same path as that of packets.



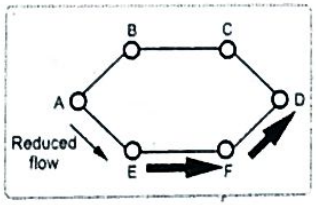
(c)

iv) The choke packet reaches to the source node A.



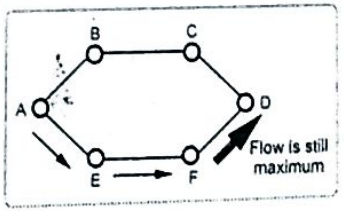
(d)

v) After receiving first chock packet source node A reduces its flow towards destination.



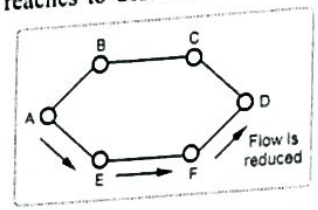
(e)

vi) The reduced packet flow follows the same reversed path i.e. the path of choke packets through various nodes.



(f)

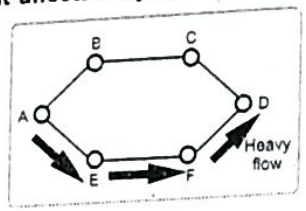
vii)The reduced flow reaches to destination node D.



(g)

Fig. Q.34.1 A choke packet that affects only the source

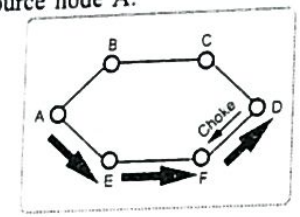
Fig. Q.34.2 shows a choke packet that affects each hop it passes through.



(a)

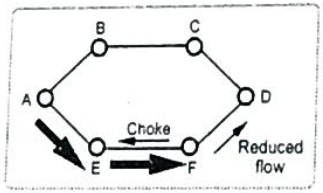
i) For the same subnet having nodes A, B, C, D, E and F source node and destination node D.

ii) When link utilization increased above its threshold destination node D starts sending choke packets towards source node A.



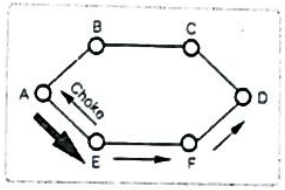
(b)

iii) The chock packets follows the exactly reversed path of traffic flowing packets. Here the chock packet reaches to node F. Immediately after reaching choke packet at node F, the traffic flow towards node D reduces.



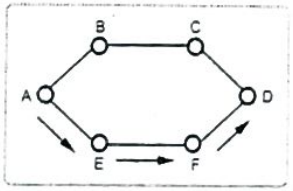
(c)

iv) As choke packet crosses node E, the traffic flow between nodes E, F, and F, D is reduced.



(d)

v) After reaching choke packet to source node A, the traffic flow between node A and node E and hence up to the destination node D is reduced.



(e)

Fig. Q.34.2 A choke packet that affects each hop it passes

Hop-by-hop choke packets

- Over long distances or at high speeds, the choke packets are not very effective.
- A more efficient way is to send choke packets hop-by-hop.
- A congested node would again generate a choke pocket, but each hop would be needed to reduce its transmission even before the choke packet arrives at the source.

4.9 : TCP and UDP for Wireless Networks

Q.35 Write short note on TCP and UDP for wireless networks.

Ans. : • TCP should not care whether IP is running over which media i.e. fiber or radio. Wireless transmission links are highly unreliable. They lose the packets all the time. The proper approach to dealing with lost packets is to send them again and as quickly as possible. When a packet

is lost on a wired network, the sender should slow down. When one is lost on a wireless network, the sender should try harder. When the sender does not know what the network is, it is difficult to make the correct decision.

- Frequently, the path from sender to receiver is inhomogeneous. It uses both, wired network and wireless network. The wireless TCP is split into two separate connection, as shown in the Fig. Q.35.1.

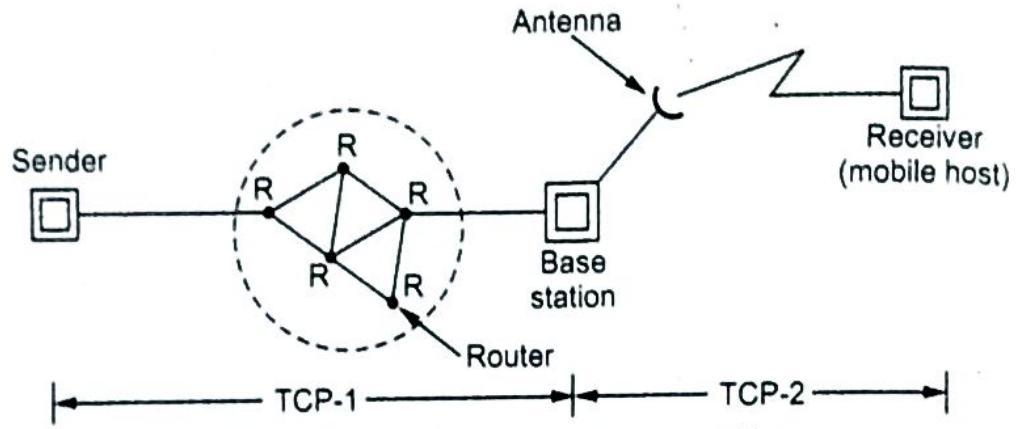


Fig. Q.35.1 Wireless TCP connection

- In TCP-1, connection goes from the sender to the base station. i.e. first stage. Mobile host (Receiver) gets data from base station antenna in TCP-2. The base station simply copies packets between the connections in both directions. This scheme solve the time out problem. Both connection, sender to base station and base station to receiver are homogeneous.
- Timeouts on the first connection can slow the sender down, whereas timeout on the second one can speed it up. Other parameter can also be tuned separately for two connections.
- The disadvantage of this scheme is, to break the TCP into two segments. UDP does not suffer from the same problem as TCP, wireless communication also introduces difficulties for it. The main trouble is that programs use UDP expecting it to be highly reliable. Wireless communication also affects areas other than just performance.

END...