mood-book

UNIT 4

Transport Layer

Syllabus

Transport Layer: Introduction and Transport Layer Services : Relationship Between Transport and Network Layers, Overview of the Transport Layer in the Internet, Multiplexing and Demultiplexing, Connectionless Transport: UDP – UDP Segment Structure, UDP Checksum, Principles of Reliable Data Transfer – Building a Reliable Data Transfer Protocol, Pipelined Reliable Data Transfer Protocols, Go- Back-N(GBN), Selective Repeat(SR), Connection Oriented Transport: TCP – The TCP Connection, TCP Segment Structure, Round-Trip Time Estimation and Timeout, Reliable Data Transfer, Flow Control, TCP Connection Management, Principles of Congestion Control – The Cause and the Costs of Congestion, Approaches to Congestion Control.

Learning Objectives

- Services provided by Transport Layer
- Role of Transport Layer in the Internet
- Various approaches that can be adopted on Transport Layer like Multiplexing and Demultiplexing
- Various Terminologies associated with UDP
- Providing Reliable Data Transfer Protocol
- Protocols used on Transport Layer like Go-Back-N and Selective Repeat
- Various Terminologies associated with TCP
- Various Congestion Control approaches.

Introduction

Transport Layer is the fourth layer of OSI reference model and is located above the network layer. The primary objective of this layer is to provide reliable and efficient end-to-end delivery of a message. The services provided by the transport layer are connection oriented services and connection less services. UDP (User Datagram Protocol) is a transport layer protocol defined for using the IP network layer protocol. It provides a best-effort datagram service to an end system. It is never used to send important data such as webpages, data base's etc., because it is a connectionless and unreliable protocol. Where as, TCP is a connection-oriented protocol and is the most commonly used protocol in the transport layer. It supports the connection management facilities of the Internet.

PART-A SHORT QUESTIONS WITH SOLUTIONS

Q1. What is a Transport Layer?

Answer :

Model Paper-II, Q1(g)

Transport layer is the fourth layer in the OSI reference model. It lies between the Network layer and session layer.

1. The transport layer is responsible for accepting data from the session layer, dividing it into small pieces (TPDUS), if required and passing these pieces to the network layer and assure the correct reception of data at the other end.
2. It sets up and terminates the connections across the network thereby, regulating the flow of information.
3. It also multiplexes the data and establishes multiple connections, when a high throughput is desired.

The transport layer allows the reliable transfer of data from source to destination. The layer defines two end-to-end protocols namely, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) to interact with other layers in the model.

Q2. What are the various transport layer protocols?

Answer :

The various transport layer protocols are as follows,

1. **Simple Protocol**

   Simple protocol is a connectionless protocol with no error and flow control techniques. In this protocol, the sender forwards packets and the receiver receives packets respectively.

2. **Stop and wait Protocol**

   Stop and wait protocol is the second protocol of the Transport layer. It is a connection oriented protocol with both flow and error control. Here, the sender and receiver makes use of a sliding window whose size is 1.

3. **Go-back-N Protocol**

   This protocol is used to send multiple packets without even receiving the acknowledgments. However, the receiver is capable of buffering only one packet.

4. **Selective Repeat Protocol**

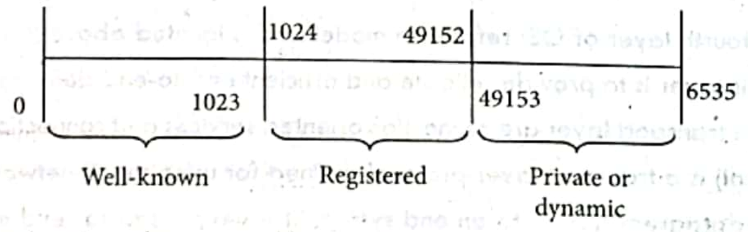
   UDP is a transport layer protocol defined for use with the IP network layer protocol. It provides a best-effort datagram service to an end system. The service provided by UDP is an unreliable service that provides no guarantee for delivery and no protection from duplication. The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases. A computer may send the UDP packet without first establishing a connection with the receipt. It completes the appropriate field in UDP header and forwards the data together with the header for transmission by the IP network layer.

Q3. What do you mean by the port number?

Answer :

Model Paper-III, Q1(g)

Port numbers are local addresses which are used to identify various programs executing at a time. Every port number identifies a unique function used over networking. In TCP/IP protocol suite, these port numbers range from 0 to 65, 535 based on their usage. They are well-known, registered and Dynamic or Private port number.



In transport layer protocol, port numbers are used in process-to-process communication to provide end-to-end addressing, and also to enable multiplexing and de-multiplexing at this layer similar to IP addressing. Some of the well-known port numbers are,

(i) Port Number: 80 – for invoking HTTP (HyperText-Transfer Protocol)

(ii) Port Number: 67 – for invoking DHCP

(iii) Port Number : 11 – for showing active users.

Q4. What are the services provided by transport layer protocol?

Answer :

Transport layer is the fourth layer of ISO/OSI model. The primary objective of this layer is to provide reliable and efficient end-to-end delivery of a message. In order to achieve this goal, it is necessary for transport layer to utilize the services provided by its lower layer i.e., network layer. Transport entity is responsible for carrying out the operations within the transport layer. Similar to network layer, transport layer even provides two types of transport services. They are,

1. Connection oriented services
2. Connectionless services.

The functionality of these services is very much similar to the services provided by the network layer i.e., connection-oriented transport services are similar to connection-oriented network services, and connectionless transport services are similar to connectionless network services.

Q5. What is UDP?

Answer : Model Paper-II, Q1(h)

UDP is a transport layer protocol defined to use with the IP network layer protocol. It provides a best-effort datagram service to an end system. The service provided by UDP is an unreliable service that provides no guarantee for delivery and no protection from duplication. The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases. A computer may send the UDP packet without first establishing a connection with the receipt. It completes the appropriate field in UDP header and forwards the data together with the header for transmission by the IP network layer. UDP header is a fixed-size header of 8 bytes which contains protocol control information.

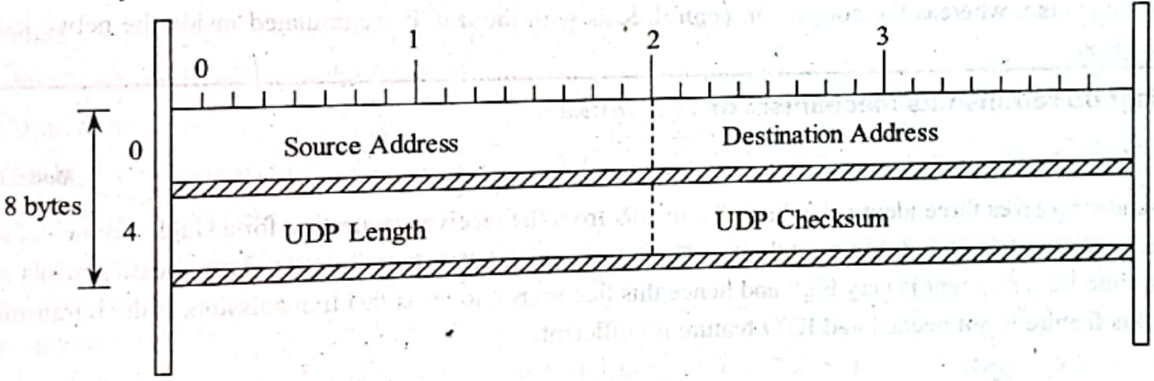


Figure: UDP Header

Q6. What is TCP?

Answer : Model Paper-I, Q1(g)

TCP is a connection oriented protocol and most commonly used protocol in the transport layer. It supports the connection management facilities of the Internet.

TCP can be used to setup a logical connection between two entities and to transfer a sequence of bytes between them. A key feature of TCP is that it provides connection oriented user-to-user byte stream service. Every byte in a TCP connection consists of a unique 32-bit sequence number that is used for acknowledgments.

The exchange of data between any two TCP entities is done using segments. The TCP segment has a fixed 20-byte header and an optional part which is followed by data bytes. The length of a segment is determined by TCP software.

Q7. Differentiate between TCP and UDP.

Answer :

The TCP is a connection oriented service and connection is established between client and server before transmission. In case if multiple clients transmits packets to server at the same time, each client is serviced using separate connection. Hence, unlike UDP the server is allowed to process multiple client request at the same time. So in TCP, server is always in concurrent mode as it is serving many client request concurrently.

Q8. Define congestion control.

Answer : Model Paper-I, Q1(h)

Congestion occurs due to the load on the network. If the number of packets go beyond the capability of the network, it leads to congestion. The mechanism that helps in controlling the congestion is called congestion control. There are two types of congestion control techniques, they are,

(a) Closed loop

(b) Open loop.

(a) Closed Loop

Closed loop are much like the feedback systems in which the metrics are used to monitor the status of the system.

(b) Open Loop

The open loop solutions ensures that the congestion never occurs which can be done by implementing good plans and designs.

Q9. What is the difference between congestion control and flow control?

Answer :

Flow control is mainly concerned with controlling the amount of data that is being transmitted between a sender and a receiver. Whereas, congestion control is concerned with the network's ability to handle the amount of incoming traffic at any given time. This traffic depends upon router's capability, its behavior etc. In other words, flow control deals with the traffic accumulated at the receiving machine, whereas the congestion control deals with the traffic accumulated inside the network. Consider the following examples.

Q10. How fast do retransmits mechanism of TCP works?

Answer : Model Paper-III, Q1(h)

If the sender receives three identical acknowledgments from the receiver requesting for a single segment, then the sender retransmits that segment, from the queue even though its retransmission timer doesn't expire. This situation arises only when the retransmission time for a segment is very high and hence this feature is known as fast transmission. If the retransmission time is too less, then this feature is not needed and RTO feature is sufficient.

PART-B ESSAY QUESTIONS WITH SOLUTIONS

4.1 INTRODUCTION AND TRANSPORT LAYER SERVICES: RELATIONSHIP BETWEEN TRANSPORT AND NETWORK LAYERS, OVERVIEW OF THE TRANSPORT LAYER IN THE INTERNET

Q11. Discuss the services/responsibilities of transport layer.

Answer :

Model Paper-I, Q8(a)

Services/Responsibilities of Transport Layer

The following are the responsibilities of transport layer,

1. **Service-point Addressing**

Transport layer introduces a type of address called service point address. This address is included in the header of transport layer.

Computers are capable of running several programs simultaneously . The term source-to-destination delivery doesn't simply mean delivery of data from one system to the other, but it also means delivery of data from one specific process to the other. Therefore, in this case it becomes necessary to include a service point address in the header of transport layer.

2. **Segmentation and Reassembly**

Transport layer divides a single message into various transmittable segments wherein every segment contains a sequence number. This sequence enables the layer to reassemble the message once it reaches the destination. It also enables the layer to identify the lost packets during transmission and to replace them as well.

3. **Connection Control**

A transport layer can be either of the following,

(i) **Connection Less**

This layer considers segment as an individual packet . These packets are then individually delivered to the destination.

(ii) **Connection Oriented**

This layer initially establishes a connection with the transport layer of destination machine. It then transfer the entire data and terminates the connection after transmission.

4. **Flow Control**

Flow control of this layer is similar to the flow control of data link layer. But the only difference is that, in data link layer, flow control is performed across a single link whereas in transport layer end-to-end flow control is performed.

5. **Error Control**

Error control of this layer is similar to the error control of data link layer. But the only difference is that, in data link layer error control is performed across a single link where as in transport layer, process-to-process error control is performed. While sending the data, transport layer of sender's machine ensures that complete data reaches the transport layer of receiver's machine. In case if an error occurs, the entire data is retransmitted.

Q12. Explain the relationship between network and transport layers.

Answer :

The network layer offers logical communication among host while transport layer offers logical communication among host processes. The relationship between these two layers remain the services provided by the network layer to the transport layer.

Providing Services to the Transport Layer

The function of the network layer is to provide services to the transport layer. These services are provided at the network-transport layer interface. This interface is important because it acts as a frequent medium between the carrier (the boundary of the subnet) and the customers. The job of the carrier is to send all the packets passed to it by the customers. Therefore, it must be designed well.

The network layer services are designed such that they are independent of the subnet organization, the details of subnet topology is hidden from the transport layer and the uniform network addresses are available to the transport layer across LANs and WANs.

There are arguments on whether the network layer should provide the connectionless service or connection-oriented service.

One argument is that the job of the subnet is just to move bits from one end system to another. No matter how the subnet is designed, it is inherently unreliable. Therefore, the hosts should do packet reordering, error control and flow control themselves.

With network layer providing connectionless service only two primitives SEND PACKET and RECEIVE PACKET are required. The packet reordering, error control and flow control is all left to the hosts because, any how they do this job and there is little to be gained by doing it twice. Further, each packet must contain the full destination address because each packet is routed independently of its predecessors.

Another argument is that a reliable, connection-oriented service should be provided at the subnet.

A network layer that provides connection-oriented service between two end systems must establish a connection between them before sending data. This is the virtual connection which is given a special identifier. The identifier is used to send all the data. After all data have been sent, the connection is released. The two end systems decide the parameters, quality and cost of the service to be provided at the time the connection is set up. The data flows in both directions.

With this service the packets are delivered to destination end system in sequence and flow control is provided automatically.

In connectionless service the complexity is in the transport layer (i.e., hosts). In connection-oriented service, the complexity lies in the network layer (i.e., subnet).

Q13. Discuss about transport layer in the Internet.

Answer :

Internet uses two transport layer protocols called TCP and UDP. The data packets of transport layer protocols for TCP are referred as segments. The transport layer using Internet or IP, offers the following services,

For remaining answer refer Unit-IV, Q11.

4.2 MULTIPLEXING AND DEMULTIPLEXING

Q14. Write notes on multiplexing and demultiplexing.

Answer : Model Paper-III, Q9(b)

The service of multiplexing and demultiplexing is necessary for all computer networks. Extending host-to-host delivery to process-to-process delivery is called transport layer multiplexing and demultiplexing.

At the receiving host, transport layer is responsible to deliver the data present in the segments Transport Layer (PDVU's) to the correct process running on the host. For instance, consider an example that user is working on the computer system and the user is downloading the web pages while running one FTP session and two Telnet sessions. Thus, the computer has four application processes such as two **Telnet** processes, one **FTP** process and one **Http** process. The transport layer must send the data collected the network layer to any of the four processes.

Each and every process contains a socket. The transfer of data from process to network and from network to process takes place completely through socket i.e., socket is the door between application process and TCP. Thus, in the receiving host, transport layer transmitts data to the intermediary socket rather than transmitting data directly to the process.

In the receiving host, there can be more than one socket, where each socket contains a unique indentifier which distinguishes one socket from another.

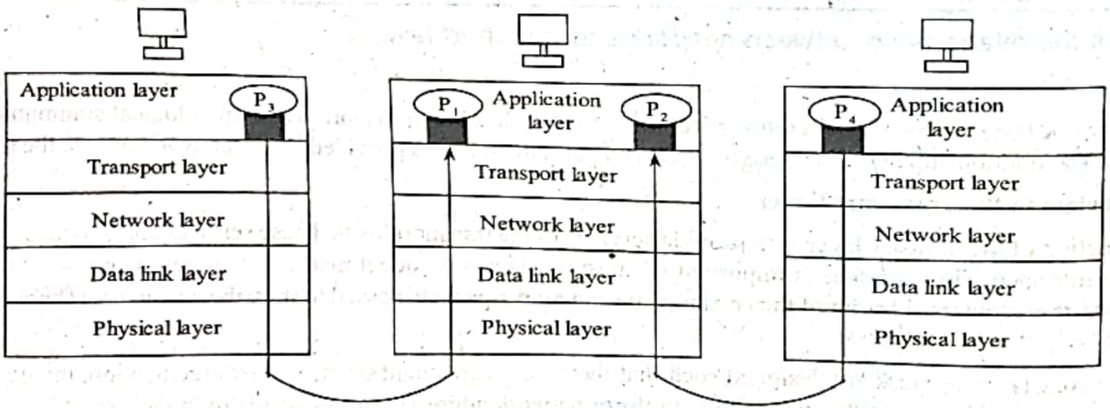


Figure: Multiplexing and Demultiplexing in Transport Layer

Each segment of transport layer contains a set of fields which is used by the receiving end to identify the receiving socket so that the segment can be transmitted to the correct socket. Thus based on the above scenario it can be said that,

Demultiplexing refers to process of transmitting the data of transport layer segment to the appropriate socket.

The data chunks collected from different sockets at the source host are encapsulated with header information to create segments and the process of sending these segments to the network layer is known as **multiplexing.**

In the middle host, transport layer is responsible to carry out two things.

(i) Demultiplexing segments coming from network layer to either process P_1 or P_2 which is carried out by sending the data to the socket of corresponding process.

(ii) Collecting outgoing data from the sockets, creating transport layer segments and then directing these segments to the network layer.

In transport layer, multiplexing and demultiplexing is carried out in a host as follows. It is known that, in transport layer, multiplexing needs two things.

(i) Each socket must contain unique identifier.

(ii) Each segment contain special fields like **source port number field** and **destination port number field** in order to identify the receiving socket so that the segment can be directed to the correct socket. Each port number refers to 16-bit number, whose range is from 0 to 65535. Among these port numbers only well known port numbers (whose range is from 0 to 1023) are used for application protocols like HTTP.

Demultiplexing is implemented as follows a port number is allotted for each and every socket in the host and when a segment arrives to the receiving host, the transport layer examines the fields of segments for destination port number to identify the receiving socket and then directs the segment to that socket.

Q15. Explain in detail about connection-oriented multiplexing and demultiplexing.

Answer :

The protocol which provides a reliable, connection-oriented service to the application is TCP (Transmission Control Protocol). The difference between UDP socket and TCP socket is UDP socket is a two tuple whereas TCP socket is four tuple. The four values of TCP socket include source address, source port number, destination IP address and destination port number, which are used by the receiving host to direct the segment to the appropriate socket unlike UDP two TCP segments having different source IP address or source numbers are sent to the two different sockets.

TCP demultiplexing can be understood by taking a closer look at TCP sockets and TCP connection establishment.

TCP server application contains welcoming socket, which waits for connection establishment requests from clients. With server process running, the client process can initiate a TCP connection to the server. This is done in the client program by creating a socket object.

When the client creates its socket object it specifies the address of the server process, namely the IP address of the server and the port number of the process which is shown below.

Socket client socket = new socket (“ServerHostName”, 6789);

The above line creates a socket for the client process through which transmission of data takes place usually and connection establishment request means a TCP segment containing destination port number (6789) as well as specifying connection establishment but set in the TCP header.

When the server receives connection establishment request from the client it tells the process to create a connection socket.

Socket Connection socket = welcomesocket.accept( );

In the connection request segment, the server notes source port number, source IP address, destination port number and destination IP address. These four values are used to demultiplex the TCP segment to the correct socket.

Consider host *A*, host *B* and server *C* where host *A* initiates one HTTP session to sever *C* and host b initiates two HTTP sessions to server *C*. The IP addresses of host *A* and *B* and server *C* are *A*, *B* and *C* respectively. Host *A* assigns source port number of 26145 to its HTTP connection and host *B* assigns two source port numbers to its two HTTP connections. Server *C* directs the segments to the appropriate socket even though the connections have same source port numbers with different source IP addresses.

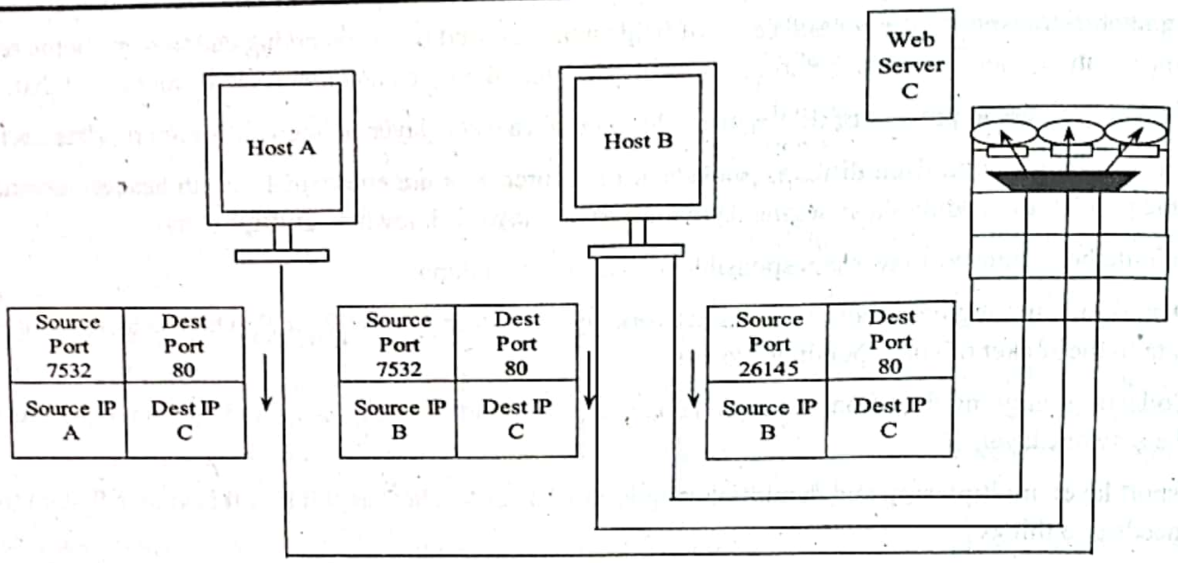


Figure: Hosts A and C using Same Destination Port Number to Communicate with Same Web Server Application

Q16. Discuss about connectionless multiplexing and demultiplexing.

Answer :

Connectionless Multiplexing and Demultiplexing

The protocol which provides unreliable connectionless service to the application is UDP (User Datagram Protocol). UDP multiplexing and demultiplexing can be understood by the port numbers assigned to UDP sockets.

UDP sockets can be created as follows,

DatagramSocket Socket1 = new DatagramSocket( );

Here, transport layer is responsible to automatically assign a port number ranging from 1024 to 65535 to the UDP socket. UDP sockets can also be created by specifying the port numbers.

DatagramSocket Socket2 = new DatagramSocket(19158);

Here, application is responsible to assign a specific port number (i.e., 19158) to the UDP socket usually, at the client side, transport layer is allowed to assign a port number whereas at the server side, application is allowed to assign a port number.

A process in host A with port number **19158** can transmit data to a process in host B with port number **46428** as follows,

Initially, in host A, transport layer creates a transport layer segment (PDU) consisting of application data, source port number and destination port number. These segment is then passed to the network layer, which it encapsulates in an IP datagram and sends that segment to the receiving host.

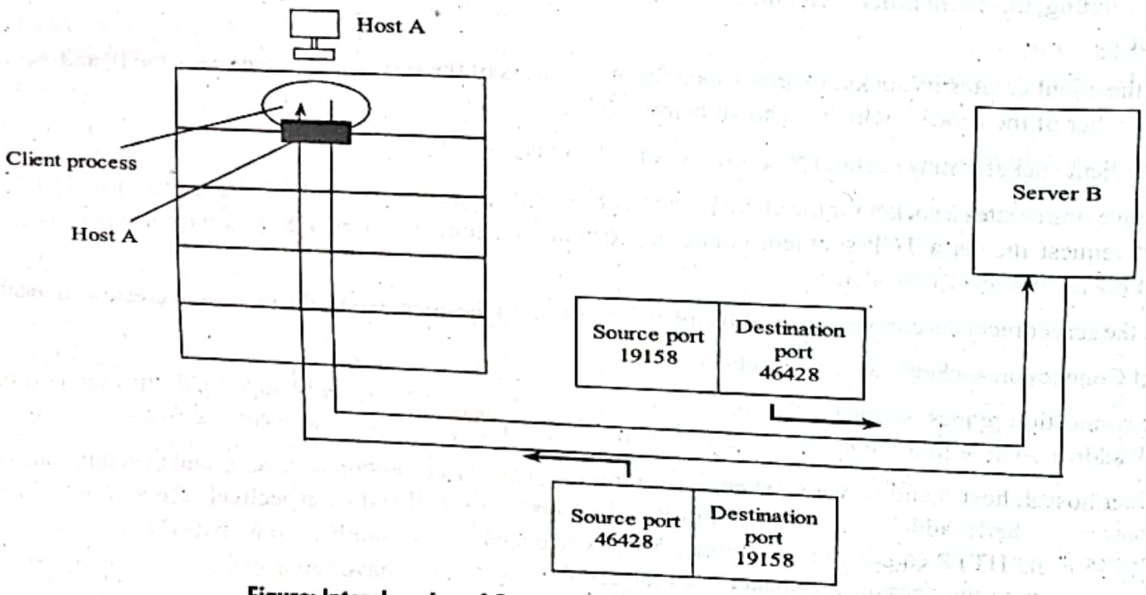


Figure: Interchanging of Source and Destination Port Numbers

At the receiving end, transport layer in host B examines the destination port number to identify the receiving socket and then directs the segment to the appropriate socket.

UDP socket is a two tuple consisting of destination **IP address** and **Destination port number.** Two arriving UDP segments with different source IP address and source port numbers but with same destination IP address and destination port number then the two segments are passed to the same destination process through the socket with same destination port number.

When host B wants to send a segment back to host *A* then host *B* extracts the source port number from the *A*-to-*B* segment and uses that source port number as a destination port number in the *B*-to-*A* segment.

4.3 CONNECTIONLESS TRANSPORT: UDP – UDP SEGMENT STRUCTURE, UDP CHECKSUM

Q17. What is UDP? List its features and explain its header.

Model Paper-I, Q8(b)

Answer :

UDP

UDP is another commonly used connectionless, unreliable protocol on the Internet. UDP is never used to send important data such as webpages, databases, etc. UDP is a connectionless, unreliable protocol. UDP is a transport layer protocol defined for use with the IP network layer protocol. It provides a best-effort datagram service to an end system. The service provided by UDP is an unreliable service that provides no guarantee for delivery and no protection from duplication. The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases. A computer may send the UDP packet without first establishing a connection with the receipt. It completes the appropriate field in UDP header and forwards the data together with the header for transmission by the IP network layer.

Features of UDP

The two important features of UDP are,

(i) **Simple**

It is a simple protocol that uses a very straightforward messaging structure that is similar to the message format used by many other TCP/IP protocols. The real goal of this protocol is to serve as an interface between networking application processes running at the higher layer and the internetworking capacities of IP. Like TCP, UDP layer is on the top of IP which is a method of addressing through the use of UDP port numbers. It does include an optional checksum capability for an error-detection but adds virtually no other functionality.

(ii) **Fast**

UDP is a fast protocol as it doesn't require error-detection mechanism. This make it unsuitable for use in many networking applications.

UDP was developed for use by application protocols that do not require reliability, acknowledgment or flow control features at the transport layer. It is designed to be simple and fast providing transport layer addressing in the form of UDP ports and an optional checksum capability.

UDP Header

UDP header is a fixed-size header of 8 bytes which contains protocol control information.

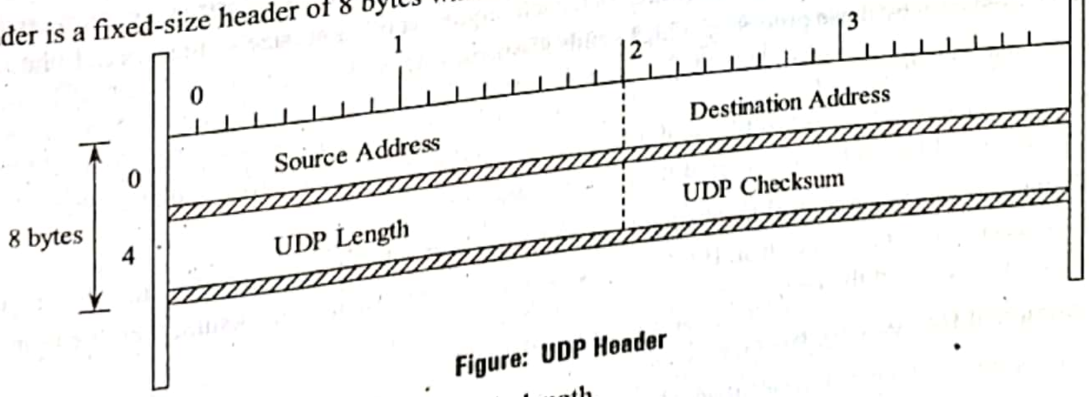


Figure: UDP Header

UDP header consists of four fields each of 2 bytes in length.

(i) **Source Address**

It specifies the port number of the sender which describes where a reply packet is to be sent. This can actually be set to zero if it is not used. For example if a user don't require a reply packet, then the source address for that packet can be set to zero.

(ii) Destination Address

It specifies the port number of the receiver where the packet is to be transmitted. If the destination host is a server, then the port numbers are reserved.

(iii) UDP Length

It specifies number of bytes comprising the combined UDP header information and payload data. The total length of UDP user datagram is 65,535 bytes, but as the datagram is stored within the IP packet, the length of UDP datagram is, 65535 – 20 = 65515 bytes.

i.e., UDP length = IP length – IP header length

(iv) UDP Checksum

This checksum is the same kind of checksum used in TCP header except that it contains a different set of data. It is computed in the same way as that of TCP. A checksum is used to verify that end-to-end data that has not been corrupted by routers or bridges in the network. If checksum is not required, the value zero is placed in this field in which case the data is not checked by the receiver, checksum comprises of pseudoheader, UDP header and UDP data. The pseudoheader serves the same purpose as that of TCP. But for UDP, the value for protocol is 17.

Q18. Define UDP checksum and discuss the operation of UDP.

Answer : Model Paper-II, Q8(a)

UDP Checksum

UDP checksum is defined as a safety feature in which its value represent an encoding of a datagram. Initially it's values is calculated by the sender and then by the receiver. If the given value of the sender doesnot matches with the receiver's data, then the UDP help's in detecting the error by using checksum algorithm. The checksum value of the UDP is mandatory for the sender but not to the receiver because if the given value is checksumed, then automatically the receiver gets the correct data send by the sender. If the sender checksum's contain all 0's then the receiver also receives all zeros.

Operation of UDP

The concepts of transport layer is used in UDP operation. The operation of UDP includes the following criteria,

1. Connectionless Services

UDP supports connectionless service, therefore, the establishment and termination of connections are not required in UDP operation. This causes each user datagram to travel on different path. Hence, all the user datagrams that are send by UDP are independent datagrams and are not numbered. The different user datagrams don't have any relationship among them, though they come from same source process or go to same destination program.

In a connectionless service, any process that is implementing UDP to transmit a datastream cannot predict that the UDP splits the data in to fragments when data is large. It ensures that each request is of small size so that it can fit into a user datagram. Thus, UDP should be used only by those processes which sends short messages.

2. Flow and Error Control

UDP is simple and unreliable transport protocol. Flow control and window mechanisms are not present in UDP protocol. This may result in overflow of incoming messages at the receiver end.

Also, UDP doesn't have error control mechanism (except for the checksum). Because of this, the sender is unaware of whether the message has been duplicated or lost. If the receiver detects an error using checksum, then the user datagram will be discarded without the knowledge of the receiver.

3. Encapsulation and Decapsulation

When messages are being sent from one process to another process, it gets encapsulated at the sender's side and decapsulated at the receiver's side by the UDP protocol.

4. Queuing

Queues of the UDP are associated with port numbers shown in the figure below,

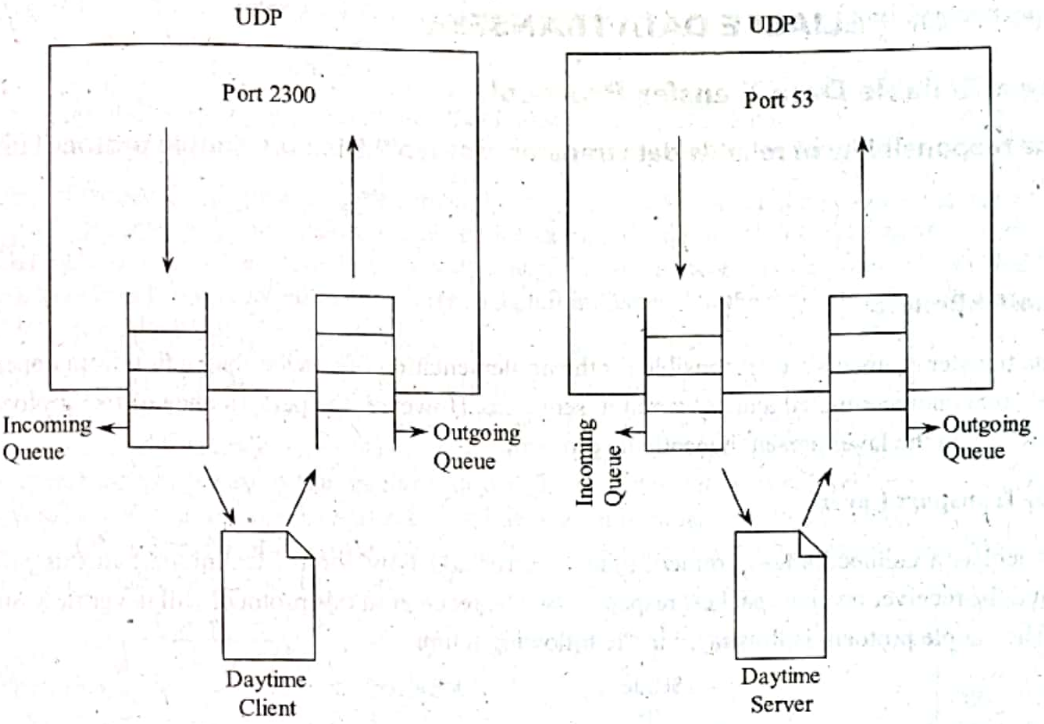


Figure: UDP with Queues

Two types of queues can be created for each ports of UDP they are incoming queue and outgoing queue. These queues help in performing certain implementations which can create both incoming queue and outgoing queue associated with each process whereas there are some other implementations that can create only incoming queue associated with each process.

When a process is initiated at the client site, it sends a request to the operating system for the port number. The operating system provides only one port number, one outgoing and one incoming queue, even if it wants to communicate with several processes.

Mostly the queues that are opened by the client are identified by ephemeral (temporary) port numbers. These queues does it process as long as the process runs and gets destroyed on the process termination.

Messages being sent by the client process using the requested source port number, will get stored into the outgoing queue. These messages are then removed by UDP one by one. After removing, UDP adds UDP header to it and then deliveres them to IP. The continuous sending of the messages may cause queue overflow. To stop this, the operating system asks the client process to delay message sending until the messages of the queue gets delivered.

If a message arrives at the client process (user datagram), then UDP examine its port number present in the field of destination port number to see whether an incoming queue has been created or not. It finds such queue then it stores the message to the queue end. But if the UDP doesn't finds such queue, it discards the message and informs the server by sending 'port unreachable' message using ICMP protocol. All the incoming messages for a client program whether from same or different sources, are sent to the same queue. The continuous receiving of incoming messages may cause incoming queue overflow. To present this, the UDP discards the user datagram and informs the server by sending a 'port unreachable' message. At the server site, the creation of queue is different but, simple when a server process is initiated it uses its well-known port number for creating incoming and outgoing queues. As long as server runs, the queues will function.

When a message arrives at the server then UDP examine its port number present in the field of destination port number to see whether an incoming queue has been created or not. If it finds such queue then stores the message to the end of the queue. But if it doesn't finds such queue, discards the user datagram and asks the ICMP protocol to send a 'port unreachable' message to the client.

All the incoming messages for a server program whether from same or different sources, are sent to the same queue. The continuous receiving of messages may cause overflow of incoming queue. To avoid this, the UDP discards the user datagram and informs the client by sending a 'port unreachable' message using ICMP protocol.

A server can respond to a client by sending a message to the outgoing queue using the source port number specified in the request. The messages from the queue is removed one by one by UDP and after adding the header delivers the messages to IP. The continuous message sending may cause outgoing queue overflow. To stop this overflow, the operating system asks the server process to delay message sending until the queue becomes empty.

4.4 PRINCIPLES OF RELIABLE DATA TRANSFER

4.4.1 Building a Reliable Data Transfer Protocol

Q19. What is the responsibility of reliable data transfer protocol? Discuss simple protocol used at transport layer.

Model Paper-III, Q8(a)

Answer :

Reliable Data Transfer Protocol

Reliable data transfer protocols are responsible for the implementation of service abstraction from upper layers. It ensures that the data packets remain uncomputed and delivered in sequence. However, the performance of this protocol can be effected if there exist inefficiency in the layer present beneath the protocol.

Simple Protocol in Transport Layer

Simple protocol is a connectionless protocol with no error and flow control techniques. In this protocol, the sender forwards packets and the receiver receives packets respectively. The receiver in this protocol will never flow out i.e., it will never be overwhelmed. The simple protocol is illustrated in the following figure.

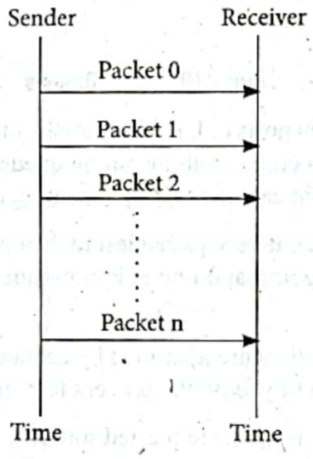


Figure (1): Simple Protocol

Finite State Machine for Simple Protocol

In this protocol, if application layer of the sender has a message to forward then only sender forwards the packet to the receiver and then the receiver forwards the message to its application layer when the packet is arrived. This scenario is shown using two finite state machines.

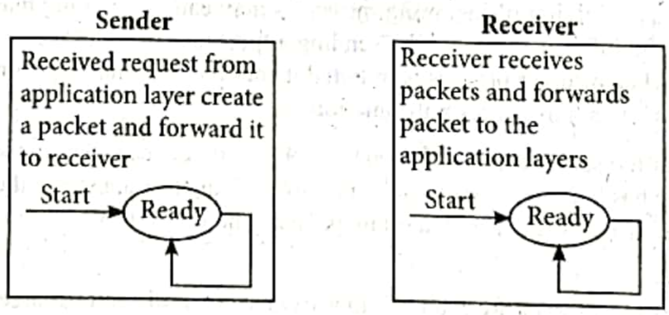

Sender
Received request from application layer create a packet and forward it to receiver
Start
Ready
Receiver
Receiver receives packets and forwards packet to the application layers
Start
Ready

Figure (2): FSM's for Simple Protocol

The above figure indicates two finite state machines for simple protocol. Both FSM's contains only one ready state. The sender stays in ready state until the arrival of request from the application layer and then the sender converts the message to packet and forwards it to the receiver. Then the receiver in steady state converts the message out of packet and delivers to the application layer.

Q20. Explain the stop and wait protocol of transport layer.

Answer :

Stop and wait protocol is the second protocol of the Transport layer. It is a connection oriented protocol with both flow and error control. Here, the sender and receiver makes use of a sliding window whose size is 1.

In the process of stop and wait protocol, the sender transmits one data packet at a single instance of time and waits until it receives an acknowledgement from the receiver before transmitting the other data packet. In order to identify whether the data packet are corrupted there is a checksum added to every data packet. That is, whenever a packet is received the receiver checks the packet, if the packet's checksum is found to be incorrect, then it indicates that the arrived packet is corrupted and the receiver silently discards the corrupted packet.

If there is no acknowledgement sent from the receiver after receiving the packet, then the sender believes that the packet transmitted is either corrupted or lost. Besides this, a fixed timer is also used inorder to record the time at which the packet is transmitted and received. If an acknowledgment is received before the timer expires, then the timer is switched off and the next data packet in the queue is transmitted to the receiver. In case if the timer gets expired and the sender does not receives any acknowledgment from the receiver then the sender retransmitted packet is either lost or corrupted. This means that the sender has to maintain a copy of every packet which is transmitted until it receives an acknowledgment from the receiver. The figure below shows the process of stop and wait protocol.

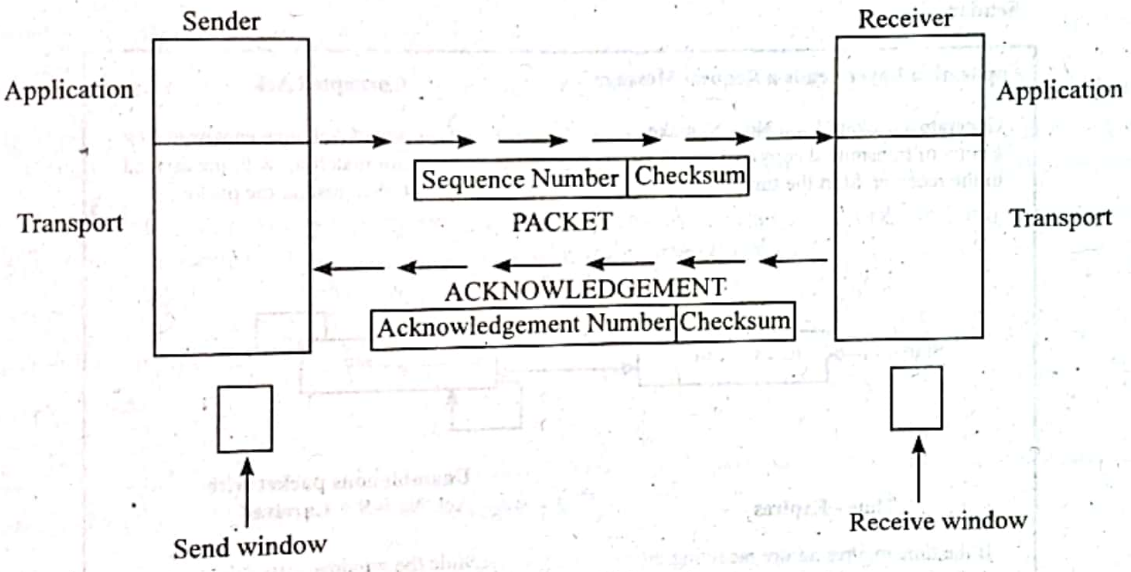


Figure: Stop and Wait Protocol

Sequence Numbers

The Sequence Numbers are the numbers that are added to the packet header. These numbers are used to prevent the packet duplication. The most important thing about the sequence numbers is the range of the sequence numbers. Every user demands a packet whose size is small, because the small data packets provides unambiguous communication.

Acknowledgement Numbers

The acknowledgement numbers are the numbers that announces the sequence number of the successive packet present in the queue. For instance, if packet 0 reaches safely. Then the receiver forwards an acknowledgement 1 (which means that packet 1 is required). If packet 1 is reached safely, then the receiver forwards an acknowledgement 0 (which means that the next packet 0 is required).

FSMs

To show the FSMs for the stop and wait protocol it is necessary for both the sender and the receiver to be in established state before sending and receiving the data packets.

Sender

Initially the sender is in the ready state and the variable 'S' is assigned to 0.

Ready State

In this state the sender keeps waiting for an event to occur. If it encounters a request, then it immediately generates a packet whose sequence number is set to 'S'. The sender creates a copy of the packet which is transmitted and it also sets the timer. After this the sender switches to the blocking state.

Blocking State

In this state there are three possible events that can occur,

1. If the sender receives an unambiguous acknowledgment from the receiver, then the packet whose sequence number was announced by the acknowledgment number is sent i.e., ack no = (s + 1) modulo 2 and the timer is switched off. After this, the sender is switched to the ready state.

2. If the sender receives a corrupted acknowledgement, with the ack No ≠ (s + 1) modulo 2, then that specific ACK is discarded.

3. If the timer expires, before receiving the acknowledgment, then the sender retransmitts the data packet and reset the timer.

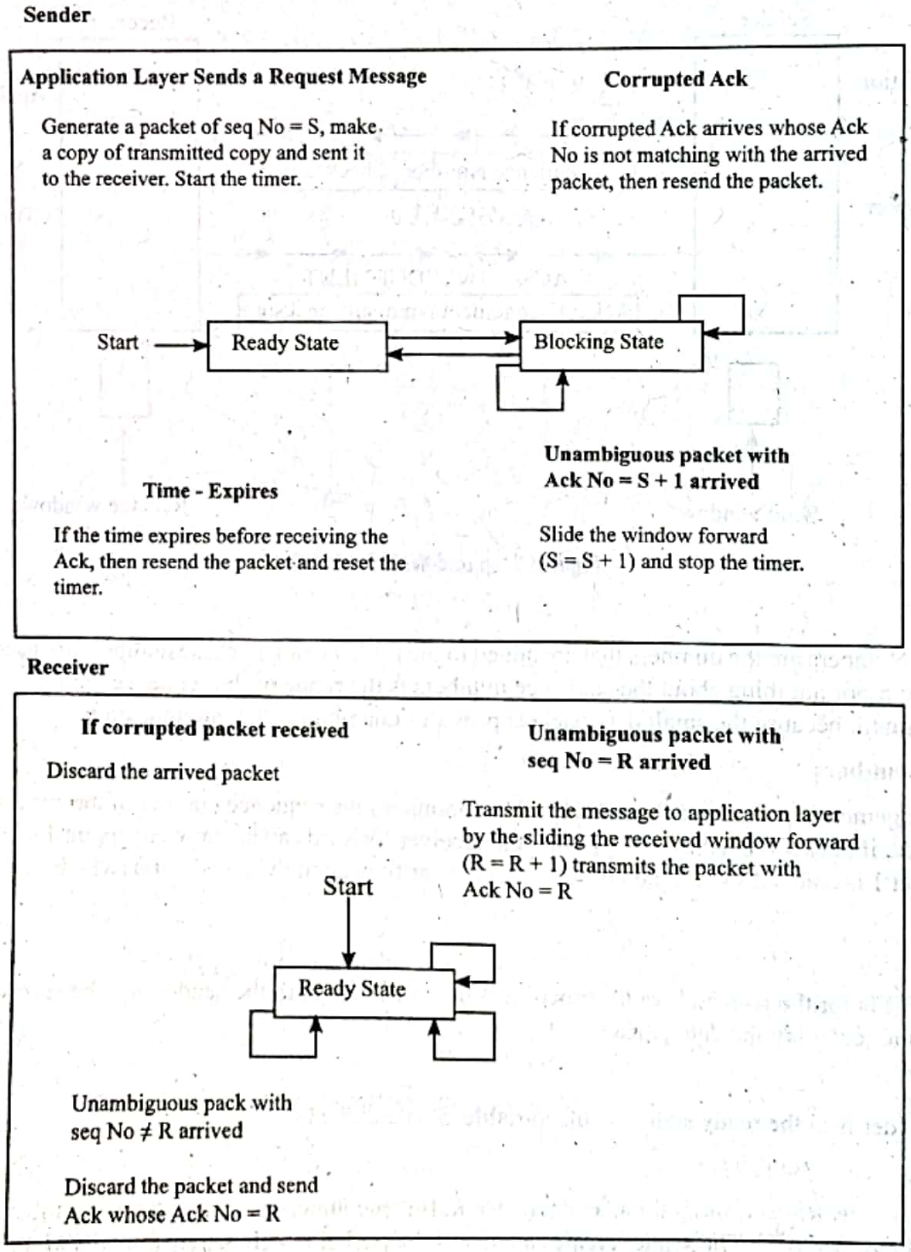
Receiver

The receiver is always found in the ready state. There are three possible events that can occur at receiver's side.

1. If an unambiguous packet whose seq No (sequence number) = R reaches the receiver, then the message of the arrived packet is sent to the application layer. After this, the window R = (R + 1) modulo 2 slides. Lastly the ACK whose ack No (acknowledgement number) = R is transmitted.

2. If an unambiguous packet whose seq No ≠ R reaches the receiver, then the arrived packet is discarded. And an ACK that has ack No = R is transmitted.

3. If the receiver receives a corrupted packet, then the arrived packet is discarded.



Application Layer Sends a Request Message

Generate a packet of seq No = S, make a copy of transmitted copy and sent it to the receiver. Start the timer.

Corrupted Ack

If corrupted Ack arrives whose Ack No is not matching with the arrived packet, then resend the packet.

Unambiguous packet with Ack No = S + 1 arrived

Slide the window forward (S = S + 1) and stop the timer.

Time - Expires

If the time expires before receiving the Ack, then resend the packet and reset the timer.

If corrupted packet received

Discard the arrived packet

Unambiguous packet with seq No = R arrived

Transmit the message to application layer by the sliding the received window forward (R = R + 1) transmits the packet with Ack No = R

Unambiguous pack with seq No ≠ R arrived

Discard the packet and send Ack whose Ack No = R

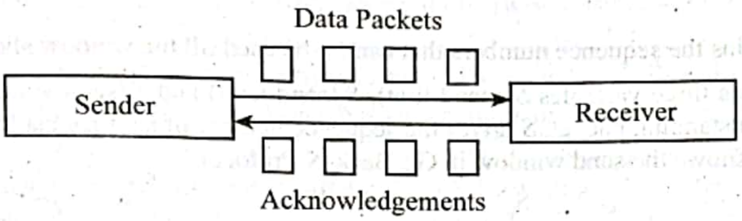
Figure: FSMs of Stop and Wait Protocol

4.4.2 Pipelined Reliable Data Transfer Protocols

Q21. Describe the pipelined protocol for reliable data transfer.

Answer : **Model Paper-III, Q9(a)**

The stop and wait protocol faces performance issues as it makes the transfer slow. To overcome such issues, pipelined protocol is used. In case of stop-and-wait protocol, the sender has to wait for the acknowledgement from the receiver to send the next packet which however eliminated using pipelined protocol. Here, packets are sent one after the other while the receiver sends the acknowledgements in sequence to the sender.



The approach reduces much of the time consumed in waiting for the acknowledgements, the time reduced is usually 3 times the time consumed waiting for acknowledgements.

In pipelined protocol, the following consequences occur.

1. Increase in the count of sequence numbers as the number of packets and acknowledgments need unique sequence numbers.
2. Buffering of multiple packets possible at both sender and receiver side. On sender side, buffering is performed with respect to unacknowledged packets and on receiver side, buffering is performed with respect to corrupted packets.
3. The above requirements depend on the approach followed in response to the lost, corrupted or delayed packets. Go-back-N and selective repeat are two approaches which are based on pipelined protocol.

Go-Back-N

For answer refer Unit-IV, Q22.

Selective Repeat

For answer refer Unit-IV, Q23.

4.4.3 Go-Back-N (GBN), Selective Repeat (SR)

Q22. With an example, explain Go-Back-N protocol of transport layer.

Answer :

Go-Back-N Protocol (GBN)

This protocol is used to send multiple packets without even receiving the acknowledgments. However, the receiver is capable of buffering only one packet. A copy of sent packets must be maintained until acknowledgements are received. Using this protocol, multiple data packets and acknowledgments can be allowed to be in same channel at same time. The following figure shows the transmission of data packets using Go-Back-N Protocol.

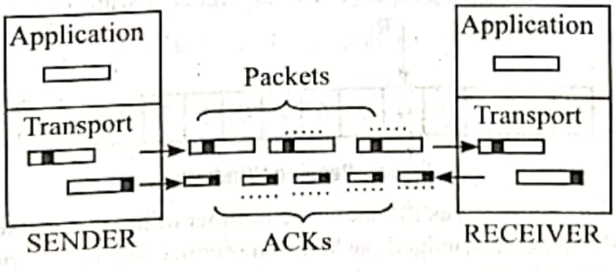


Figure: Go-Back-N Protocol

The sequence numbers in this protocol are modulo 2^m. Here, m gives the sequence number field size in bits the acknowledgement number is cumulative. It gives the sequence number of the next arriving packet of the acknowledgement number is 6 then the packets upto 5 are said to have arrived and the receiver waits for packet of sequence number 6.

Go-Back-N has two windows namely, send window and receive window.

Send Window

Send window is enclosed by the sequence numbers of the ongoing packets or that are to be sent. The size of this window is $2^m - 1$. It divides the sequence numbers into four regions as follows,

1. The first left most region contains sequence numbers of acknowledged packets.
2. The second region contain the sequence numbers of sent and unacknowledged packets. These packets are known as outstanding packets.
3. The third region contains the sequence numbers of packets that are to be sent. However the data to be sent is not get received.
4. The fourth region contains the sequence numbers that cannot be used till the window slides.

The send window defines three variables S_f (send first), S_n(send next) and S_s(send window size). The variable S_F gives the sequence number of just outstanding packet, S_n gives the sequence number of next packet to be sent and S_S gives the size of window. The following figure shows the send window in Go-Back-N Protocol.

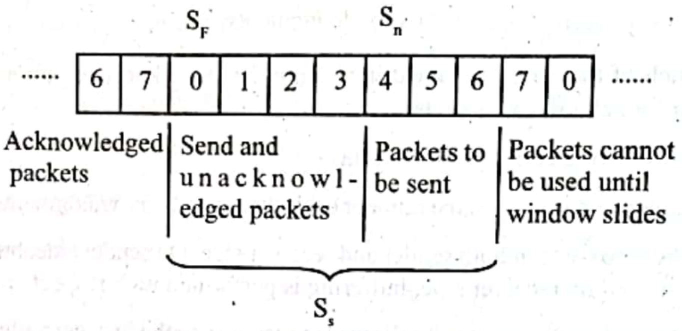


Figure: Send Window

The send window can slide to more than one slots on the arrival of acknowledgment. The acknowledgment number must be greater than or equal to S_F and less than S_n. The following figure shows the sliding of window.

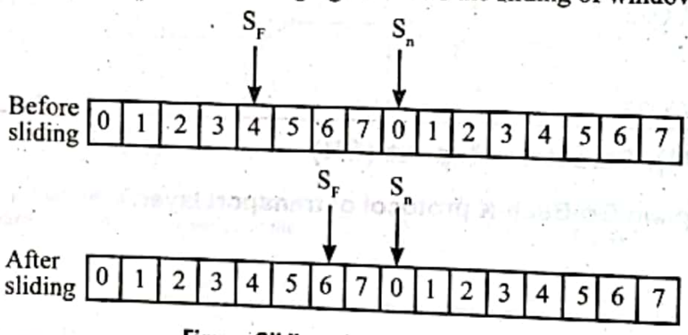


Figure: Sliding of Send window

Receive Window

Receive window ensures correct arrival of packets and correct sending of acknowledgements. It discards packet that arrives out of order and resends it with size 1. The following figure shows the receive window,

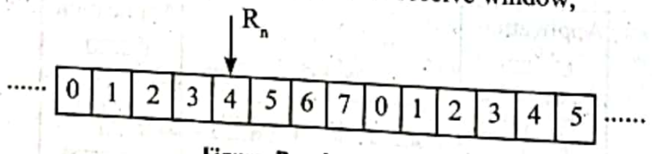


Figure: Receive Window

Receive window has a variable R_n which gives the sequence number of next packet to be received. The sequence numbers present on the left side are of received and acknowledged packets. The sequence numbers present on the right side are of packets which cannot be received. A packet whose sequence number matches with R_n is only taken and acknowledged. If a received packet with a sequence number in any of the two regions is found, then it is discarded when a correct packet is received then the window slides by only one slot at any instant.

Timers in Go-Back-N

Go-Back-N protocol uses only one timer. Here, the timer for the first outstanding packet expires first. Then all the outstanding packets are resent. Hence, the protocol Go-Back-N is named so.

Finite State Machines (FSM)

Sender FSM

Initially the sender is in ready state. Later it can be in either ready or blocking states. The variables S_F and S_n are initialized to 0 when the sender is in ready state, then any of the following four events are likely to happen.

1. If the application layer sends the request, then the sender creates an packet and sets the sequence number to S_n. It stores a copy of packet and sends it. It also starts the timer if it is not running. After this, S_n is incremenated and $(S_n + 1)$ module 2^m is obtained. If the window becomes full then the sender moves into the blocking state.
2. If an error-free ACK which matches with any of the outstanding packets arrive then the send window slides of all the outstanding packets are acknowledged then the times is stopped otherwise it is restarted (i.e., all outstanding packets are not acknowledged)
3. If an ACK whose Acknowledgement number (AckNo) is not related with the outstanding packet arrives then it is discarded.
4. If there is a time-out then all the outstanding packets are resend and the timer is restarted when the sender is in blocking state then the following three events are likely to occur,

   (a) If an acknowledgement whose acknowledgement number (AckNo) related to any of the outstanding packets arrive, then the window is slided. If all the outstanding packets gets acknowledged then the timer is stopped otherwise the timer is restarted if some of the outstanding packets are left unacknowledged. Then the sender goes to ready state.

   (b) If an ACK who Acknowledgment number is not related with the outstanding packet which is arrived then it is discarded.

   (c) If there is a time-out then all the outstanding packets are resend and the timer is restarted.

The following figure shows the above events,

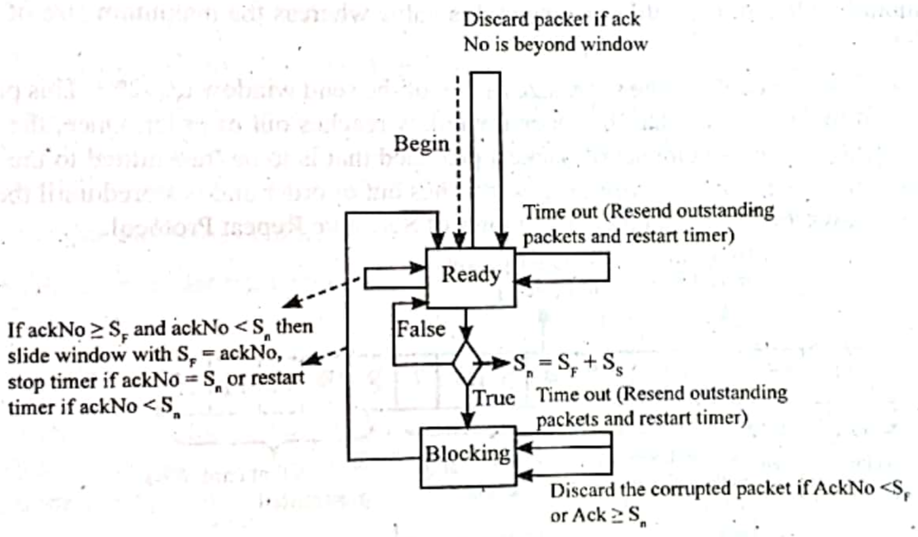

Discard packet if ack No is beyond window
Begin
Time out (Resend outstanding packets and restart timer)
Ready
False
If ackNo ≥ S_F and ackNo < S_n then slide window with S_F = ackNo, stop timer if ackNo = S_n or restart timer if ackNo < S_n
S_n = S_F + S_S
True
Time out (Resend outstanding packets and restart timer)
Blocking
Discard the corrupted packet if AckNo <S_F or Ack ≥ S_n

Figure: Sender FSM

Receiver FSM

The Receiver will always be in the ready state. It's variable R_n is initialized to 0. The following three events are likely to occur in this state,

1. If a packet with its sequence number R_n arrives then the message in it is sent to the application layer. Then the window slides with $R_n = (R_n + 1)$ modulo 2^m and an Ack is sent with acknowledgment number equivalent to R_n.
2. If a packet with seqNo beyond the window arrives then it is discarded. However an Ack is sent with acknowledgment number R_n.
3. If a packet which is corrupted arrives, then it is discarded.

The following figure shows these three events.

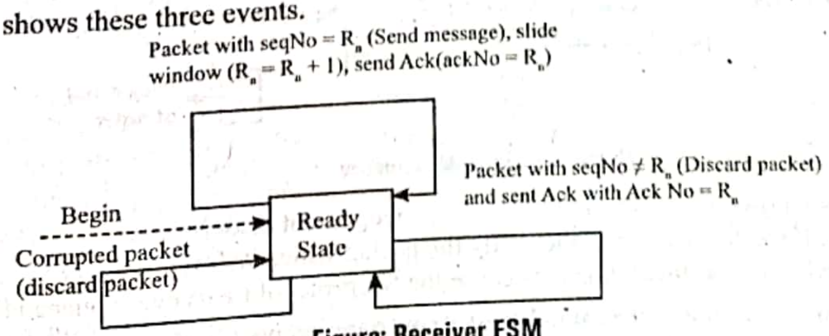

Packet with seqNo = R_n (Send message), slide window (R_n = R_n + 1), send Ack(ackNo = R_n)
Packet with seqNo ≠ R_n (Discard packet) and sent Ack with Ack No = R_n
Begin
Ready State
Corrupted packet (discard packet)

Figure: Receiver FSM

Q23. Explain the operation of selective repeat protocol of transport layer.

Answer :

Selective Repeat Protocol

The selective repeat protocol has been designed to overcome the drawbacks faced by the Go-Back-N protocol. As the name itself suggest, it resends only those selective packets that were lost during the transmission. The figure below shows the outline of selective-Repeat Protocol.

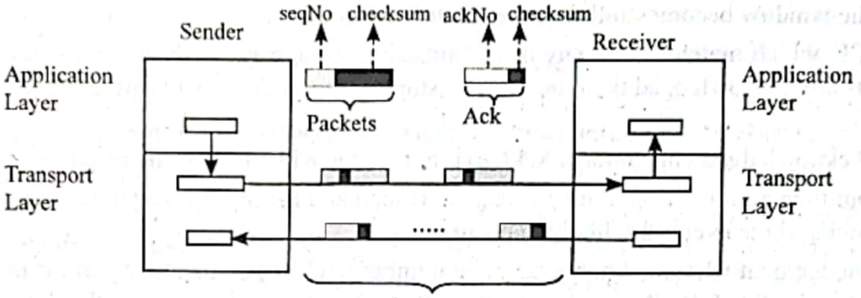


Figure (1): Selective-Repeat Protocol

As shown in the above figure, this protocol also makes use of two windows namely a send window and a receive window. But the tasks performed by these windows are different from the tasks performed by the windows of GBN (Go-Back-N) protocol. In SR (Selective Repeat) protocol, the size of the send window can be 2^{m-1}. It must be noted that the maximum size of the send window of SR protocol should be 8 and it should not exceed this value whereas the maximum size of the send window in Go-Back-N protocol should be 15.

The receive window of SR protocol has the same size as that of the send window i.e., 2^{m-1}. This protocol allows maximum number of packets to transmit in such a way that the receive window reaches out of order. Once, the receive window reaches out of order, it is kept idle until there is another set of packets provided that is to be transmitted to the application layer. Since, the size of both the windows are same, the send window also reaches out of order and is stored until the other set of packets are delivered. The figure below shows the send and receive windows of Selective Repeat Protocol.

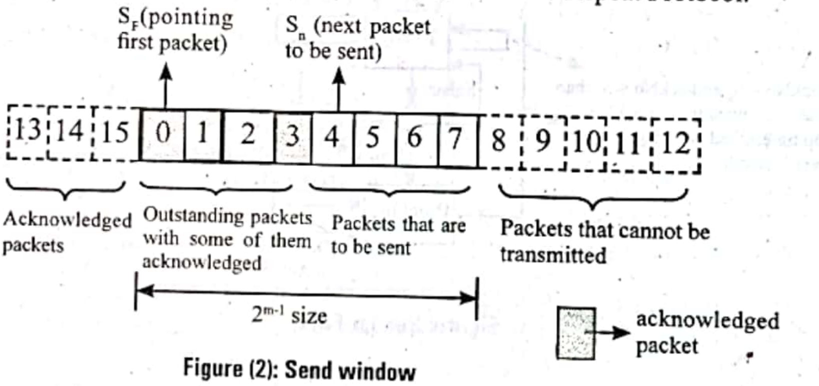

S_F(pointing first packet)
S_n (next packet to be sent)
13
14
15
0
1
2
3
4
5
6
7
8
9
10
11
12
Acknowledged packets
Outstanding packets with some of them acknowledged
Packets that are to be sent
Packets that cannot be transmitted
2^{m-1} size
acknowledged packet

Figure (2): Send window

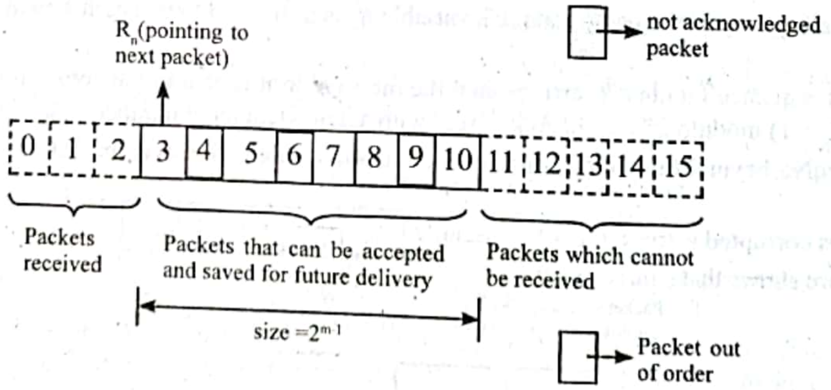

R_n(pointing to next packet)
not acknowledged packet
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Packets received
Packets that can be accepted and saved for future delivery
Packets which cannot be received
size $=2^{m-1}$
Packet out of order

Figure (3): Receiver window

Apart from these two windows, there is a timer and acknowledgement associated with this protocol. The timer here is used for every out standing packet. If the timer expires, then only the packet which was suppose to reach the receiver is resent. There is only one timer associated with this protocol, this is because the SR protocol treats every outstanding packet as an individual.

The AckNo here specifies the consequtive number of an outstanding packet, that is received in a safe made.

FSMs

The FSMs of SR protocol is similar to the FSMs of the GBN protocol, but has some minor differences.

Sender

Initially the sender is in ready state, later on it can be found in any of the two states i.e., either ready or blocking state.

Ready state

When the sender is in ready state, the following four events can occur,

1. If the application layer requests for some packets, then the sender sends a packet whose sequence number is "S_n". A copy of this transmitted packet is stored. The sender checks the timer if it is not running, then the sender sets the timer. The value of the transmitted packet 'S_n' is incriminated i.e., $S_n = (S_n + 1)$ modulo 2^m. If the window is found to be full, then the sender shifts to the blocking state i.e., $S_n = (S_f + S_{size})$.
2. If an unambiguous Ack reaches with Ack-No corresponding to any of the outstanding packets, then the packet is counted as acknowledged. If the Ack-No = S_f is determined, then the window is shifted to the right until the 'S_f' addresses to the first acknowledged packet. If there are many out standing packets, then the timer is reset else the timer is ended.
3. If a corrupted packet is received and the Ack-No is not matching with any of the outstanding packets, then the received packet is discarded.
4. If the time expires, then the sender retransmits all the unacknowledged data packets present in the window and resets the timer.

Blocking State

There are three possible events that can occur. They are as follows,

1. If an unambiguous Ack arrives whose Ack-No is corresponding to any of the outstanding packets, then that packet is counted as acknowledged. If the Ack-No = S_f is determined, then the window is shifted to the right until the S_f addresses to the first unacknowledged packet. If the window is shifted, then the sender is moved to the ready state.
2. If a corrupted Ack whose Ack-No is not matching to the outstanding packets, then that Ack is discarded.
3. If the time expires, then the sender retransmits all the unacknowledged packets present in the window and resets the timer.

Receiver

The receiver is always found in the ready state. The following are the three events that many occur.

1. If an unambiguous packet with the se No reaches, then the received packet is stored and the Ack whose AckNo = seq No is sent. If the seqNo = R_n is determined, then the current packet and the previously received packets are forwarded to application layer and the window shifts in such away that the R_n points to the starting empty slot.
2. If an unambiguous packet whose seqNo is outside the window arrives then the arrived packet is discarded but an Ack whose AckNo = R_n is sent to the sender. This is because, it allows the sender to shift its window if any Acks corresponding to the packets whose seq No < R_n were missing.
3. If any corrupted packet reaches, then that packet is discarded.

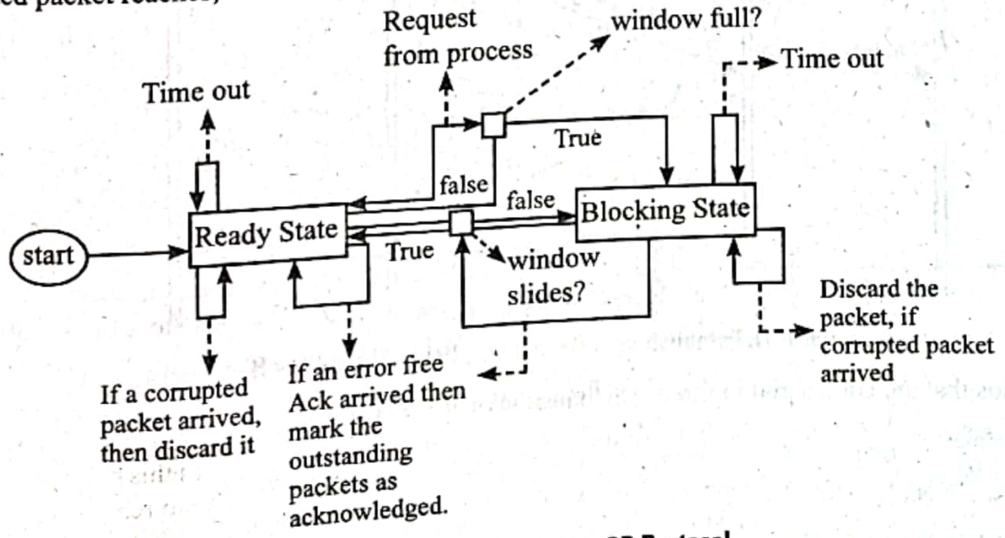

Request from process
window full?
Time out
Time out
True
false
false
Ready State
Blocking State
start
True
window slides?
Discard the packet, if corrupted packet arrived
If a corrupted packet arrived, then discard it
If an error free Ack arrived then mark the outstanding packets as acknowledged.

Figure (4): Sender's FSM for SR Protocol

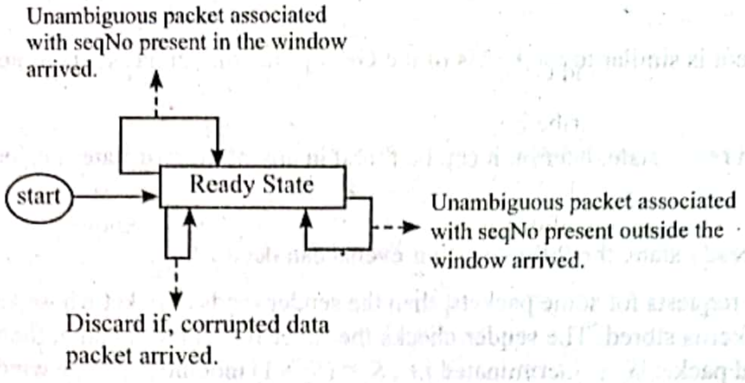

Unambiguous packet associated with seqNo present in the window arrived.
start
Ready State
Unambiguous packet associated with seqNo present outside the window arrived.
Discard if, corrupted data packet arrived.

Figure (5): Receiver's FSM for SR Protocol

4.5 CONNECTION ORIENTED TRANSPORT :TCP

4.5.1 The TCP Connection, TCP Segment Structure

Q24. Explain the three way handshaking protocol to establish the transport level connection.

Answer : **Model Paper-II, Q8(b)**

Three-way Handshake

The type of handshake used by TCP is called three-way handshake because three segments are exchanged. The TCP's three-way handshake is the process for establishing a TCP connection. This process starts by server i.e., server a request that it is ready to accept a connection. Such request is known as passive open. When client needs a connection, it sends a request to a particular server. Such request is known as active open. A TCP connection is established as shown in example below. In this example, Assume that a client computer is contacting a server to send some information.

1. The client sends a packet with the SYN bit set and a seq number x.
2. Server sends a packet with an ACK + SYN bit set.
3. The client sends a packet with an ACK (connection is established).
4. Client sends the data.

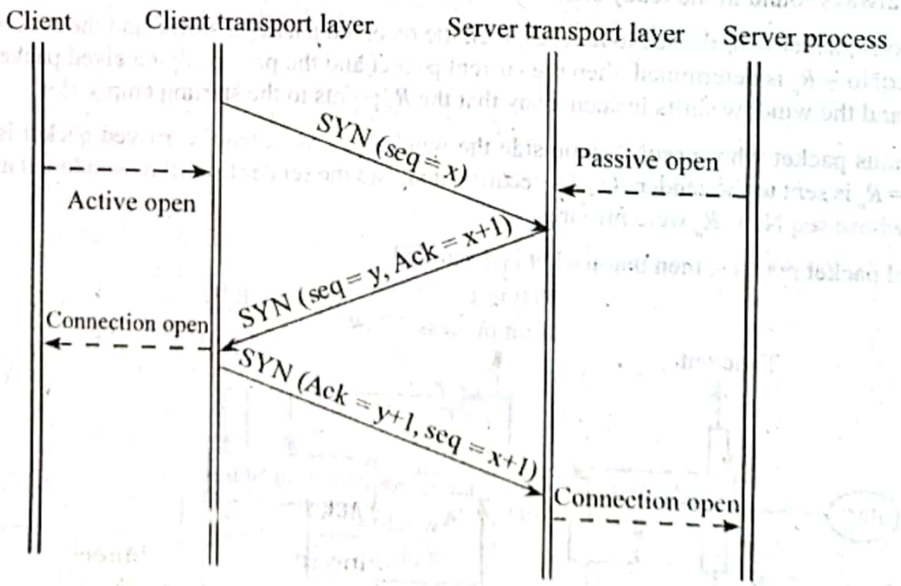


Figure (1): Establishing a Connection Using Three-Way Handshake

Three scenarios that are considered in three-way handshake are as follows,

1. Normal case
2. Duplicate SYN segment
3. Duplicate SYN and ACK segment.

1. **Normal Case**

In this case client starts the process and establishes the connection.

Steps and figure are same as that described in previous example.

2. **Duplicate SYN Segment**

In this case, server receives a TPDU without the client's knowledge. Server acknowledges this TPDU. When ACK TPDU is received by client it rejects to establish a connection by sending REJECT TPDU. Server thinks that it is a duplicate SYN TPDU and releases the connection.

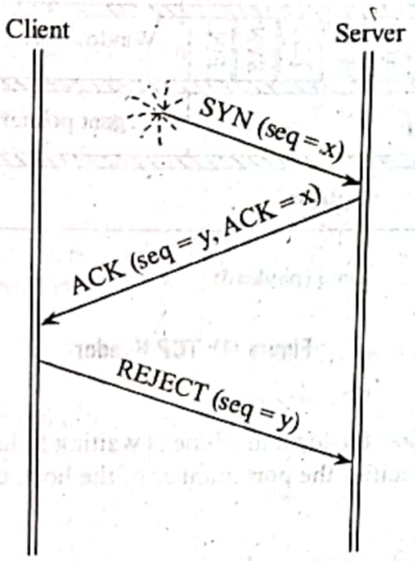


Figure (2): Duplicate SYN Segment

3. **Duplicate SYN and ACK Segment**

In this case, server receives a TPDU, that is, requesting a connection server which acknowledges it by sending the ACK TPDU. Client sends the REJECT TPDU as soon as it receives the ACK. In meantime before receiving a REJECT TPDU, server receives an ACK TPDU, then it sees that the ACK is sent for some other TPDU but not the one it was expecting. It will release the connection thinking that it is an old duplicate packet.

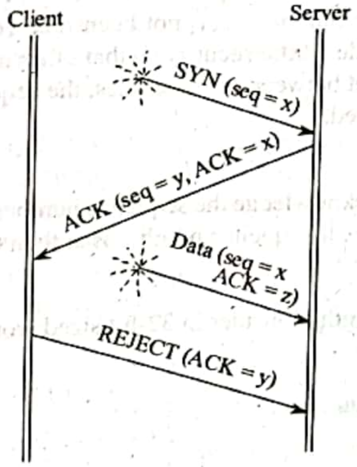


Figure (3): Duplicate SYN and ACK Segment

Q25. Explain various fields of the TCP header and the working of the TCP protocol.

Answer :

TCP Header

TCP segments are present within the Internet datagram. The Internet protocol header carries several information fields including the source and the destination host address. A TCP header follows the Internet header supplying information specific to the TCP protocol. The IP header is of 20 bytes and TCP header is of 20 bytes. The maximum amount of payload that can be transmitted is followed after the header is 65535 – 20 – 20 = 65495 bytes. That is 65495 data bytes may follow after the option.

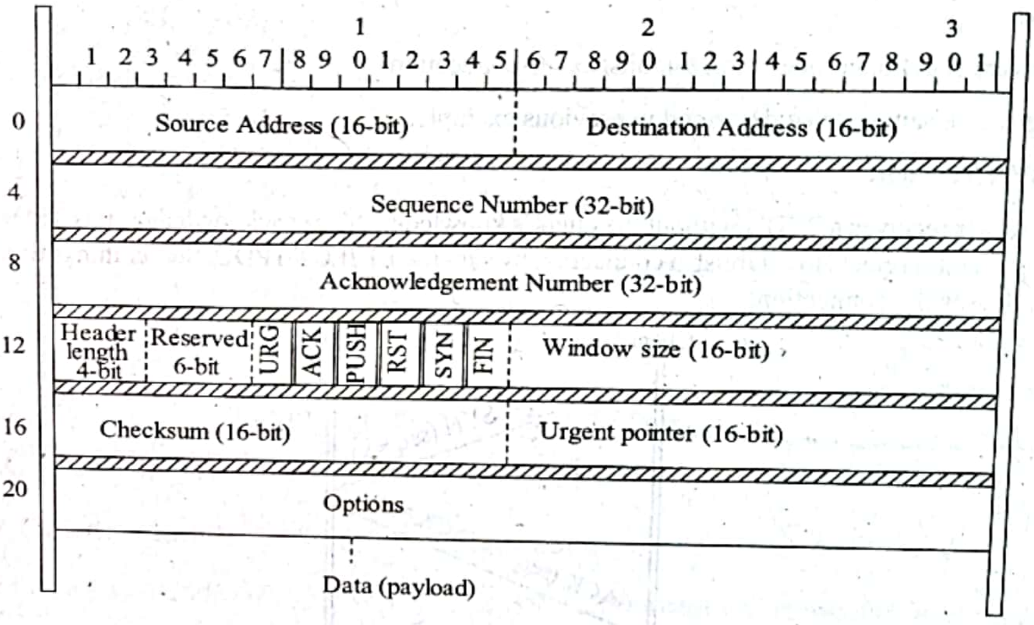


Figure (1): TCP Header

1. **Source Address**

It is a 16-bit field that specifies what port the local machine is waiting to listen for responses from the destination machine when a connection is attempted. It specifies the port number of the host, that is sending the segment.

2. **Destination Address**

It is also a 16-bit field that specifies what port number is going to be used by the local machine as a packet is being sent out to its destination. It specifies the port number of the host, that is receiving the segment.

3. **Sequence Number**

It is a 32-bit field, that is used to help the destination to know which packet to acknowledge. To ensure the connectivity, each byte that is being transmitted should be numbered. In sliding window protocol like TCP, the sequence number allows both TCP stacks to know which packet has been received and which one have not been received.

For example, If the sender sends the packet with sequence numbers 1, 2, 3, 5, 6, 7, 8, 9, 10. Then the receiver can easily recognize that the packet with sequence number 4 has not been received and then it request the sender to retransmit a copy of packet 4. It is also used to provide a little security so that other users cannot easily break during the middle of the connection. Once a communication is set between the machines, the sequence number increases are directly proportional to the number of bytes that are transmitted.

4. **Acknowledgment Number**

It is also a 32-bit field which is used to acknowledge the sequence number sent by sender and then specifies the next packet expected not the last received packet i.e., if sequence number is x, then the receiver acknowledges it with $x + 1$.

5. **Header Length**

It is a 4-bit field which indicates the length of header in 32-bit sized words.

6. **Reserved Field**

It is a 6-bit field, that is reserved for future use.

7. **Control Fields**

It is a 6-bit field that controls the fields. The various control fields are as follows,

(a) **URG FLAG**

This 1-bit field specifies that largest pointer included in the packet is valid i.e., the bit is set to 1.

(b) **ACK FLAG**

This field specifies that the portion of the header with the valid acknowledgment number is valid i.e., the bit is set to 1.

(c) **PUSH FLAG**

This tells the TCP/IP stack that the data packet received should be pushed up to the application layer as soon as it is received

(d) **RST FLAG (Reset)**

It is used to reset a connection that has been terminated due to a hardware crash.

(e) **SYN FLAG**

This flag is set when a connection is to be established. It is used to synchronize sequence number with the acknowledgment number for both hosts. When sender is transmitting a request for connection it sends SYN = 1, ACK = 0, then the receiver reply by sending an acknowledgment with SYN = 1, ACK = 1. In short, SYN is used for requesting and accepting a connection.

(f) **FIN FLAG**

This is used to specify that a connection is finished according to the side that sent the packet with FIN flag set.

8. **Window Size**

It is a 16-bit field which specifies how many bytes may be received by the receiving side before being halted from sliding any further. Since it is a 16-bit field, the maximum size of window is 65,535 ($2^{16} - 1$). The size of a window can be zero which specifies that the receiver wants sender to wait until it updates the window size.

9. **Checksum**

Checksum covers the header and data portion of a TCP packet to allow the receiving host to verify the integrity of an incoming TCP packet. The checksum is done by adding all the 16-bit words using 1's complement contained in the header and text. If a segment contains an odd number of headers and text octet that are to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purpose.

The checksum also covers a 96-bit pseudoheader conceptually prefixed to the TCP header. This pseudo-header contains the source address, destination address, protocol and the TCP length fields. This gives TCP protection against misrouted segments. This information is carried in the internet protocol and is transferred across the network interface.

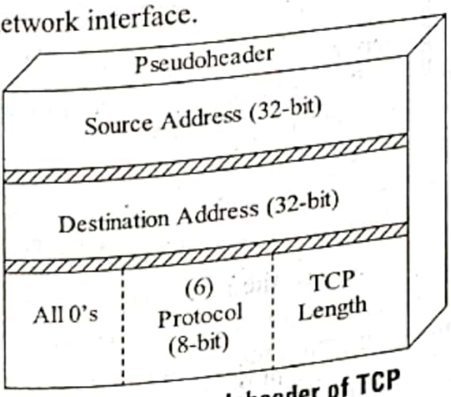


Figure (2): Pseudoheader of TCP

TCP length is the TCP header length plus the data length in octet. The value for the protocol field is 6. sources, address and destination address are 32-bit fields.

10. **Urgent Pointer**

This allows for a section of data that is specified by the urgent pointer to be passed by the receiving host quickly. It points to the sequence number of the octet following the urgent data. This field is only interpreted in the packet if the URG control bit is set.

11. **Option**

It is of variable length. Options may occupy space at the end of the TCP header and are multiples of 8-bits in length. It is generally used to provide more information that was not covered in the TCP header.

12. **Data**

It is of variable length. This is the payload or the data portion of the TCP packet.

4.5.2 Round-trip Time Estimation and Timeout

Q26. With an example, explain Round-trip time estimation in TCP.

Model Paper-III, Q8(b)

Answer :

Round-trip Time Estimation

When a packet is transmitted from source to destination, then the destination after receiving the packet should forward an acknowledgment to the source. And if it does not reply the source should wait for reply for some specific time and retransmit the lost packet.

Here, a question arises i.e., how long the source should wait for the reply?

To answer this question, TCP provides an estimate exchange time by monitoring normal exchange of packets. This estimation is known as "Round Trip Time Estimation" which turned out to be an important parameter for the better performance of TCP exchange as it avoids dropping and retransmitting of packets in large data transfers. When the round trip time is low, TCP retransmits the packets unnecessarily while the source waits for the time out in case of high estimation time.

In case of MANETs in which devices are connected wirelessly , when a mobile device is moved from one link to another, RTT considers that links as broken.

Example

Establish a TCP connection, send some data and disconnect the connection by removing the cable and send some more data after disconnecting the cable. After a long time, TCP will display the result as "Connection closed by foreign host".

Welcome (Normal message transfer)

Hello (Send this message after disconnection)

Connection closed by foreign host (TCP result after 10 minutes).

Urgent Data

TCP refers to stream oriented protocol. This indicates that data transmission between application program and TCP occurs in the form of stream of bytes. Each byte of data in a stream of bytes has its own position. In some situations, the sending application program wants to send some urgent bytes which have to be read out of order by the receiving application program. For example, the sending application program sending a huge amount of data to be processed by the receiving application program. When the result of processing comes back, the sending application program determines that everything is wrong. Now sender needs to abort the process. If it issues an abort command, using (Control + C) then these two characters will be at the end of the receiving TCP buffer, which will be delivered to the receiving application program after the completion of data processing. The TCP handles this situation bye sending a segment with **URG** bit set. This means that the sending application program informs the sending TCP that the data being sent is urgent. Now, a segment is created by the sending TCP with urgent data at the beginning and remaining of the segment include normal data. In the header, urgent pointer field defines the end of the urgent data and the start of the normal data. The receiving TCP extracts. The urgent data with the help of urgent pointer and delivers it out of order to the receiving application program.

4.5.4 Flow Control

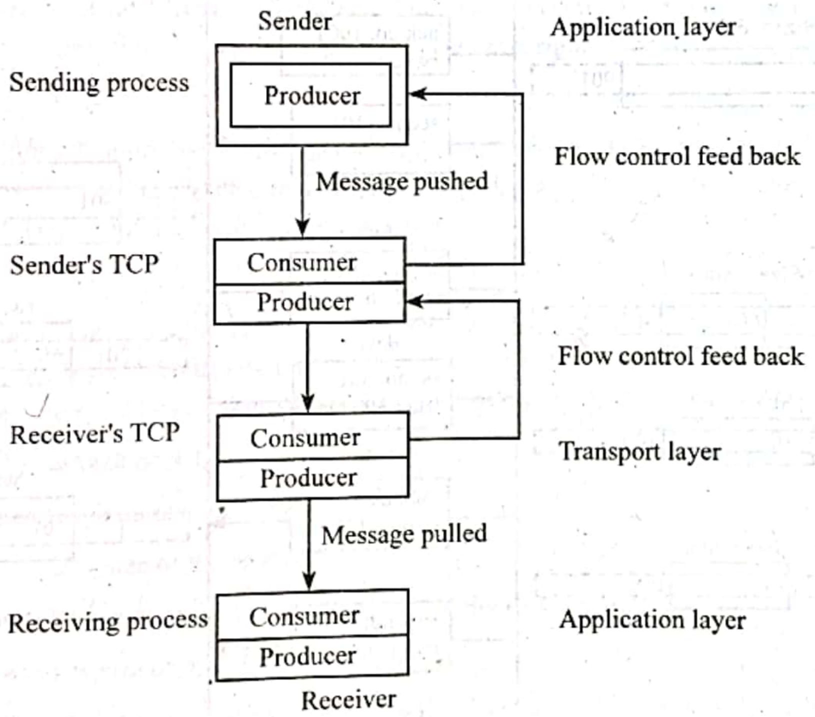
Q28. Explain the flow control mechanism of Transport Layer.

Answer : **Model Paper-II, Q9**

Flow Control

In TCP, flow control is the process to balance the flow of data produced by a sender and consumed by a receiver. However, flow control process is separate from error control process in TCP.

Consider the below figure representing data flow (unidirectional) from a sender to the receiver.



In the above figure, a unidirectional flow is represented from sender to the receiver. The sending process sends the messages initially to the sender TCP, next from sender TCP to receiver TCO. Finally from receiver TCP to the receiving process.

To control this process of sending the messages, flow control feedback is sent from receiver TCP to the sender TCP and from sender TCP to the sending process.

However, controlling the flow from sender's TCP to the sending process can be carried out by simply rejecting the packets. But the flow control process is to be carried out at the receiver's TCP to sending TCP.

Hence, to control the flow, TCP makes use of the adjust the window size of sender and receiver even if the size of the buffer is fixed at the time of connection establishment.

Window

To control the process of data flow, TCP makes use of window which is similar to the sliding window of data link layer. But here, TCP makes use of bytes instead of frames. Usually the size of the window will be fixed at the time of connection establishment, however according to the requirement the size is adjusted.

The process of opening, closing and shrinking of window is shown below.

Closing Window

If the window wall of receiver moves from left to right then window is said to be closing window. Window is closed as more number of byte are received from the sender.

Opening Window

If the window wall moves its right wall to the right then it is said to be opening window. Window is opened as more bytes are pulled at the receiver side.

Example

Consider a unidirectional flow of data from a client to server. Also consider that these flow is free from error. This can be shown as follows,

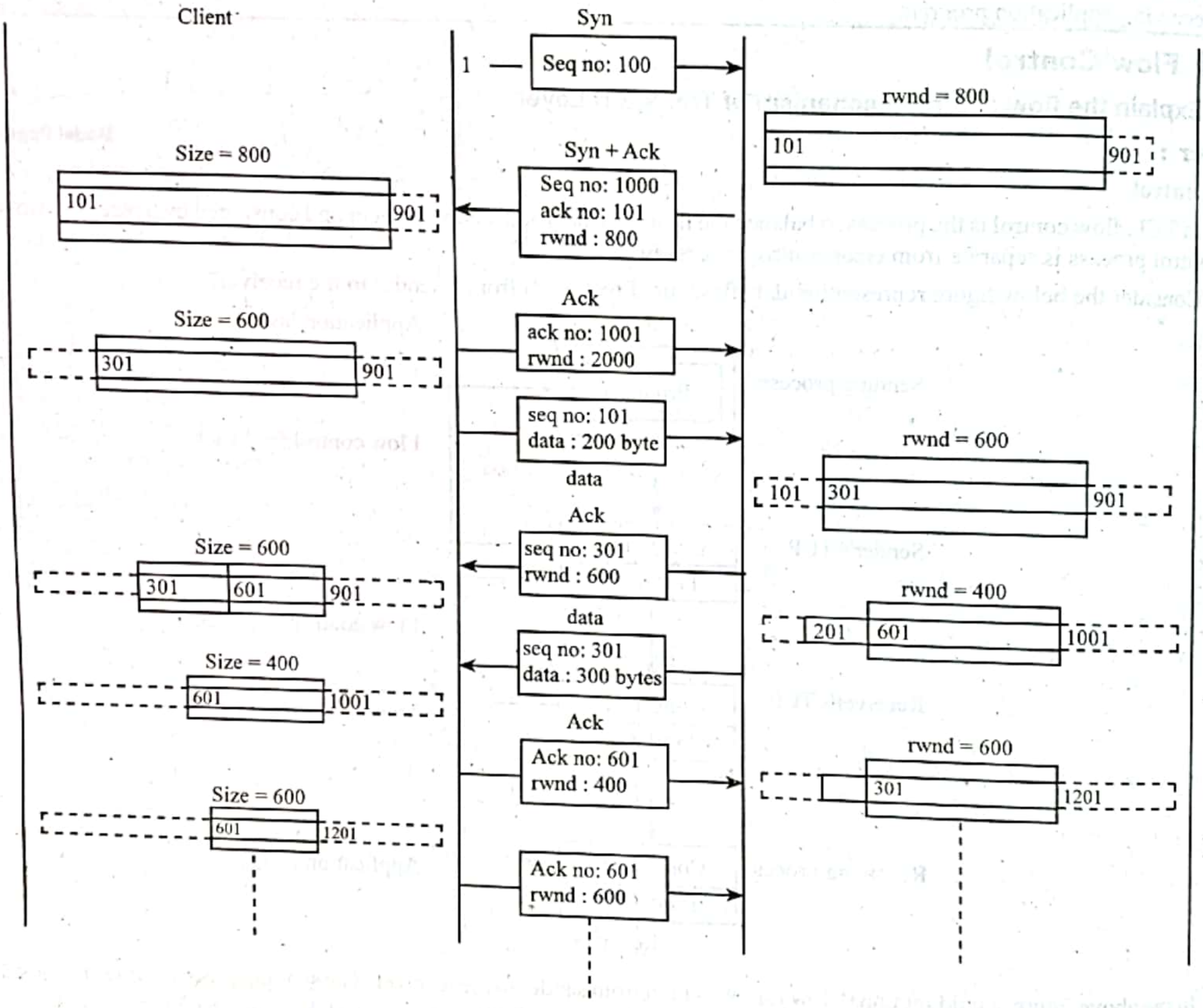


Figure: Scenario Which Depict Flow of Data From Sender to Receiver

The process of sending the message can be shown as follows,

Step 1

Initially, to establish a connection, client sends a request to the server i.e., (SYN). Also client announces that its initial sequence number (seq no) is 100.

Step 2

When the server receives the request, it allocates some window size, say it 800 i.e., rwnd = 800. Server acknowledge that its sequence number is 1000 and it expect data from 101 (i.e., rwnd = 800. Server acknowledge that its sequence number is 1000 and it expects data from 101 (i.e., SYN + ACK).

Step 3

Next, the client receives the acknowledgement and sets its window size to 800. Also it receives window size as 2000. It is ignored in the above examples was the flow considered here is unidirectional.

Step 4

Now actual process of sending the packets is carried out. Client's TCP sends the packet from 101 to 300. After sending the packets client adjust its window which was closed to show that the 200 bytes are sent and waits for an Ack. When server receives the packet window of receiver closes and sends Ack that it is expecting bytes from 301.

Step 5

After receiving the packets, the server stores the packets in the buffer and sends an Ack that window size is 600 (as it is already storing the 200 bytes of data).

Step 6

Next client sends 300 more byte with starting seq no 301. When server receives the packets it stores them. However, process pulls 100 bytes from the previously stored bytes. After this, the window of 100 bytes are free to use.

Thus, the window is now closed form left and opened from right so that 100 bytes are adjusted toward right. Now window size is set to 400 bytes.

Step 7

Now server sends an acknowledgement (Ack) slating that its window size is 400. When client receives this Ack, it shrinks it window size to 400 i.e., it closes the window from left and opens from right (i.e., move 100 bytes right).

Step 8

Thus, the server now contains 200 more bytes which are pushed from server by the process. Hence the window size of the receiver becomes 600. It's seq no is 601. When client receives this Ack it adjust it window size by opening its window by 200 and without closing it. Hence window size of client is 600.

Shrinking of Window

Shrinking of receiver window is not possible. However, the sender window can shrink if required. If the receiver set some value for receiver window size (rwnd) then sender's window can shrink. Even such implantations will not allow the sender's window shrinking. Hence to avoid the window shrinking the receiver should ensure the following relationship.

New Ack No + New rwnd $\geq$ last ack no + last rwnd

Here, New Ack no is acknowledge number of the new position.

New rwnd is the receiver window size of the new position.

Last Ack no is the Acknowledgements of the Previous position of the window wall.

Last rwnd is the receiver window size of the pervious position.

This represents that the window should not shrink i.e., right wall of window should not move to the left.

Example

Consider a scenario where the last Acknowledgement Number (last Ack No) is 206 and rwnd is 12. Let the bytes from 206 and 209 are consumed then New ACKNO is 210 and new rwnd is 4.

Then

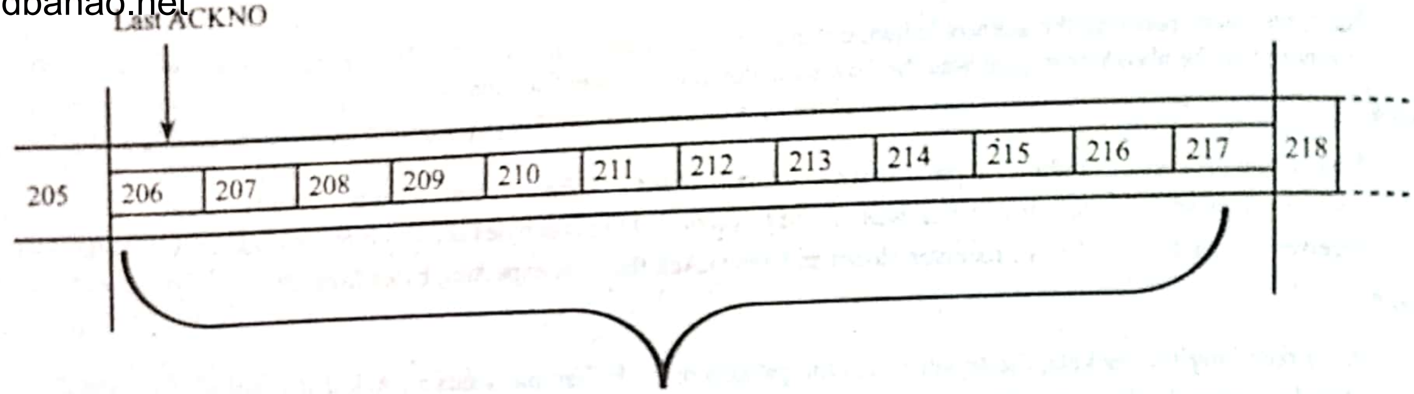
New Ack No + New rwnd $\geq$ last Ack No + last round

$210 + 4 \geq 206 + 12$

214 218

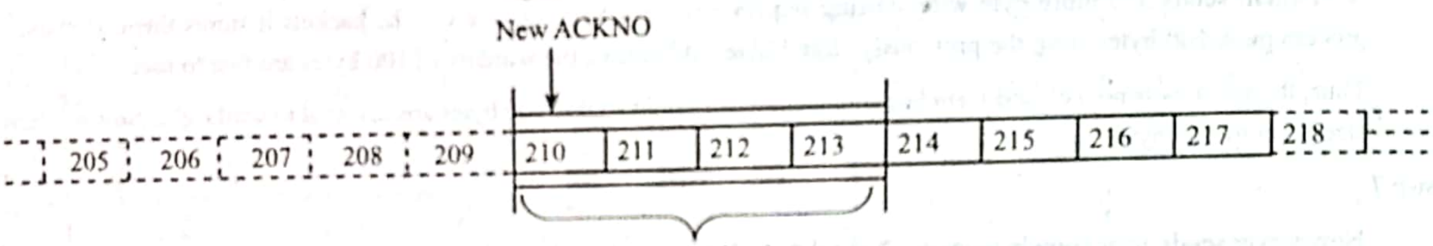
$214 < 218$

Hence this result in window shrink. This can be shown below,



Last rwnd = 45

Figure: Window of Last Advertisement



New rwnd = 4

Figure: Window After Advertisement

But it can be observed that after window gets shrunked, the byte 241 is out of window, which is already sent. Now the receiver has no knowledge of which byte is already sent from 210 to 217. Hence this creates a problem. To avoid such situation, receiver can delay the feed back till the time it has enough window buffer size.

Window Shutdown

To avoid the window shrinking, receiver can stop sending the packets for sometime. This can be done by sending an Ack as receiver window size as 0. This is known as window shotdown. Due to this, the server stop sending data till it receives a new advertisement. Even if the window is shutdown it can send 1 byte of data. This is known is probing. Probing is used to avoid the dead lock.

4.5.5 TCP Connection Management

Q29. Explain the three ways of connection termination in TCP using state transition diagram.

Answer :

TCP Connection Management

TCP uses a handshaking technique to open connection. It is referred to as three-way handshaking or as "SYN-SYN-ACK". This mechanism is designed so that two systems attempting to initiate a single connection for communication can negotiate one connection at a time independent of each other.

Connection Establishment

1. Server executes LISTEN and ACCEPT primitives and waits for the incoming connection request.
2. Client executes CONNECT by specifying the port number and IP address, maximum size of TCP segment that it can accept. CONNECT primitive is sent with SYN bit-ON and ACK bit-OFF.
3. Server finds out if any process has done a LISTEN, if not it will send RST bit for rejecting the connection. If any process is listening to port specified in destination address field, then server sends the response by setting the ACK bit on.

(a) Normal Way for Establishing a Connection

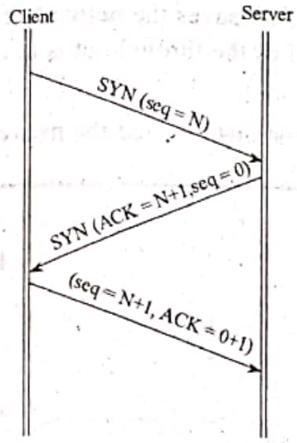


Figure (1): Establishing a Connection in Normal Way

Steps

1. Client sends a packet with the SYN bit set and a sequence number of N.
2. Server sends a packet with an acknowledgment number of $N + 1$, SYN bit set and sequence number of 0
3. Client sends a packet with ACK number $X + 1$. After this, the connection is established.

It is also possible for two hosts to simultaneously start an active open.

(b) Starting Connection between Two Hosts Simultaneously

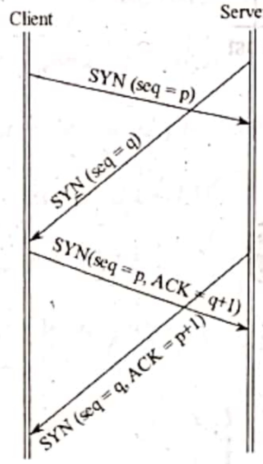


Figure (2): Two Hosts Connecting Simultaneously

An arbitrary initial sequence number is required to increment approximately for every 4 ms, this avoids delayed segments from a previous connection getting mixed up with the new connections.

(c) Connection Termination

Following are the steps, that are used for terminating a connection.

(i) Client sends a FIN bit to server

(ii) Server acknowledges the FIN bit

(iii) Server sends a FIN bit to client

(iv) Client acknowledges the FIN bit.

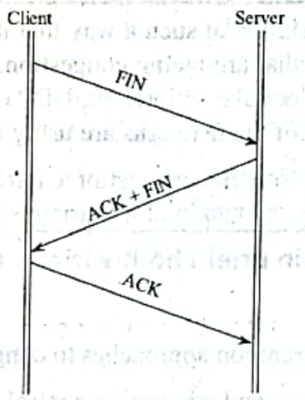


Figure (3): Terminating a Connection

Therefore, four TCP segments are used to release a connection one FIN, one Ack in each direction. FIN bit specifies that the host doesn't have any data to transmit, but it can still receive the data.

In some situation the Ack to FIN might be lost, then the host that is waiting for Ack in response to the closing FIN, will resend the FIN. If this arrives before the timer expires that is before twice maximum segment life, there is no problem, after this the FIN does not appear to belong to whatever connection might exist between the two hosts.

4.6 PRINCIPLES OF CONGESTION CONTROL – THE CAUSE AND THE COSTS OF CONGESTION, APPROACHES TO CONGESTION CONTROL

Q30. Discuss congestion control in traditional TCP.

Answer :

Congestion Control

Networks with fixed end-systems use a transport protocol that suits them. TCP (Transmission Control Protocol) is one such protocol. A packet is a piece of information that is transmitted over the network. This transmission requires both hardware copper wires, network adapters, special hardware for routers, fiber optics etc., and software. Hardware or software errors may not be the reason for packet loss in a fixed network. Ideally, the hardware does not introduce any errors in the network and if the software is of good quality, it will not drop the packet. Thus, the only reason could be a temporary overload at a node in the transmission path. Such a temporary overload is called "Congestion".

Routers are carefully designed networks. It can also act as congestion nodes.

Example

When numerous packets are transmitted to a single receiver and the speed with which these packets are transmitted is higher than the capacity of the receiver, the buffers in the router gets congested quickly and cannot accommodate more packets. However, till the sender realizes, the router starts dropping the packets. This leads to a gap in the packet sequence and the receiver acknowledges to all packets and stops acknowledging at the dropped one. Then, the sender understands that due to congestion a packet might be dropped and usually tries to quickly retransmit the dropped packet. However, doing so will not resolve congestion rather increase.

TCP is designed in such a way that it acts unusually in state of congestion–the transmission rate is reduced by all those TCP connections that are facing congestion. This reduces the speed of transmission, but saves the network from crashing, also makes TCP most popular in internet. UDP cannot overtake TCP as only at initial stages the throughput is in higher, which gets disappeared if all of the networks are using UDP.

Thus, it is true that congestion control effects the efficiency of TCP, but at least ensures that the network does not crash due to congestion and bandwidth sharing is carried out.

Q31. Discuss in brief about various approaches to congestion control.

Answer :

Model Paper-I, Q9(b)

The two common approaches to congestion control are,

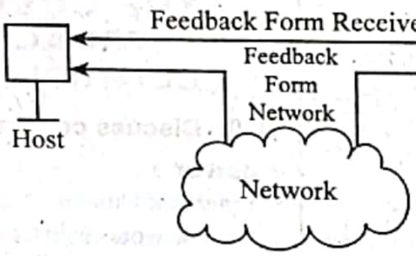
1. End-to-end congestion control
2. Network-assisted congestion control.

1. End-to-end Congestion Control

The network layer does not support congestion control in case of end-to-end approach. The responsibility of handling congestion is completely handed over to the end-systems which usually perform their work based on network performance. Moreover, there exist no approach of forwarding feedback to the systems regarding congestion occurring in the network. Therefore, it is necessary for TCP to adopt certain criteria for offering congestion control. In some cases, TCP segment loss can be considered as occurrence of congestion based on which TCP can reduce its window size.

2. Network-assisted Congestion Control

In this type of congestion control, senders can receive feedback from components residing at network layer. This feedback includes the information regarding the current state of network with respect to the congestion. The feedback is usually indicated with a single bit. Such an approach is adopted in IBM SNA and DEC DECnet architectures. Moreover, they are getting used in ATM ABR and XCP protocol.



IMPORTANT QUESTIONS

1. Discuss the services/responsibilities of transport layer. **Refer Q11**
2. Write notes on multiplexing and Demultiplexing. **Refer Q14**
3. Define UDP checksum and discuss the operation of UDP. **Refer Q18**
4. Describe the pipelined protocol for reliable data transfer. **Refer Q21**
5. Explain the three way handshaking protocol to establish the transport level connection. **Refer Q24**
6. Explain the flow control mechanism of Transport Layer. **Refer Q28**
7. Discuss in brief about various approaches to congestion control. **Refer Q31**