AUTUMN MID-SEMESTER EXAMINATION- 2019

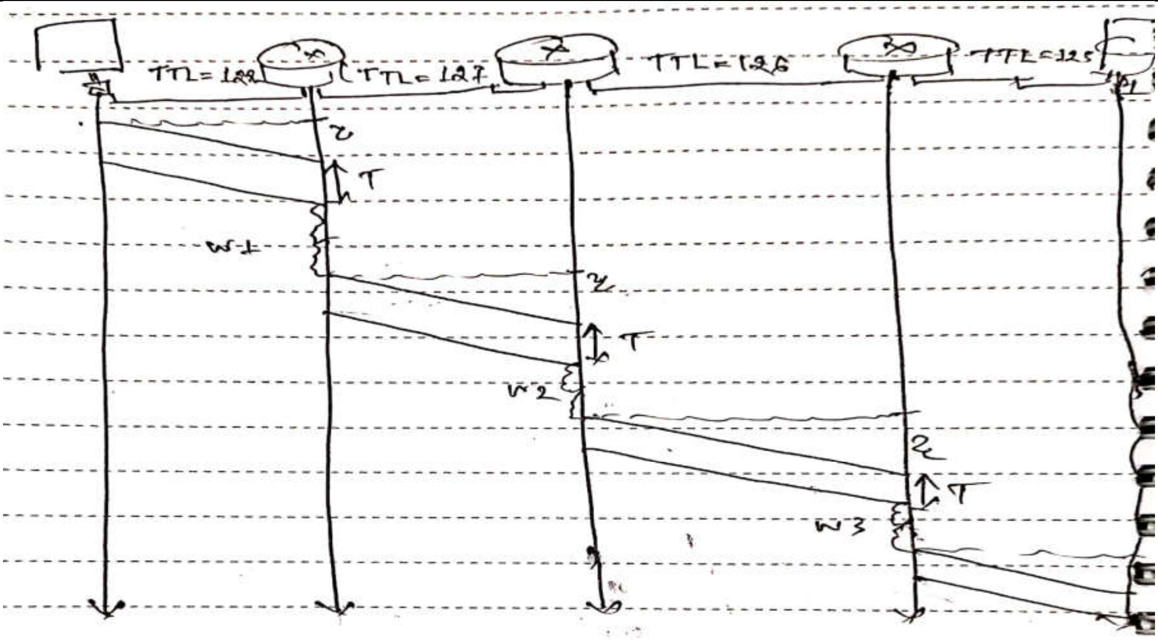
Computer Networks

[IT-3001]

<u>Solution & Scheme of Evaluation</u>

Q.No 1	Solution	Mark
a	**DNS Records:** Various types of records maintained by DNS (A, NS, MX, CNAME etc.). **A:** If Type=A, then Name is a hostname and Value is the IP address for the hostname. Thus, a Type A record provides the standard hostname-to-IP address mapping. As an example, (relay1.bar.foo.com, 145.37.93.126, A) is a Type A record. **NS:** If Type=NS, then Name is a domain (such as foo.com) and Value is the hostname of an authoritative DNS server that knows how to obtain the IP addresses for hosts in the domain. This record is used to route DNS queries further along in the query chain. As an example, (foo.com, dns.foo.com, NS) is a Type NS record. **MX:** If Type=MX, then Value is the canonical name of a mail server that has an alias hostname Name. As an example, (foo.com, mail.bar.foo.com, MX) is an MX record. MX records allow the hostnames of mail servers to have simple aliases. **CNAME:** If Type=CNAME, then Value is a canonical hostname for the alias hostname Name. This record can provide querying hosts the canonical name for a hostname. As an example, (foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.	**0.25 X 4 = 1** (0.25 for each DNS record)
b	**HTTP Cache:** The purpose of an HTTP cache is to store information received in response to requests for use in responding to future requests. This will overcome the loophole in making the stateless HTTP to make it more time dependent for HTTP transactions.	**1**
c	**FTP in-band & out-band Communication:** FTP uses two different parallel connections to transfer a file, that is why it is said to be 'out-of-band'. It uses a control connection and a data connection in parallel. Control connection is used to send information such as user identification, password, commands etc. Because of this control connection FTP is called "out-of-band" Port 20 for data and 21 for control connection.	**0.5 + 0.5 = 1**
d	**Go-Back-N Protocol:** Window Size=63, ACKNo. =13 means Packet 10, 11, 12 Acknowledged. So the sender will slide 3 position to right. Now, Sf=13, Sn=18, Rn=13.	**1**
e	<u>**TCP Header Length Field Calculation:**</u> HLEN=1101 = 13 (decimal) 13 X 4 = 52 bytes of Header | 52 – 20 = 32 bytes (Since 20 bytes is compulsory header) 32 Bytes of options are included in TCP segments. <u>**UDP Payload (data) Calculation:**</u> If 1101 = 13 would be total length of UDP | UDP header is fixed 8 bytes UDP Payload = UDP Total length – UDU header = 13 – 8 = 5 bytes 5 Bytes data UDP segment carries.	**0.5** **0.5**

Q.No 2	Solution	Mark
a	**(i) Number of intermediate router present = 3**	**0.5**



$$4T + 4\tau + w_1 + w_2 + w_3$$

Waiting time at each Router

= Queuing delay + Average Processing delay

$= (20 + 10)$ msec.

$= 30$ msec.

T = Transmission time of each packet

$$= \frac{\text{Size of packet}}{\text{Data rate}} = \frac{64 \text{ KB}}{100 \text{ mbps}}$$

$$= \frac{64 \times 8 \times 10^3}{100 \times 10^6} = 5.12 \text{ msec.}$$

τ = Propagation delay

$\because$ Speed = $\frac{\text{Distance}}{\text{time}}$

$\therefore$ Propagation delay = $\frac{\text{Distance between adjacent nodes}}{\text{Speed of Electrical Signal}}$

$$= \frac{100 \text{ km}}{\frac{2}{3} \times (3 \times 10^8) \text{ m/sec.}}$$

$$= \frac{100 \times 10^3 \text{ m}}{2 \times 10^8 \text{ m/sec.}}$$

$= 50 \times 10^{-5}$ sec.

$= 0.5$ msec.

Total time $= 4(5.12) + 4(0.5) + 30 + 30 + 30$

$= 20.48 + 2 + 90$

$= \boxed{112.48 \text{ msec.}}$ Ans

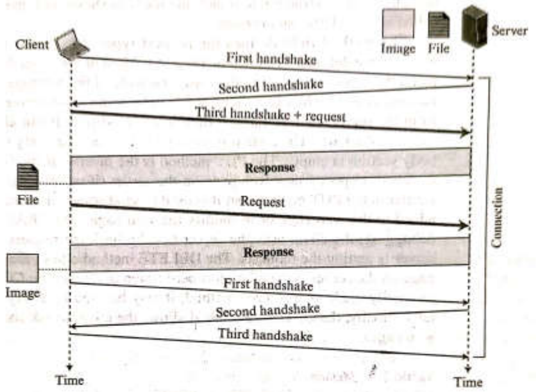
2

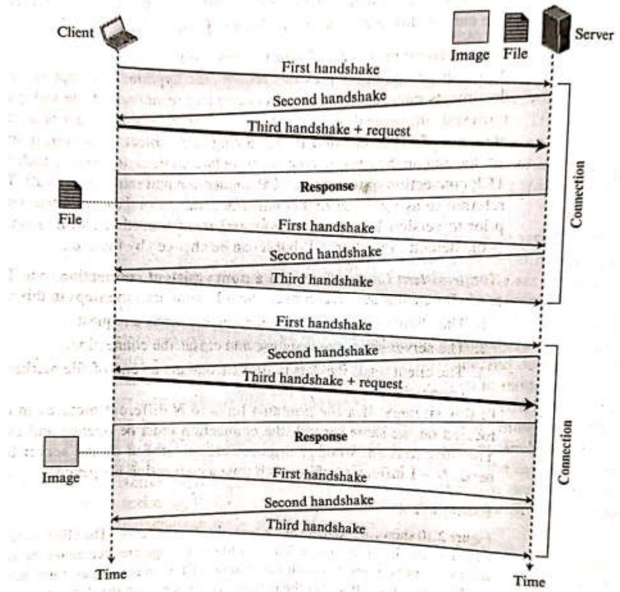
(ii) Total delay = 112.48 milliseconds

b **HTTP Persistent and Non-persistent**

Persistent	**Non-persistent**
1. TCP Connection (3 way handshaking) is made once and all subsequent HTTP Request/Reply are sent over single connection 2. The Connection remains open till all HTTP files and objects are sent	1. Each time a file/object is sent a separate TCP Connection is made (3 way handshaking for establishment & 3 way for connection termination) 2. Each time a new object is sent a new TCP Connection is done

1.5





Different methods: **GET, POST, PUT, HEAD, DELETE** etc.
GET: Request a document from server
POST: Send some information from client to server
PUT: Sends a document from the client to the server **1**
HEAD: Requests information about a document but not the document itself
DELETE: Removes the web page
Example: GET Method

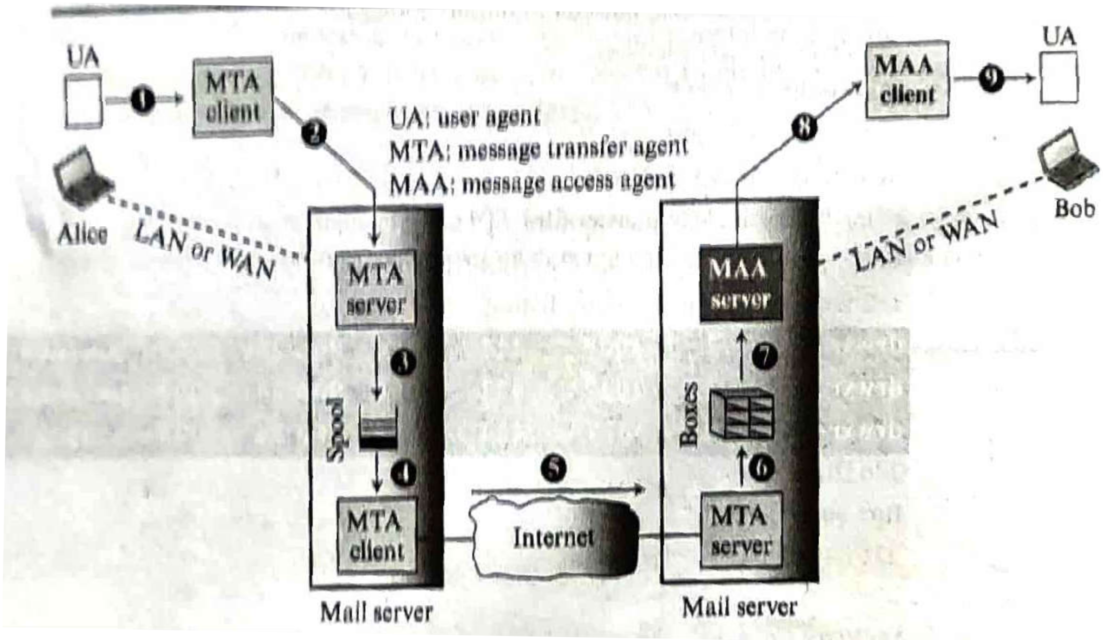
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Q.No	Solution	Mark
3		

a **PUSH and PULL protocol for e-mail communication**

- E-mail protocols SMTP a Asynchronous Protocol
- Each e-mail is an one-way communication **0.5**
- Lets assume Alice is Sender and Bob is receiver It is not necessary that Bob would reply to each and every E-mail sent by Alice all the instances.
- If any case if Bob sends a reply that would be another one-way communication
- E-mail works of **PUSH** (**MTA**) and **PULL** (**MAA**) Architecture

It is technically not feasible that receiver would always be **ON** to get a e-mail and even not possible to run a mail server so the mail-Sever is deployed in between with MTA and MAA agents (Programs) to implement a client/Server model on demand



1

E-mail communication steps:

1. UA of client sends the e-mail to a MTA client
2. MTA Client program of Sender sends mail to Mail Server (Sender end). MTA Server program gets the mail.
3. MTA Server puts the e-mail in a MessageQueue (Spool)
4. At Mail Server at sender end the e-mail is fetched from the Spool and using a MTA client program it is forwarded to the relevant MTA Server program running on the receiver mail server.
5. MTA Server at receiver mail server forwards the e-mail to the designated Mail-Box of the destination user. 1
6. Based on the user destination: **bob@hotmail.com** the e-mail is PUT in to the mail-box database.
7. MAA Server fetches data from relevant Mail-box
8. When the receiver comes online MAA Server program running on the received mail-server forwards the mail to MAA client
9. MAA client deliver the data to UA of the receiver

The electronic mail system needs two UAs, two pairs of MTAs (client and server) and a pair of MAAs (client and server)

b Distributing a File to fixed number of Peer in both **Client-Server Vs. P2P Architectures**

- Server and Peers are connected to Internet with access link

<u>**Nomenclatures:**</u>

***U_s*: Upload rate of Server's access link**

***U_i*: Upload rate of i^{th} peer's access link**

***d_i*: Download rate of i^{th} peer's access link**

***F*: Size of the File to be distributed (in bits)**

***N*: Number of Peers that wants a copy of the File**

- The **distribution time** is the time it
  takes to get a copy of the file to all *N* peers.
- Distribution time for both client-server and P2P architectures are calculated with some general assumptions.
- **Assumptions:**
  1. **There no bottleneck in Internet Core only bottleneck in the access network**

2. **Server and clients are not participating in any other network applications so that all of their upload and download access bandwidth can be fully devoted to distributing this file**

Client-server architecture, none of the peers aids in distributing the file.

1. The server must transmit one copy of the file to each of the *N* peers. Thus the server must transmit *NF* bits. Since the server's upload rate is *us*, the time to distribute the file must be at least *NF*/*u*
2. Let *d*min denote the download rate of the peer with the lowest download rate, that is, *d*min = min{*d*1,*d*p,...,*dN*}. The peer with the lowest download rate cannot obtain all *F* bits of the file in less than *F*/*d*min seconds. Thus the minimum distribution time is at least *F*/*d*min **1**

Putting two observations we get:

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}.$$

This provides a lower bound on the minimum distribution time for the client-server architecture. The server can schedule its transmissions so that the lower bound is actually achieved. So let's take this lower bound provided above as the actual distribution time, that is

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Peer-to-Peer (P2P) Architecture 'D_{P2P}'

- Similar analysis for the P2P architecture, where each peer can assist the server in distributing the file.
- When a peer receives some file data, it can use its own upload capacity to redistribute the data to other peers.
- Calculating the distribution time for the P2P architecture is somewhat more complicated than for the client-server architecture, since the distribution time depends on how each peer distributes portions of the file to the other peers.
- This can be simplified with some observations.

Observations **1**

1. At the beginning of the distribution, only the server has the file. To get this file into the community of peers, the server must send each bit of the file at least once into its access link. Thus, the minimum distribution time is at least F/u_s
(Unlike the client-server scheme, a bit sent once by the server may not have to be sent by the server again, as the peers may redistribute the bit among themselves.)
2. As with the client-server architecture, the peer with the lowest download rate cannot obtain all ***F*** bits of the file in less than ***F*/*d*min** seconds. Thus the minimum distribution time is at least ***F*/*d*min**.
3. Finally, observe that the total upload capacity of the system as a whole is equal to the upload rate of the server plus the upload rates of each of the individual peers, that is, ***u*total = *us* + *u*1 + … + *uN***. The system must deliver (upload) ***F*** bits to each of the ***N*** peers, thus delivering a total of ***NF*** bits. This cannot be done at a rate faster than *u*total. Thus, the minimum distribution time is also at least *NF*/(*us* + *u*1 + … + *uN*).
Putting these three observations together, we obtain the minimum distribution time for P2P, denoted by ***D*P2P**.

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

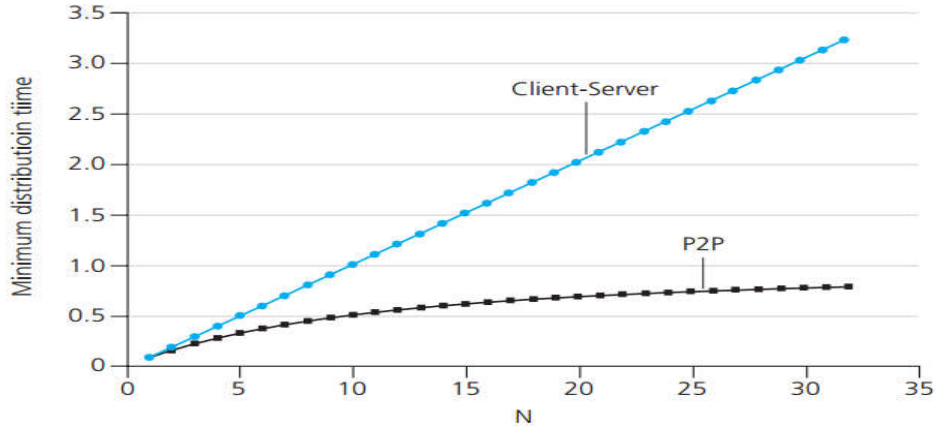
- Equation for '***D*P2P**'provides a lower bound for the minimum distribution time for

the P2P
architecture. It turns out that if we imagine that each peer can redistribute a bit as soon as it receives the bit, then there is a redistribution scheme that actually achieves this lower bound

In reality, where chunks of the file are redistributed rather than individual bits, Equation of ***D*P2P**serves as a good approximation of the actual minimum distribution time. Thus, let's take the lower bound provided by Equation of ***D*P2P**as the actual minimum distribution time, that is,

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

D_{CS} Vs. D_{P2P} Graphical Analysis



Distribution time for P2P and client-server architectures

0.5

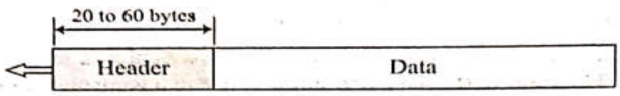
Q.No	Solution	Mark
4		
a	**UDP Checksum Calculation**	**2.5**

153.18.8.105			
171.2.14.10			
All 0s	17	15	
1087		13	
15		All 0s	
T	E	S	T
I	N	G	All 0s

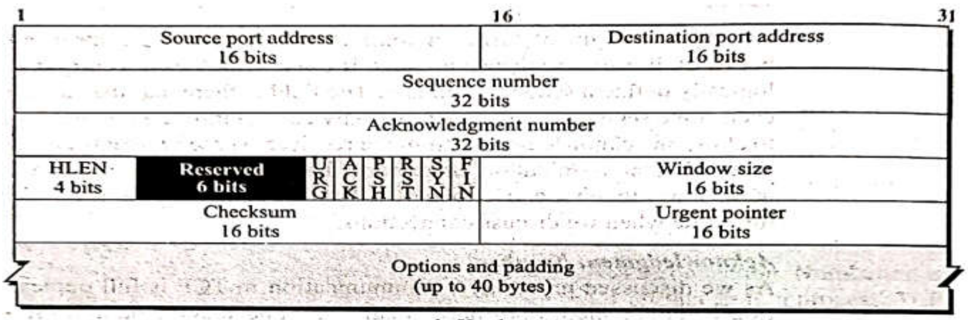
```
10011001 00010010 ---> 153.18
00001000 01101001 ---> 8.105
10101011 00000010 ---> 171.2
00001110 00001010 ---> 14.10
00000000 00010001 ---> 0 and 17
00000000 00001111 ---> 15
00000100 00111111 ---> 1087
00000000 00001101 ---> 13
00000000 00001111 ---> 15
00000000 00000000 ---> 0 (checksum)
01010100 01000101 ---> T and E
01010011 01010100 ---> S and T
01001001 01001110 ---> I and N
01000111 00000000 ---> G and 0 (padding)

10010110 11101011 ---> Sum
01101001 00010100 ---> Checksum
```

Students may do checksum calculation using HEX ()H /x16 for simplicity

b **TCP Segment**



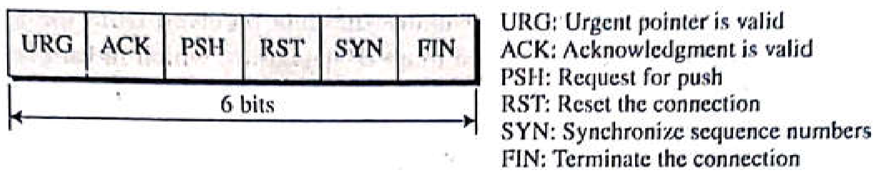
a. Segment 1



b. Header

- ***Source port address.*** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
- ***Destination port address.*** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
- ***Sequence number.*** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment (discussed later) each party uses a random number generator to create an **initial sequence number** (ISN), which is usually different in each direction. 1
- ***Acknowledgment number.*** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it returns $x + 1$ as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- ***Header length.*** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 ($5 \times 4 = 20$) and 15 ($15 \times 4 = 60$).
- *Control.* This field defines 6 different control bits or flags, as shown in Figure 3.45. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in the figure. We will discuss them further when we study the detailed operation of TCP later in the chapter.

Figure 3.45 *Control field*



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

- ***Checksum.*** This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the pro-
- ***Urgent pointer.*** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. This will be discussed later in this chapter.
- ***Options.*** There can be up to 40 bytes of optional information in the TCP header. We will discuss some of the options used in the TCP header later in the section.

Numerical:

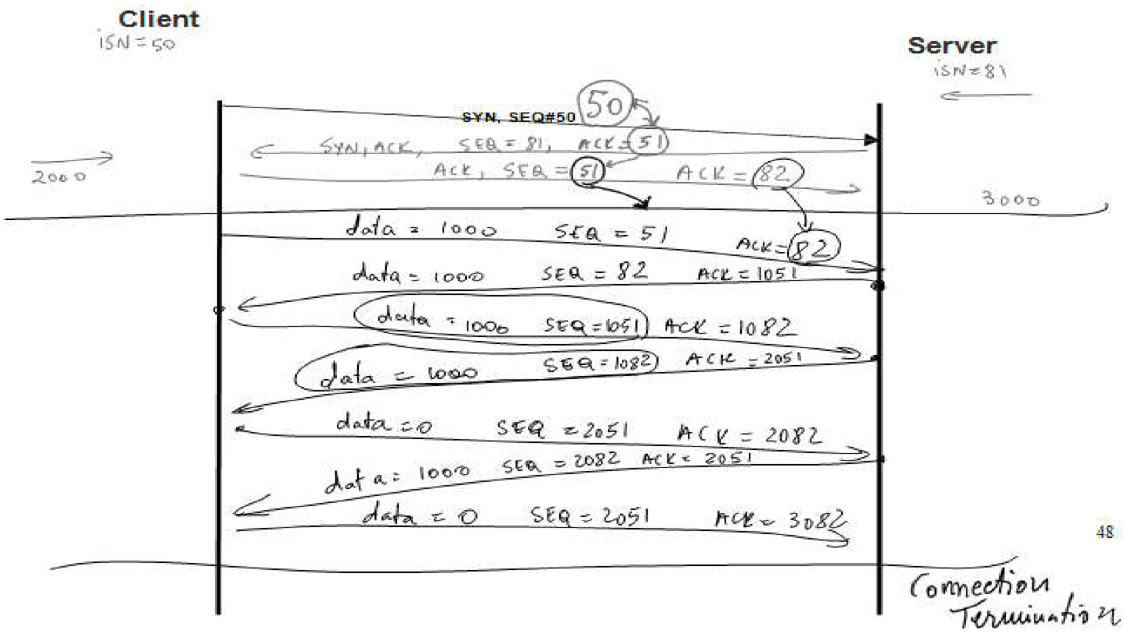
Sequence Number: 5000 Urgent pointer value: 200 **0.5**
URG (Control Field) = 1 (Enabled)
The first byte of Urgent data is 5000 and last byte of urgent data is 5200
Rest of the bytes in the segment (if present) are non-urgent data.

Q.No 5	Solution	Mark
a	**Key differences between UDP & TCP (Students can write any Five)**	1

Connectionless: UDP	Connection Oriented: TCP
• Simple • Fast • Best-effort Service • More suitable for Real-time applications: like Audio, Video • Simple Header with less overheads (8 bytes) • No Handshaking • No sequence number • Checksum based Error Control • No Flow Control • No Congestion Control	• Relatively Complex (State Machine based) • Reliable, Guaranteed Service • More suitable for reliable applications: like HTTP, FTP, SMTP etc. • Complex Header with more overheads (20 – 60 bytes) • 3-way and 4-way handshaking (3 phase 1. Connection Establishment, 2. Data Transfer and 3. Connection Termination • Segments are having Sequence Number for Error Control as well as Sliding Window based Flow Control &Congestion Control

Numerical Solution:

MSS=1000



1.5

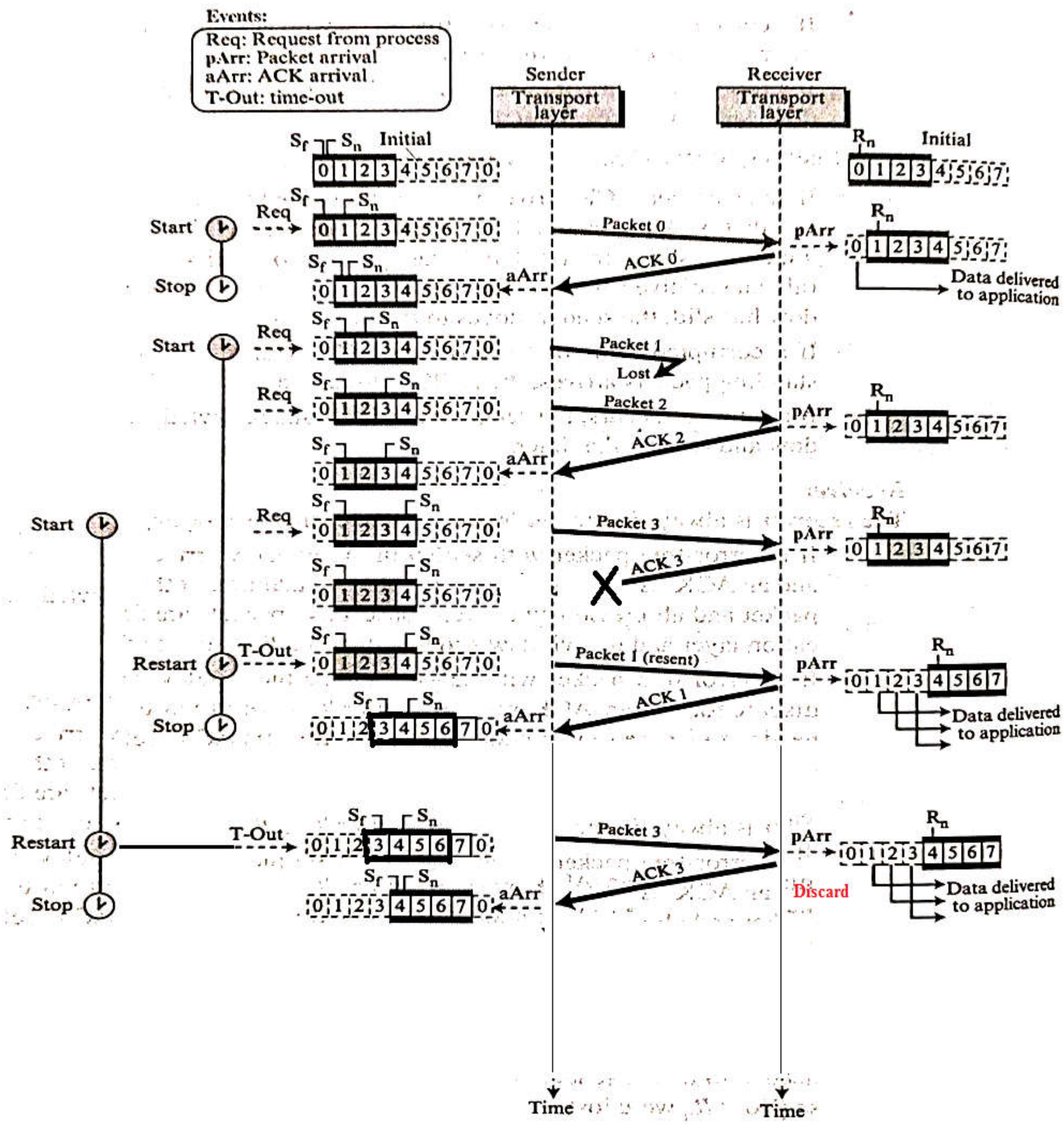
b **Selective Reepeat ARQ Protocol for Noisy Channel:**

In a Noisy Channel, Selective Repeat ARQ protocol is implemented with ***m=3*** (Number of bits for Sequence Number). Given that, **packet 0** is sent with successful acknowledged. **Packet 1** gets lost due to noise. **Packet 2&packet 3** are successfully sent. However, only packet 2 is successfully acknowledged, whereas packet 3's acknowledgement is lost due to noise. With suitable flow diagram show the process of flow & Error control using sliding window with S_f , S_n, R_n, timer and ACK.

M= 3 0.5

Size of Window = 2 $^{m-1}$ **= 4**

Packet 1 is Retransmitted (after Timeout)
Packet 3 is retransmitted (after Timeout but discarded at the reciver)



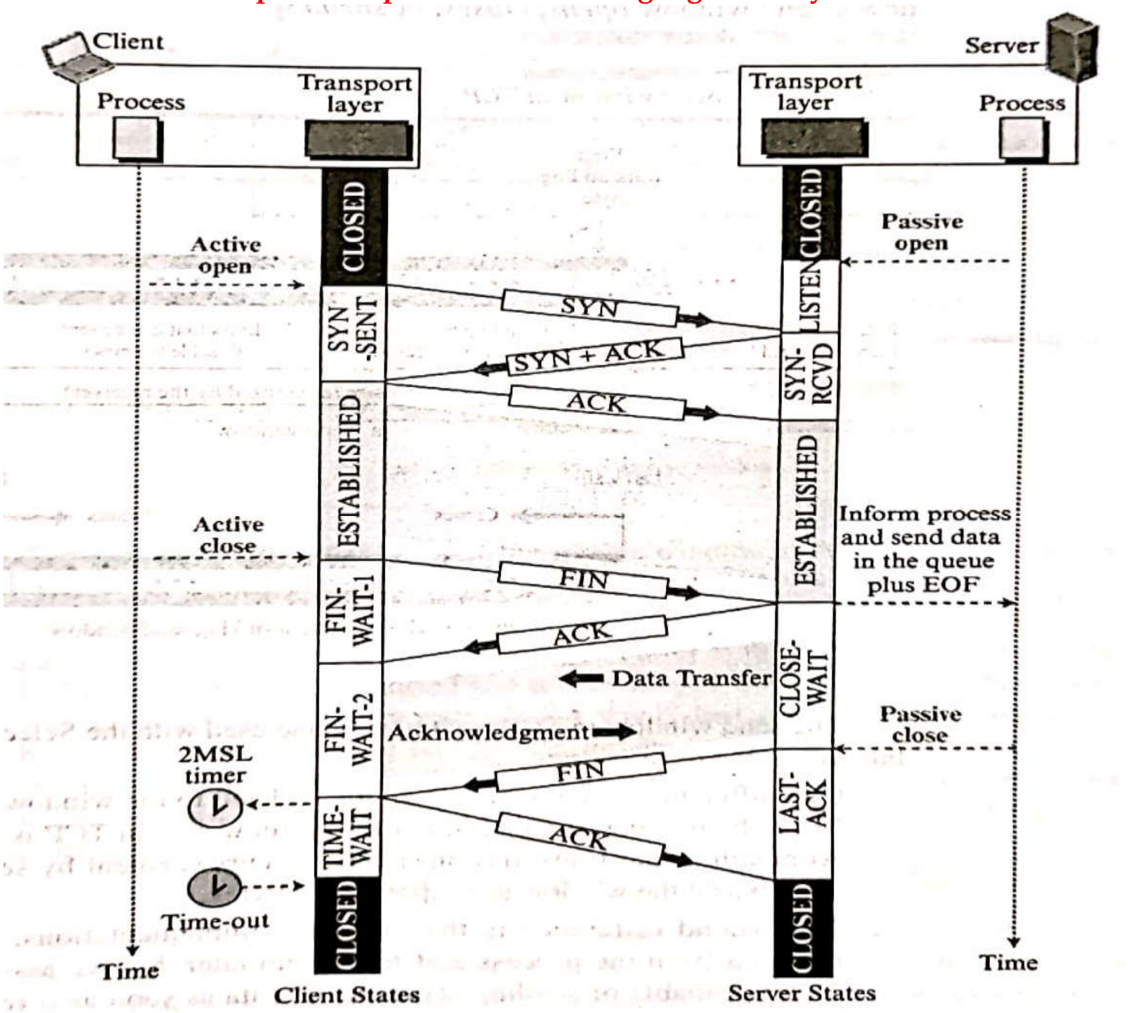
2

Q.No 6 **Solution** **Mark**

a **Half close & Full close in TCP Connection Termination:**

- **Half close (4 way handshaking→ FIN , ACK , FIN, ACK)** 1
- **Full close (3 way handshaking→ FIN, FIN+ACK, ACK)**

Students should explain the process in their language (briefly state)



- **4 way handshaking for connection termination**
- **FIN , ACK, Rest of data transmission , FIN, ACK (process has to be briefly stated)** **1.5**

Half Close Scenario in TCP Connection Termination

State transition diagram for half close:

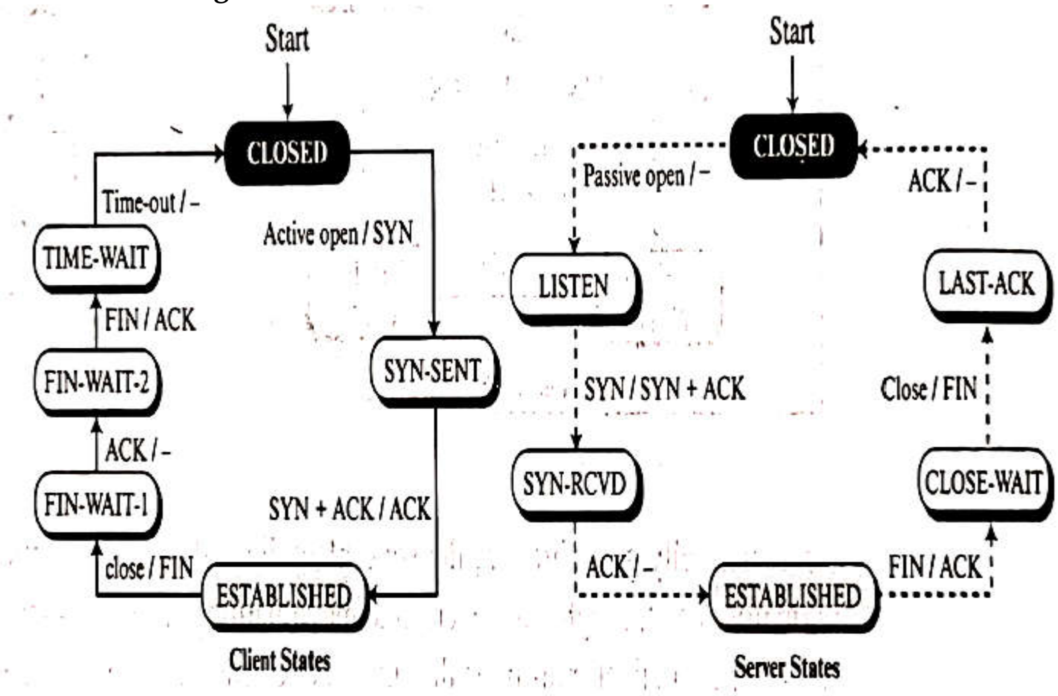


Fig. Transition diagram with half-close connection termination

b **OSI Layer Architecture (Features & Responsibilities- 7 Layer Top-to bottom)** **2.5**

Layer	Features & Responsibilities
Application	Responsible for providing service to the end user. DNS, HTTP, FTP, Electronic mail, P2P, telnet, SSH, SNMP etc.
Presentation	Responsible for translation, compression and encryption.
Session	Responsible for dialog control and synchronization.
Transport	Responsible for delivery of a segment from one process to another (process-to-process), Service point addressing (Socket), Segmentation and reassembly, connection control, flow control, error control.
Network	Responsible for the delivery of individual packets from source host to destination host, logical addressing (IP Address), Routing (finding the shortest path).
Data-Link	Responsible for moving frames from one hop (node) to the next, framing, physical addressing (MAC Address), flow control, error control, access control.
Physical	Responsible for movement of individual bits from hop (node) to the next, Physical characteristics, representation of bits, data rate, synchronization of bits, line configuration, physical topology, transmission mode.

***** END OF SOLUTION & SCHEME OF EVALUATION*****