

Transport Layer :- "Transport Layer is responsible for process to process delivery or end to end delivery."

* Port addresses are the logical ports that are given by TCP/IP Software at runtime.

Port address

→ 16 bit → $(0 \text{ to } 2^{16}-1)$

- 0 to 65535 port address.

By IANA :-

0 - 1023 ⇒ fixed ports, predefined ports,
(universal ports.

Ex:-

http ⇒ 80 ← Port No

ftp ⇒ 21
20 > Port No

SMTP ⇒ 25 ← Port No

} fixed ports

1024 - 49151 ⇒ Registered ports

49152 - 65535 ⇒ Dynamic ports, Ephemeral ports.

* Predefined ports are the ports which are used for predefined services like: http, ftp, SMTP etc.

* Registered ports are the ports which are used by the companies to test their softwares.

* Dynamic ports are the ports which are used to distinguish different processes in the Network.

①

②

h₁ http://www-google.com

→ ☒ S.IP D.IP

Source Port Address → SP₁ 60000 , DP 80 → Destination Port Address

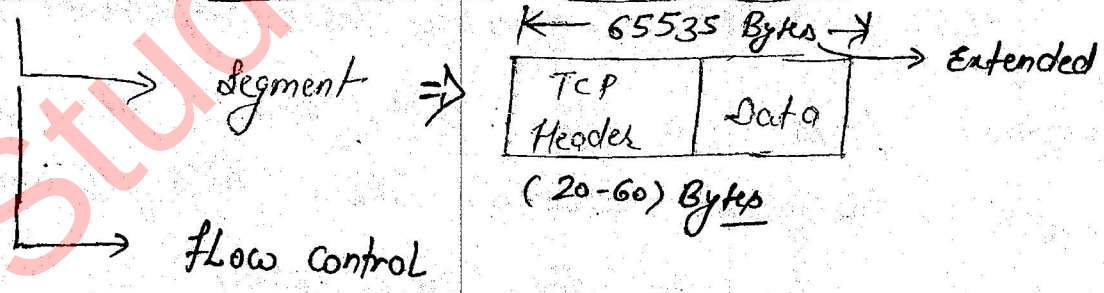
→ ☒ S.IP D.IP
SP₁ DP
61000 , 80

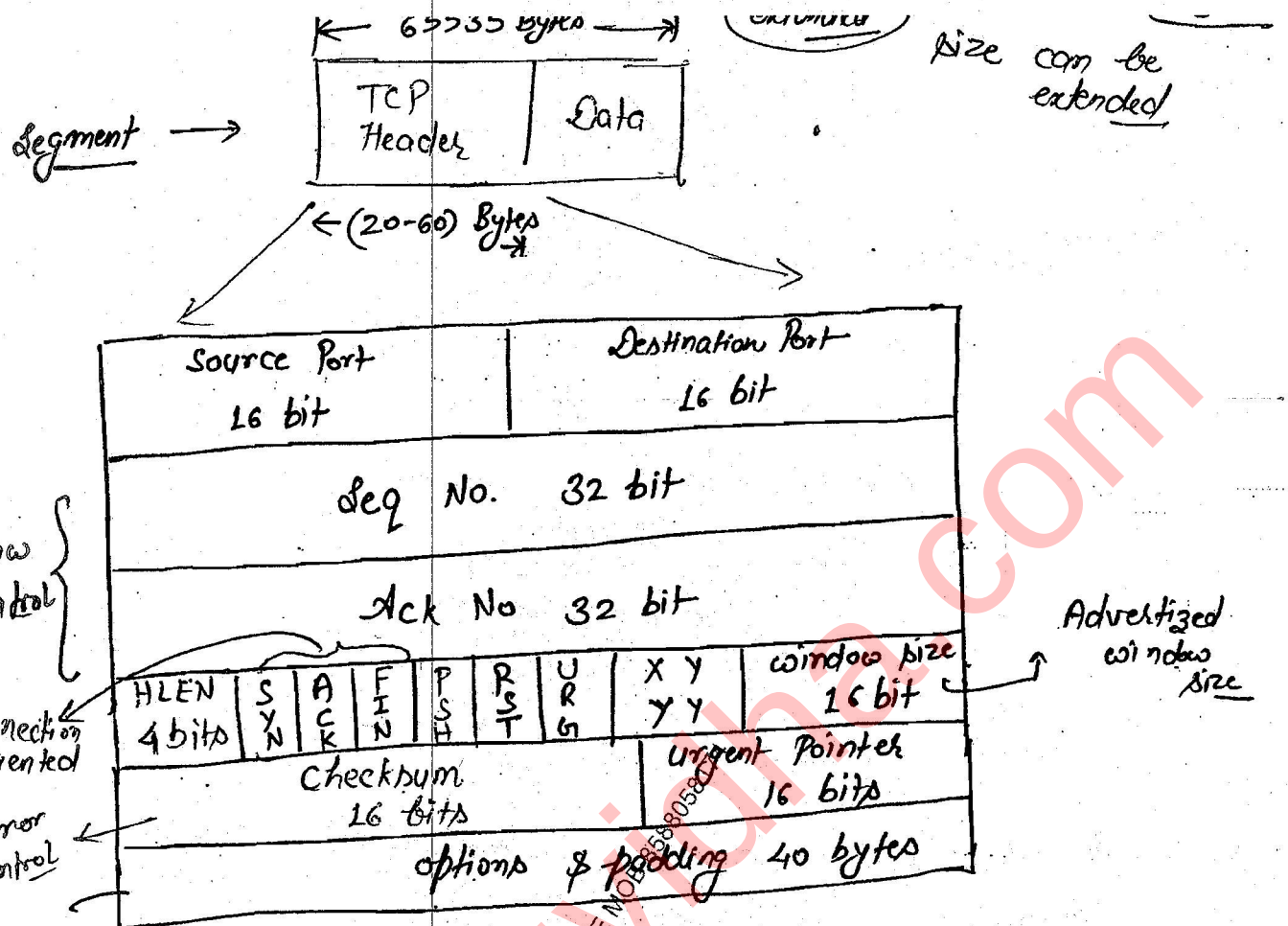
→ ☒ S.IP D.IP
SP₁ ~~62000~~ DP
64000 , 80

* When the 'Source port' is a dynamic port and the destination port is a fixed port then data is moving from client to server.

* When the 'source port' is a fixed port and the destination port is a dynamic port, then data is moving from server to client.

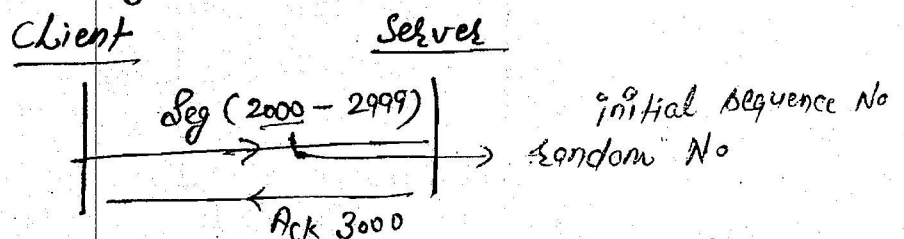
TCP :- Transmission control protocol :-





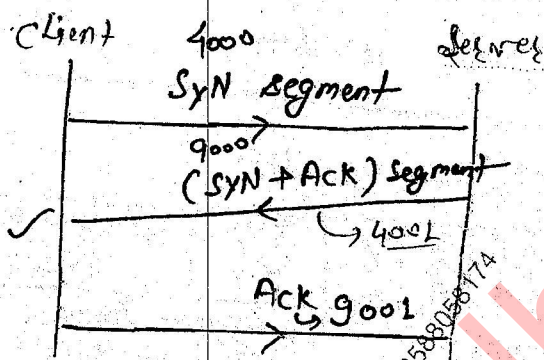
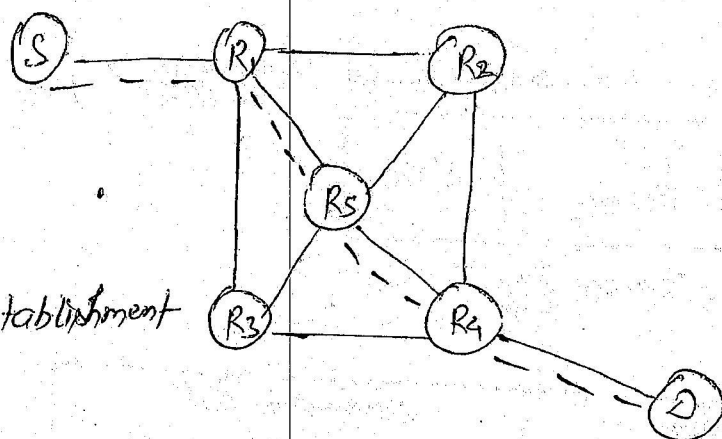
* In data link layer the sequence number is given for every frame whereas in transport layer TCP, sequence No is given for every byte in the segment.

* The initial sequence number will be a random No. in the case of TCP to provide security.

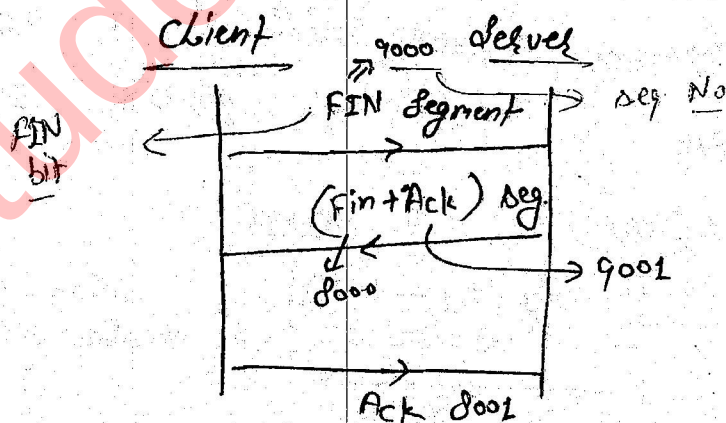


* TCP supports full duplex operation i.e. data can be transmitted in both directions simultaneously.

(i) Connection establishment



- * for complete connection establishment 3-way Handshaking is required.
- * Sync segment doesn't carry any data but it consumes 1 sequence number.
- * During connection establishment phase control segments are transmitted.



Handshaking is required.

Flow Control policies of TCP:-

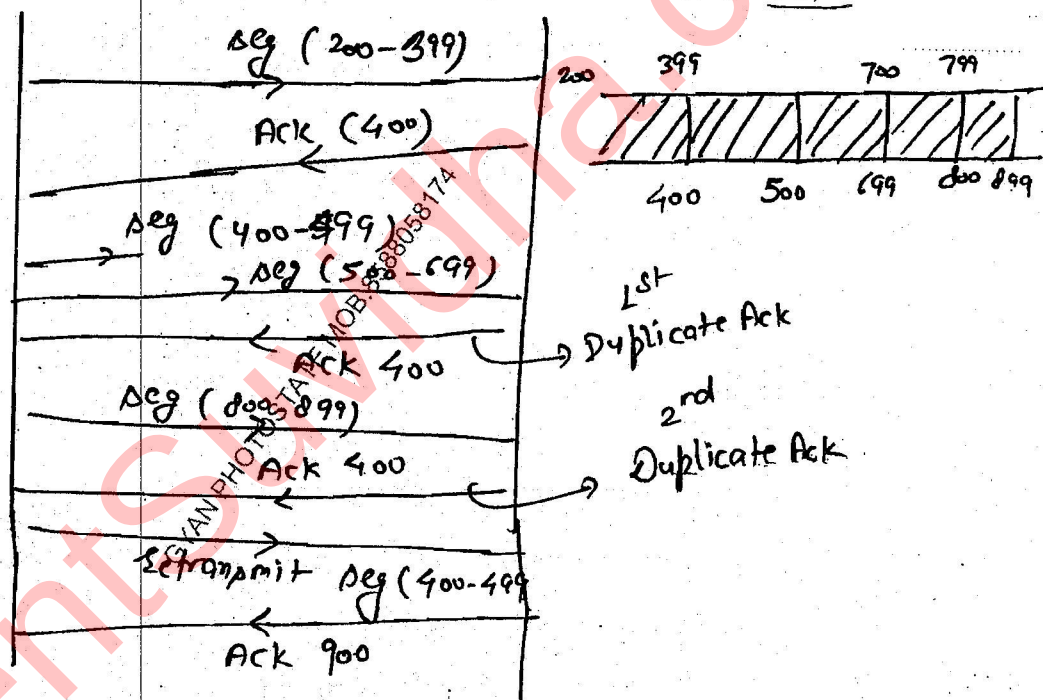
* TCP can accept out of order segments but always sends in-order Acknowledgements.

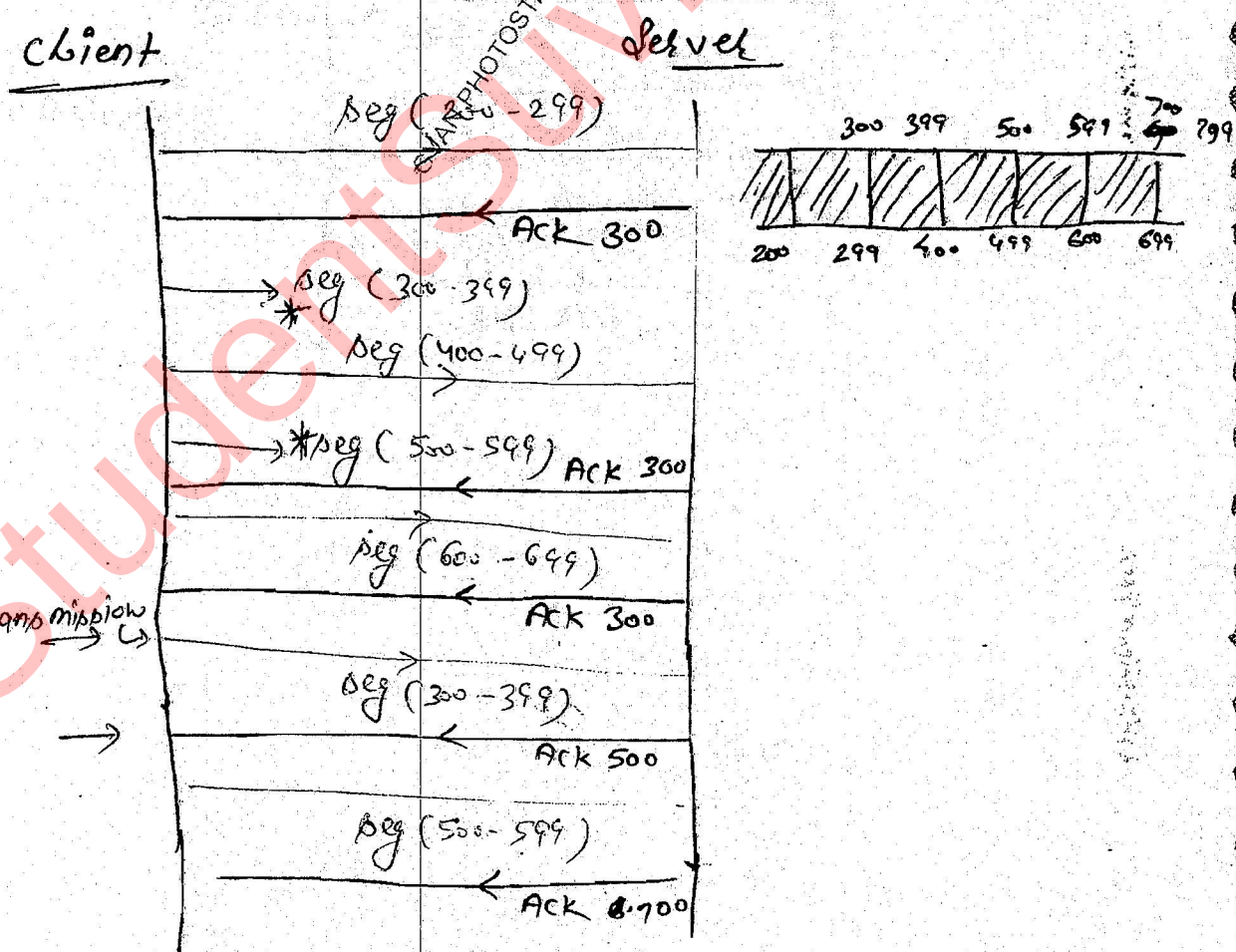
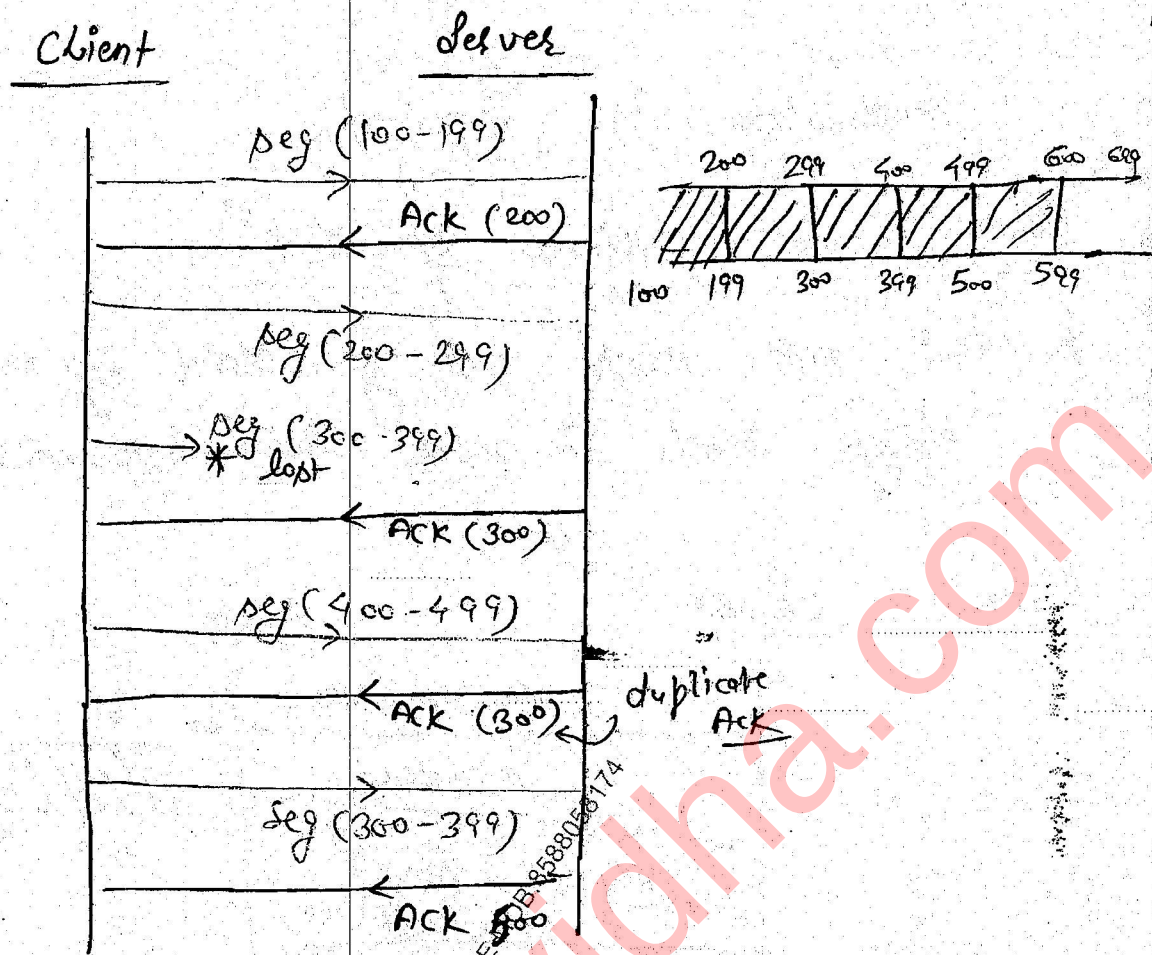
S' TCP

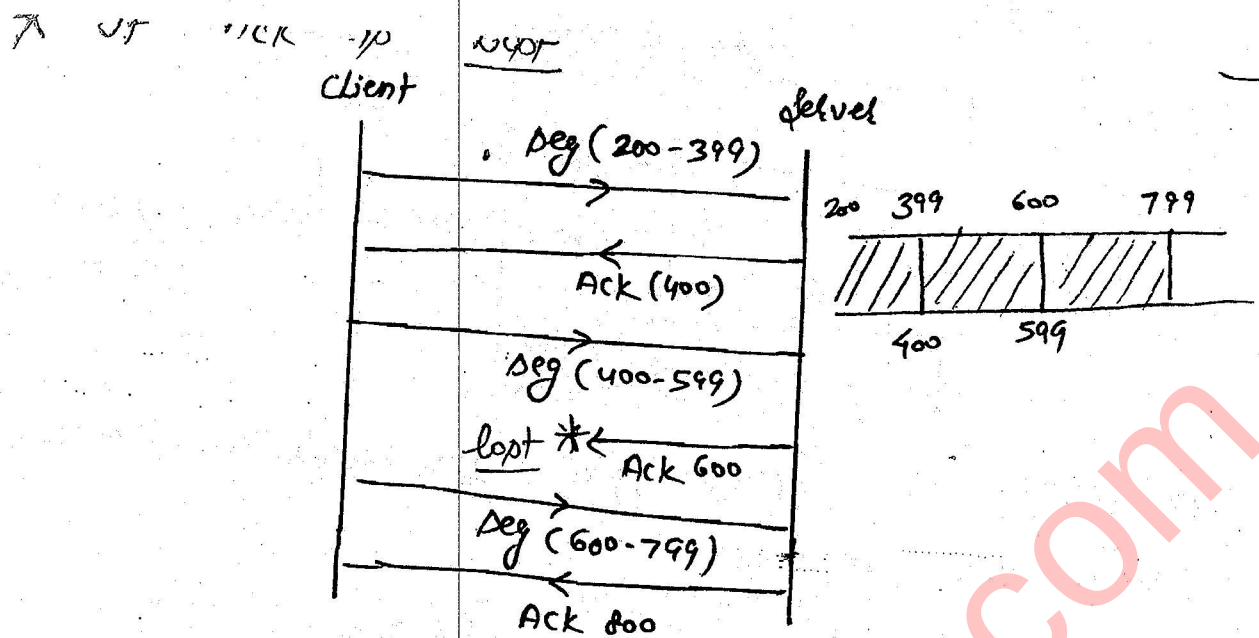
Client

server

R's TCP (Application Layer)

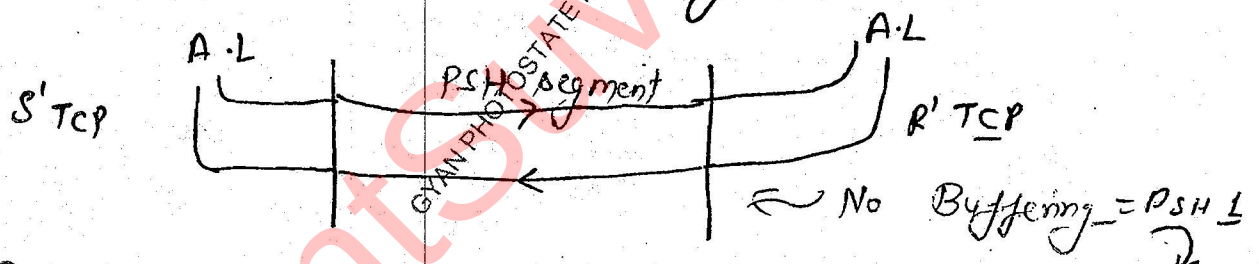






* The Next upcoming Ack will nullify the previously lost Ack.

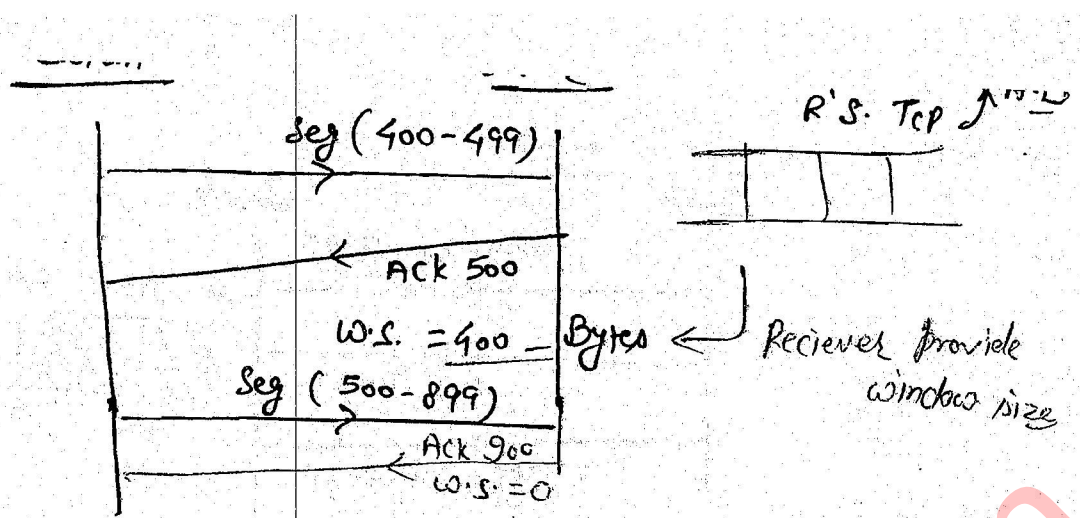
* HLEN (Header length) will indicate the size of the header in TCP segment



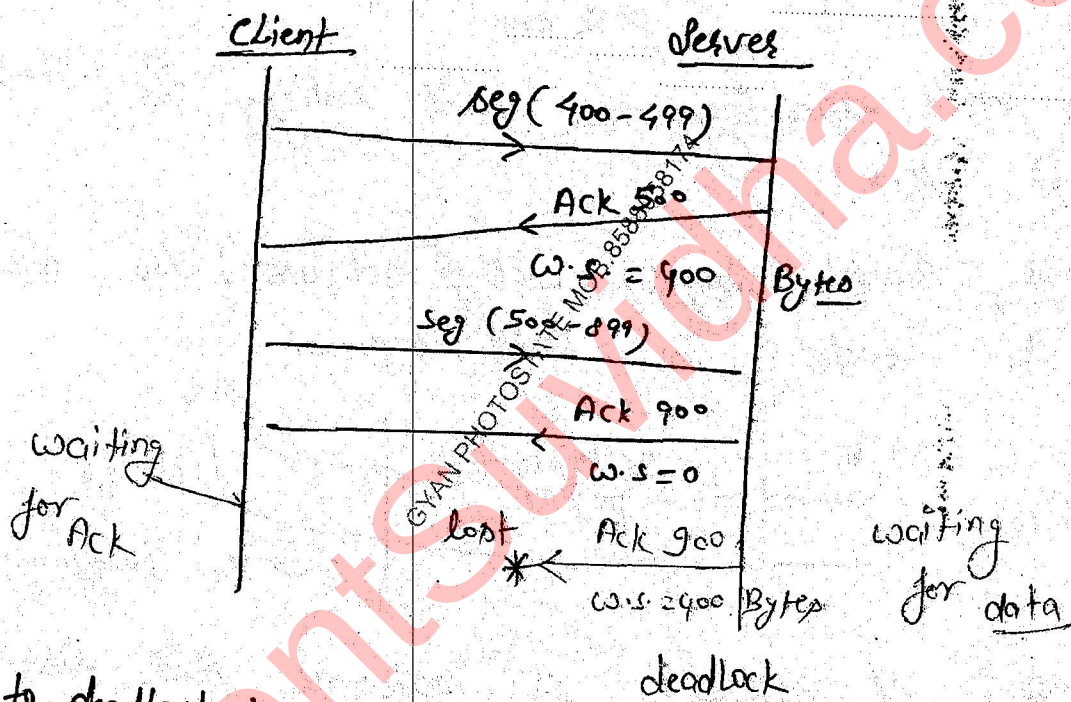
* PSH :- If $PSH = 1$, it indicates that, data is interactive and response is immediate.

* RST :- If $RST = 1$, it is used to suddenly close the connection and re-establishing the connection.

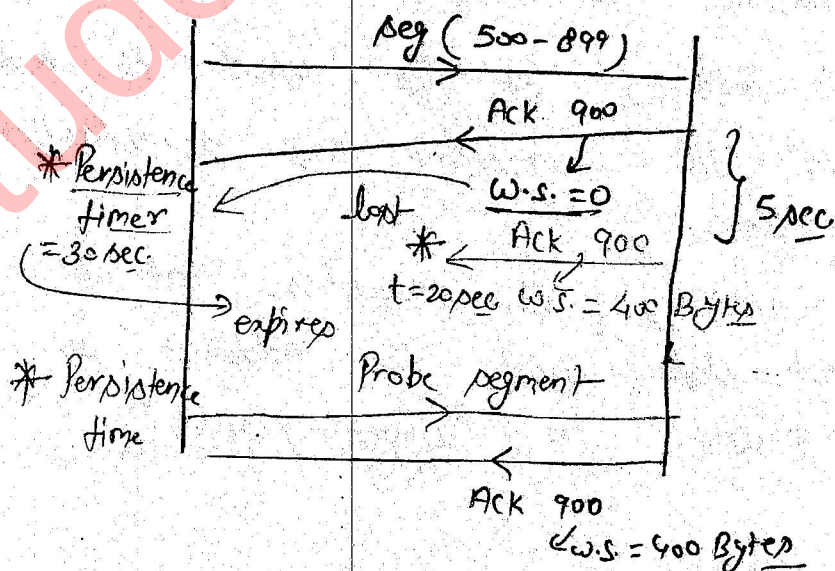
* URG :- If $URG = 1$, it indicates that it is a urgent data and the address of urgent data is available in 'urgent pointer'.



* window size is used for synchronization between sender and receiver.



Solution to deadlock :



* The problem of overhead will be resolved using persistence time.

Date
18/06

Date
18/06/2017

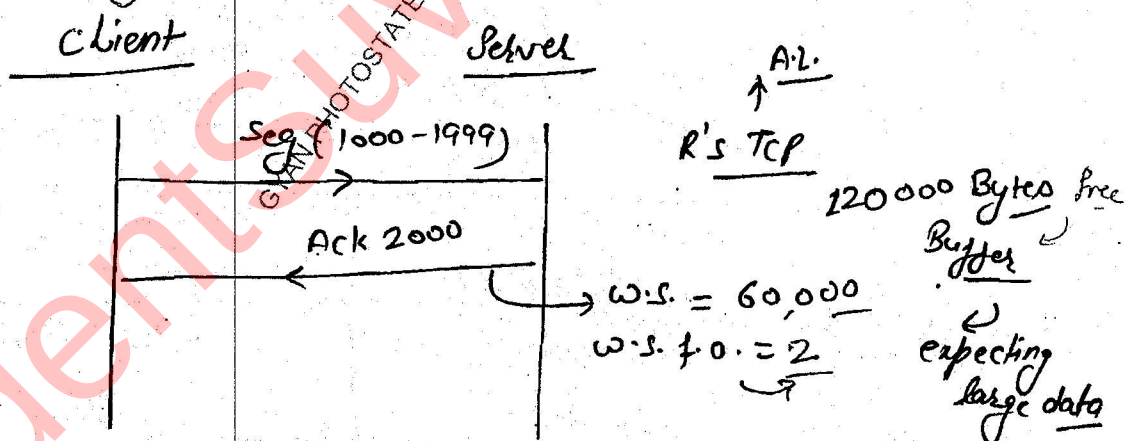
* Checksum :- TCP provides error control because of potential errors occurring at the routers.

* CRC circuit can not be designed for large data so it can not be applied in TCP or IP.

* Checksum can be used because vertical bit error will not occur because of large data.

* Options & padding :-

(i) window scaling factor option :-



* For extending the window size > 65535 , window scaling factor option enabled.

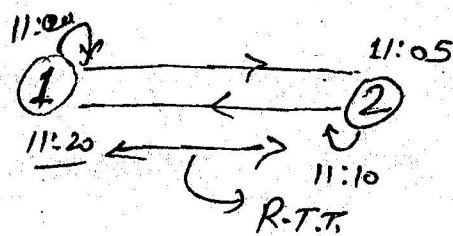
* In TCP the segment size can be of any size.

* Segmentation happens when transport layer giving data to Network layer.

* Fragmentation happens when Network layer giving data to D.L.L.

(2) Time Stamp option :-

Time stamp option is used to calculate Round trip time between two end process.



(iii) NOP option :-

fill the gaps

No option (Nop option) is used to fill the gaps between options.

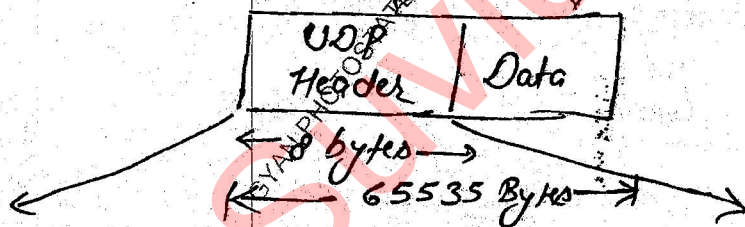
(iv) EOP option :-

End of option is used as the separator between header and data.

UDP Protocol :-

↳ Data → Datagram

UDP Header :-



8 Bytes	Source Port 16 Byte bit	Destination Port 16 bit
	checksum 16 bit	total length 16 bit

* total length indicates the size of the datagram.

(1) Total length bits
 $\Rightarrow 0000000011111111$

Size of datagram = 255 Bytes

* Header + Data = Datagram

fixed ← 8 Bytes + Data = 255 Bytes

Data = 247 Bytes

* If total length is given, we can calculate size of the datagram as well as size of the data.

(2) UDP Header is given as:

$(\text{FFFF } 0050 \text{ FFFE } \text{FFFE})_{16}$

then: $\hookrightarrow$ Sp $\quad$ Dp $\quad$ Dynamic Port

(i) calculate the source port $\text{FFFF} \rightarrow 65520$

(ii) calculate the destⁿ port $0050 \rightarrow 80$ (fixed port)

(iii) calculate the size of datagram $\text{FFFE} \rightarrow 65534$

(iv) calculate the size of payload value $\rightarrow 65534 - 8$
 $= 65526$

(v) Is the datagram travelling from client to server or server to client? $\checkmark$ Client to server

$\text{FFFF} \quad 1111 \quad 1111 \quad 1111 \quad 0000 \rightarrow 65520$

$\hookrightarrow$ or $\text{FFFF} - 000F$

$= (65535 - 15) \Rightarrow 65520$

$\text{FFFE} = (\text{FFFF} - 0001) \Rightarrow 65534$

Dynamic Port $\rightarrow$ fixed port
 $\text{Sp} \quad \text{Dp}$

Note :- If Both source and destination ports are fixed port then it is DHCP protocol.

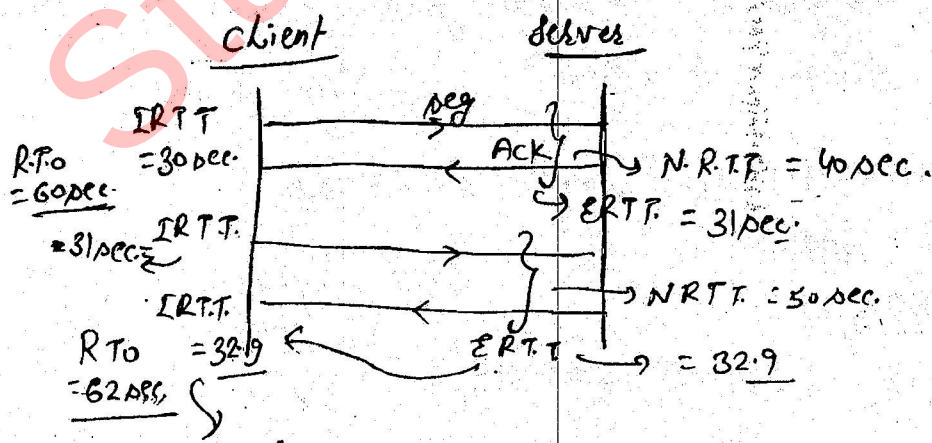
TCP

- * Header size (20-60) Bytes
- * Has flow control
- * TCP with IP is Connection oriented service
- * Checksum is mandatory
- * has error control.
- * Doesn't depend on ICMP Protocol. for error control.
- * http, SMTP, FTP, Telnet will use TCP
- * Doesn't support multicasting, & broadcasting.

UDP

- * Header size 8 Bytes.
- * No flow control
- * UDP with IP is Connection-less service
- * checksum is 'optional'.
- * has no error control
- * depends on ICMP Protocol.
- * DNS, TFTP, SNMP will use UDP.
- * support multicasting & broadcasting.

RTO Timer :- (Retransmission after Time-out timer)



$$NRTT = 40 \text{ sec.}$$

$$\alpha = 0.9$$

$$ERTT = \alpha * IRTT + (1-\alpha) * NRTT$$

$$= 0.9 * 30 + 0.1 * 40$$

$$ERTT = 27 + 4 = 31 \text{ sec.}$$

$$\rightarrow IRTT = 31 \text{ sec}$$

$$NRTT = 50 \text{ sec}$$

$$ERTT = 0.9 * 31 + 0.1 * 50$$

$$= 27.9 + 5 = 32.9$$

★ Same times is used for sending back to back segments.

for what value of α , Best Estimation R.T.T. will be the average of initial R.T.T and New R.T.T = ?

$$\Rightarrow \underline{0.5} \text{ Ans. } \alpha = \underline{0.5}$$

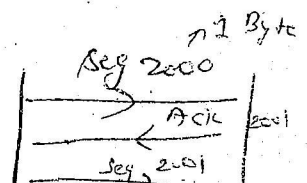
$$ERTT = 0.5 * IRTT + (0.5) * NRTT$$

$$ERTT = \frac{IRT T + NRTT}{2}$$

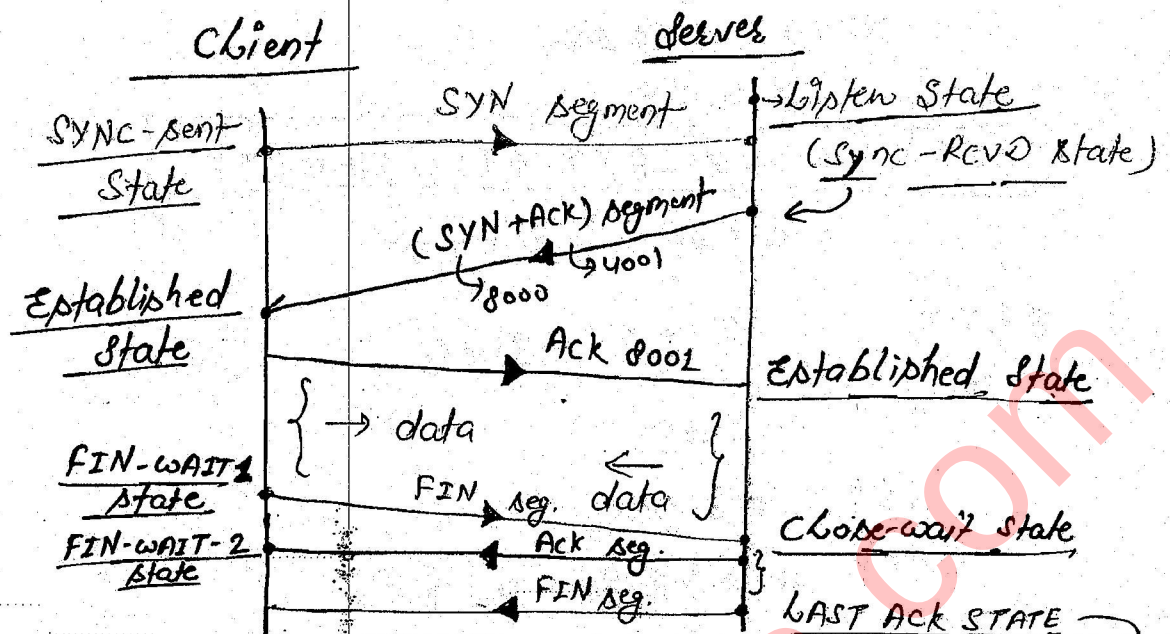
i.e. Avg of $\leftarrow$
IRT T & NRT T

★ (2) when Sequence No of data segment is m then when does the sequence no of next segment is $m+1$.

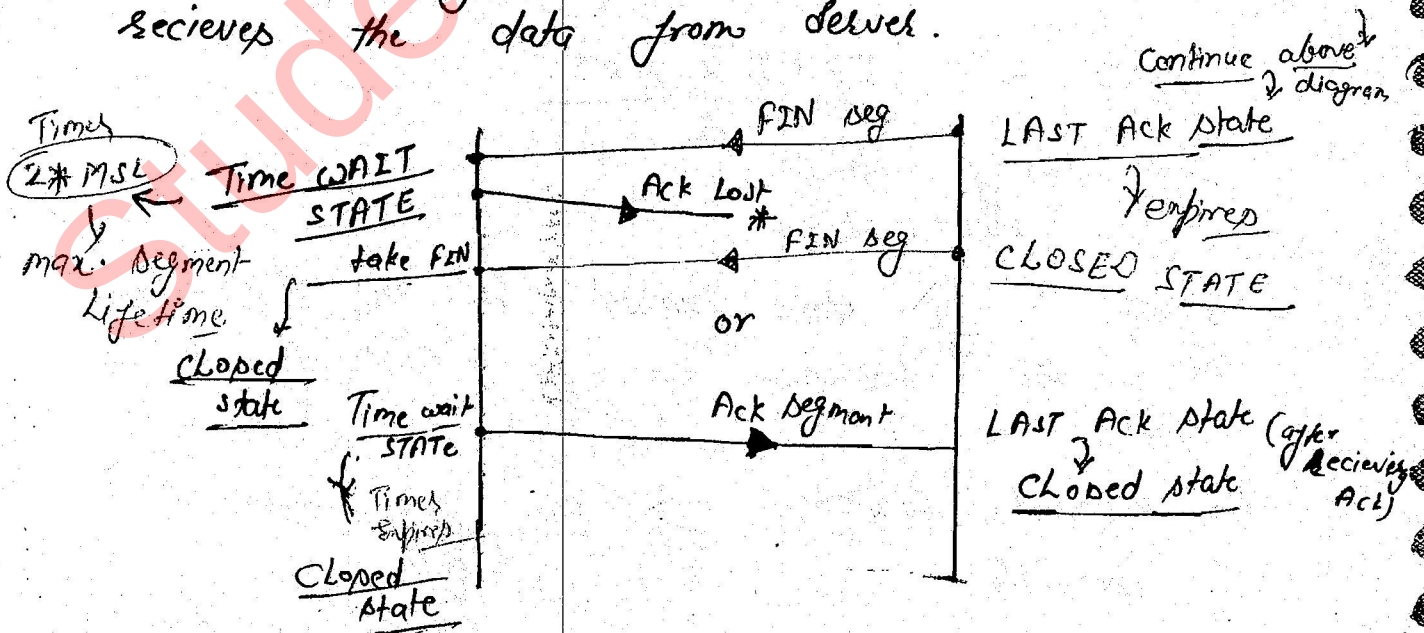
the segment contains only one byte then it is possible



STATE TRANSITION



- * Once Sync segment reaches to server, server will move from Listen $\rightarrow$ Sync-Received state.
- * Client will move from Sync-sent state $\rightarrow$ Established state when it gets (Sync+Ack) segment.
- * Server will move from Sync-Rcvd state $\rightarrow$ Established state when it gets Ack from client.
- * Both in FIN-WAIT-1 & FIN-WAIT-2 client will not send any data to the server but it receives the data from server.

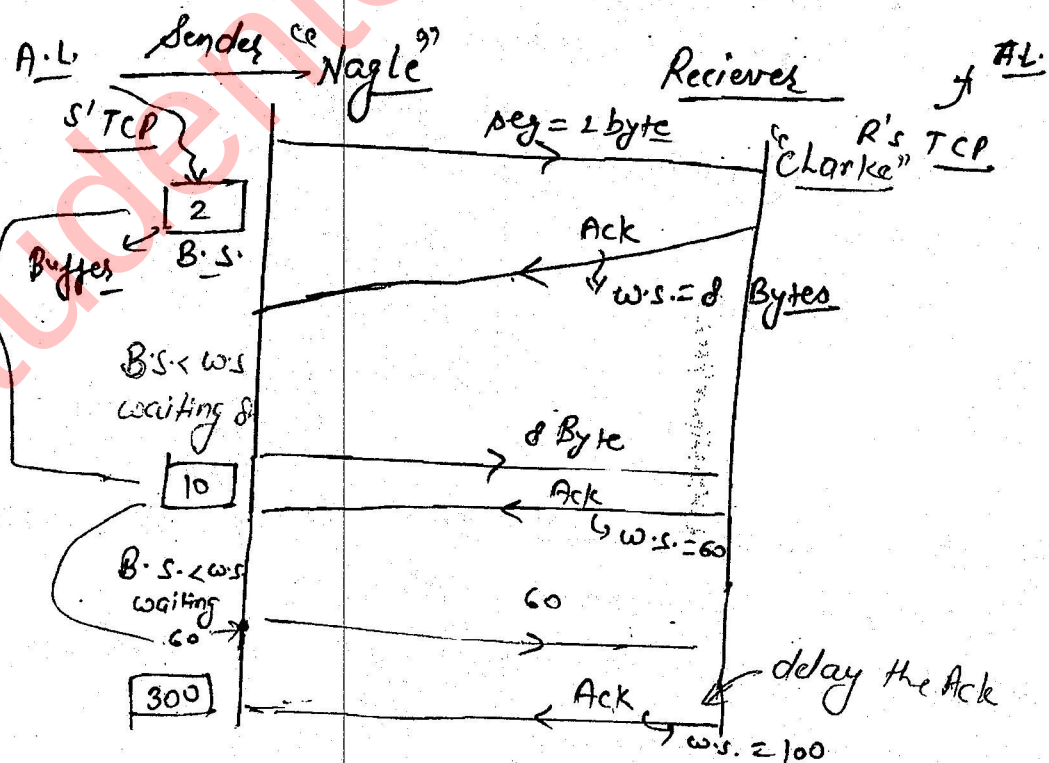
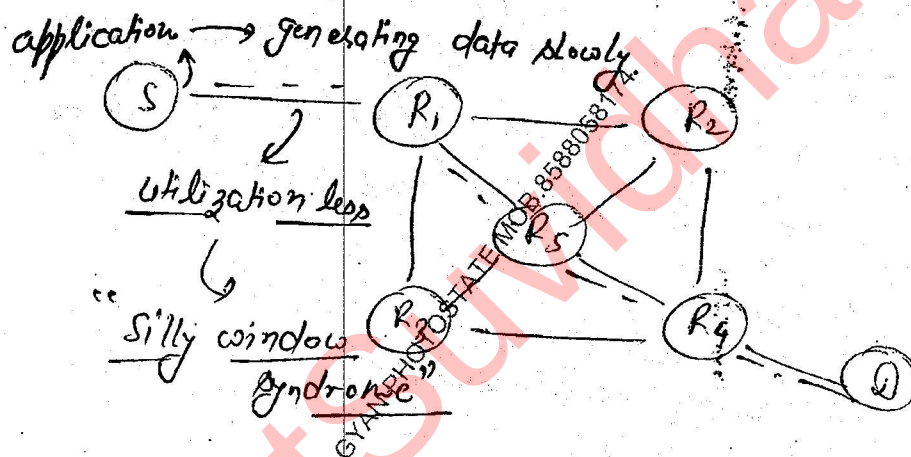


* The client will move from FIN-wait 2 state → time wait state when it gets (FIN + Ack) from server

* Server will move from Established state → LAST ACK STATE when server sends (FIN + Ack) segment.

* Silly window syndrome :-

* When the application has generated data slowly then the window size will be small so utilization of TCP will be less.



* when application "Nagle" suggested, send the 1st Byte as it is, and buffered the remaining data.

* Once Ack reaches to client, compare the buffer-size with the window size.

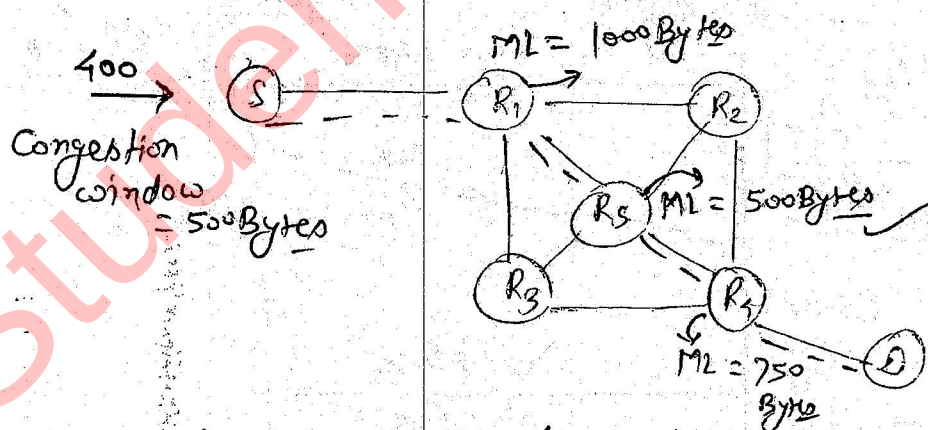
* If the Buffer-size is less, sender has to wait until the Buffer-size is equal to window size.

* So that Next time more data can be transmitted with accumulated data.

* "Clark" suggested that delay the Ack, so that parallelly window-size will increase along with the buffer-size so the problem of silly window syndrome will be fastly resolved.

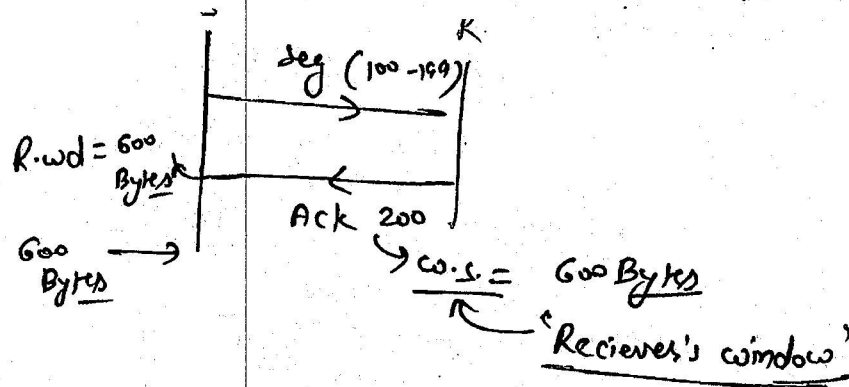
* Congestion policies of TCP :-

TCP $\Rightarrow$ Connection establishment
• Data transfer
• Connection release.



* Congestion window will be known to sender during connection establishment phase.

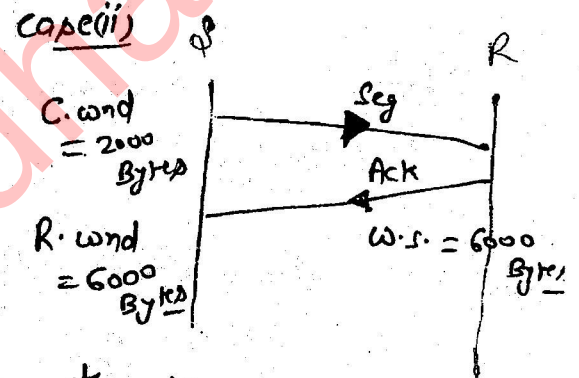
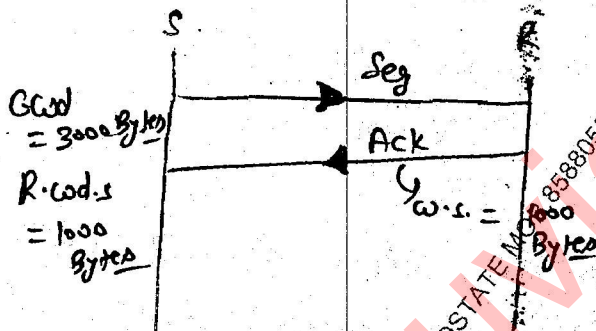
* Congestion window is one of the parameters for sender window.



* Receiver window will be known to sender during data transfer phase.

$$S.w. = \min(C.window, R.window)$$

Sender window
Case (i)



* If $R.wnd < C.wnd$ then
 $S.w. = R.wnd$
 'flow control' policies
 of TCP

* If $C.wnd < R.wnd$
 $S.w. = C.wnd$
 'Congestion
 Policies of TCP

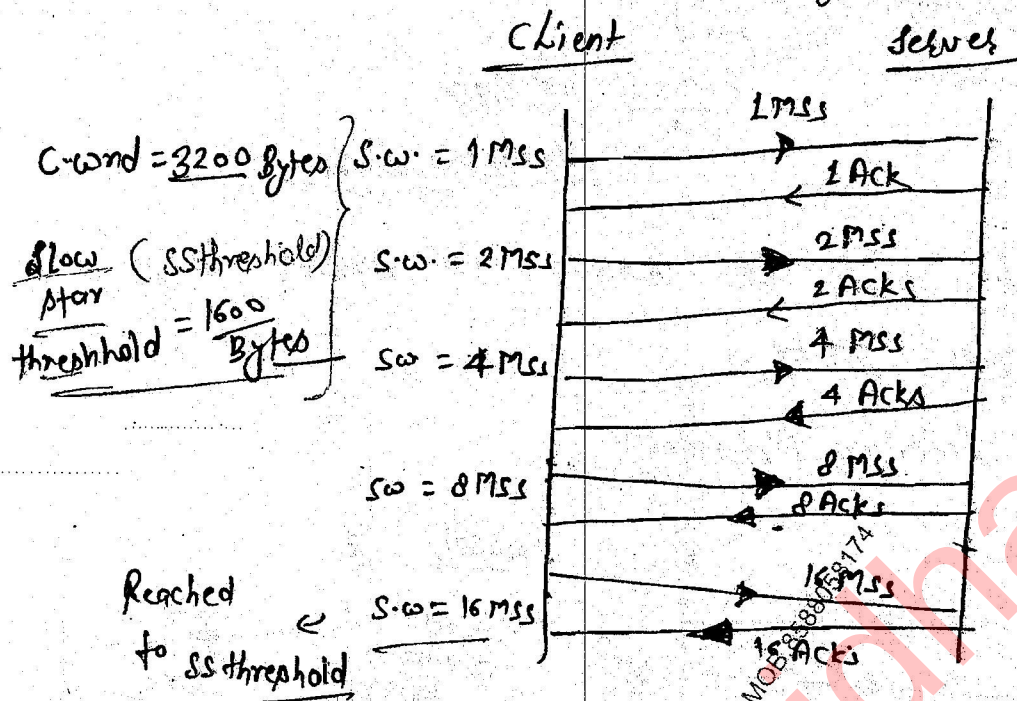
$$S.w. = \min(C.wnd, R.wnd)$$

Congestion policies of TCP:-

- (i) Slow start Algo
- (ii) Congestion avoidance
- (iii) Congestion detection

(i) Slow start algo :-

$MSS = 100 \text{ Bytes}$
(Max segment size)

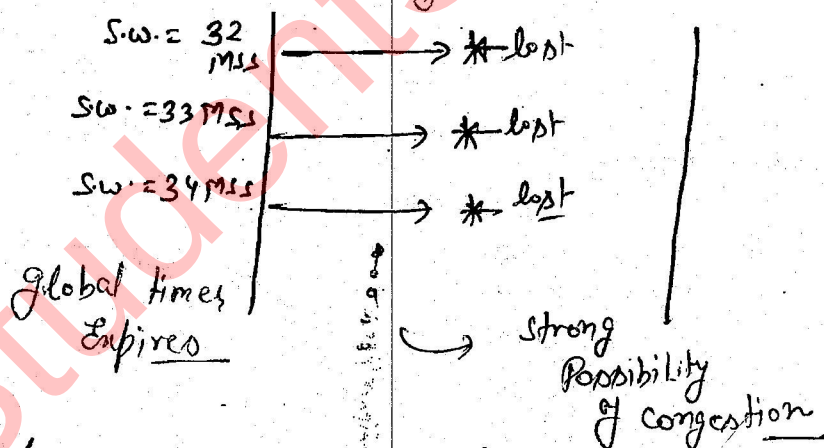
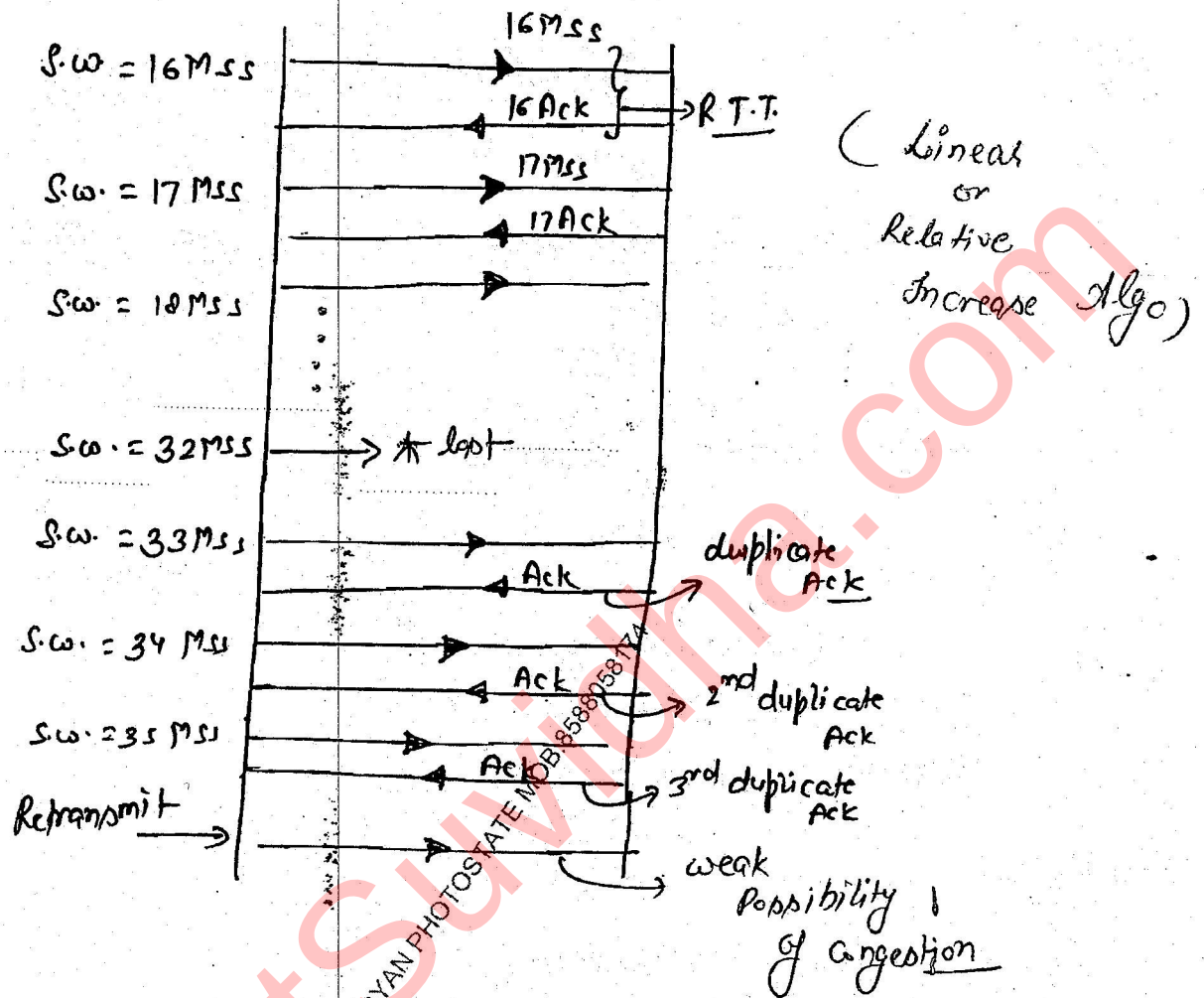


- * In Slow Start Algorithm, Data is transmitted in the form of MSS (max segment size).
- * The Increase of sender window size is based on Number of Acks.
- * In Slow Start Algorithm, Sender window size increases upto Slow-start threshold.

(Slow start algo = Exponential algo)

- * Initially S.W. = 1 MSS
- After 1 R.T.T, S.W. = 2 MSS
- After 2 R.T.T, S.W. = 2² MSS
- ⋮
- < SS threshold

(11) Congestion avoidance

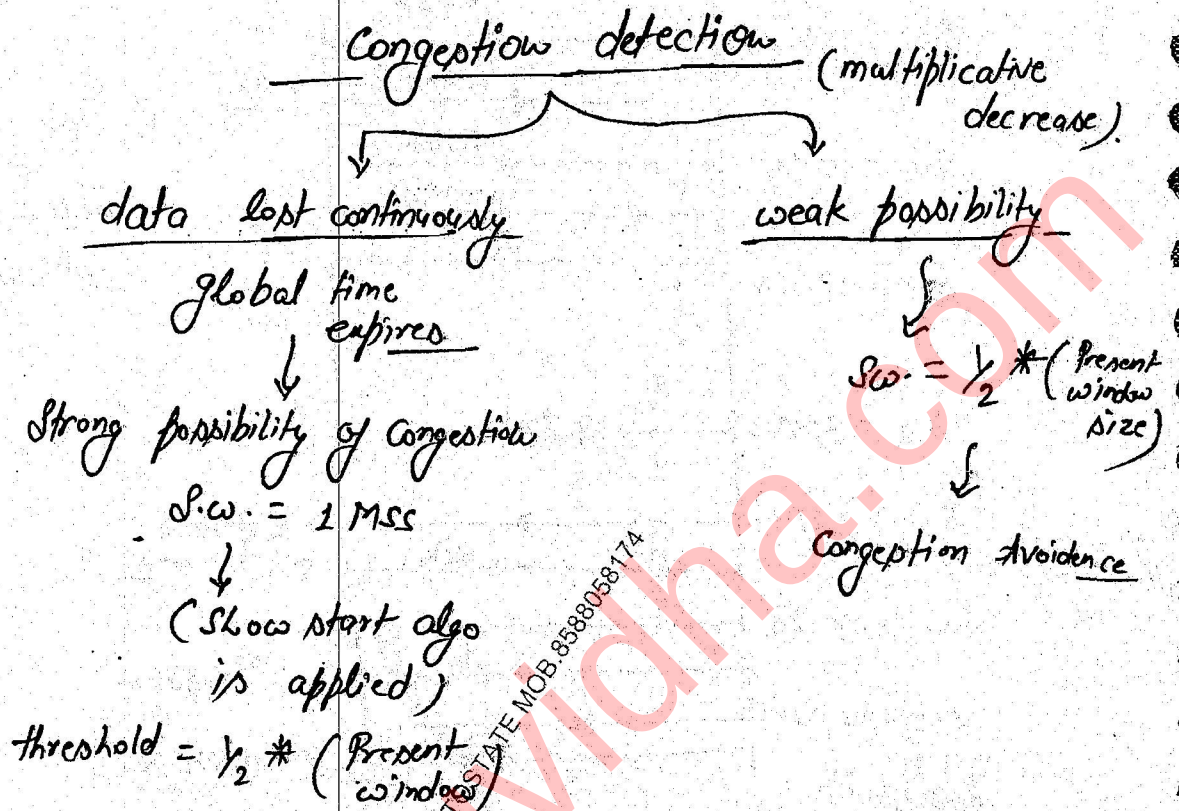


* In congestion avoidance algorithm, the increase of sender window size is based on R.T.T.

* If the data is lost continuously, until the global timer expires, it is treated as strong possibility of congestion.

★ If the data is not ACK'd, then it is treated as weak possibility of 'congestion'.

(iii)



Example :-

S.W. = 100 Bytes ← Slow start

200
400
800
→ 1600 ← threshold reached

1700 ← C.A.
1800
1900

lost * 2000
2100

weak possibility

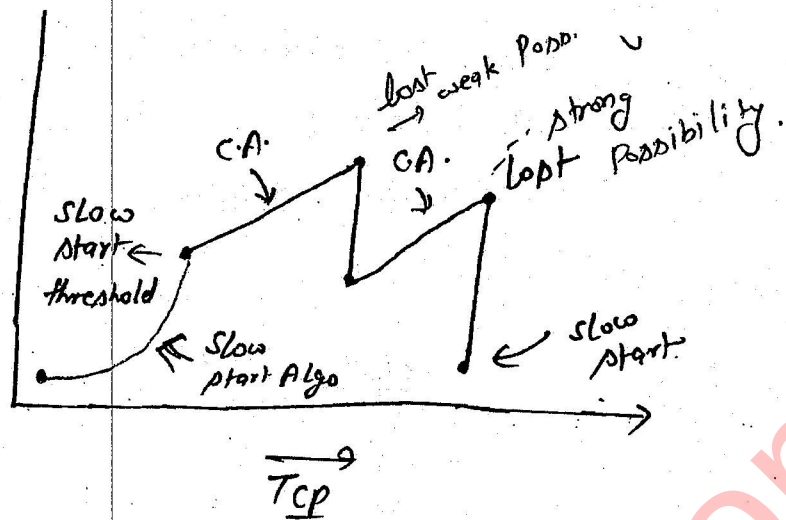
1000 ← C.A.
1100
1200
1300
1400
1500
1600

Strong possibility

* lost
* lost
* lost

$\frac{1}{2} * 1600$
threshold = 800

slow start
100
200
400
800 ← threshold reached
900 ← C.A.
1000



(1) $MSS = 100 \text{ Bytes}$

Present sender window = 800 Bytes

(i) what would be the next window

If slow start algo is applied?

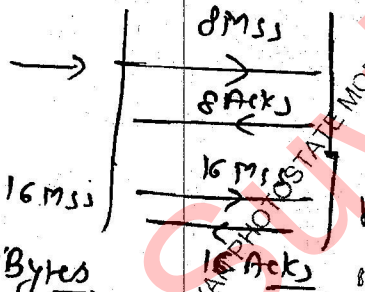
$$S.W. = 800$$

$$= 8 \times 100$$

$$= 8 MSS$$

$$S.W. = 16 MSS$$

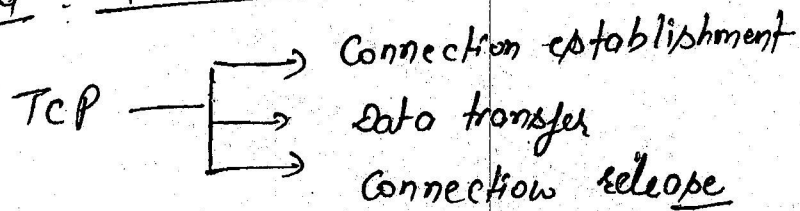
$$= 1600 \text{ Bytes}$$



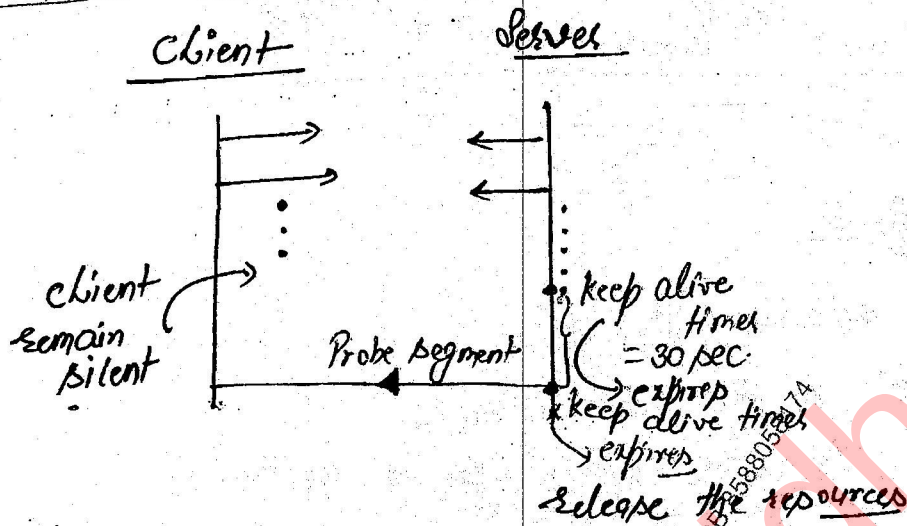
(ii) In above problem, If Congestion Avoidance is used then what would be the next window = ?

$$8 MSS + 1 MSS = 9 MSS = 900 \text{ Bytes}$$

Dated : 20/06/2011



* Keep alive timer :-



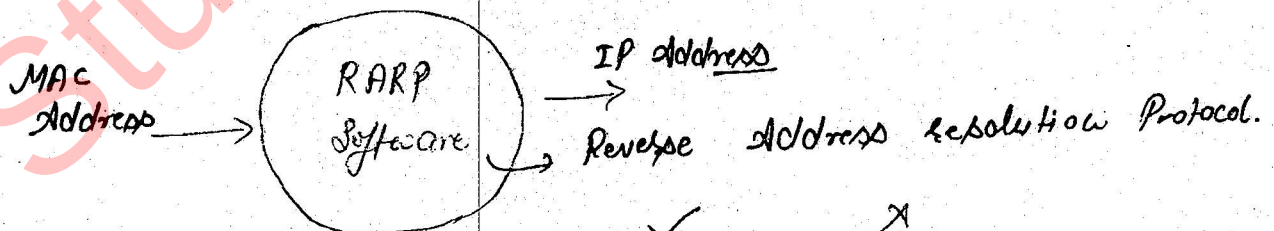
* The Purpose of keep alive timer is to find out whether the client is active or passive.

* This will be started by the server, once the client remain silent.

* The purpose of keep alive timer is to provide security to the transactions.

* RARP Protocol :-

→ RARP Software



(i) diskless station → MAC Address, IP Address

(ii) disk station → MAC Address, IP Address

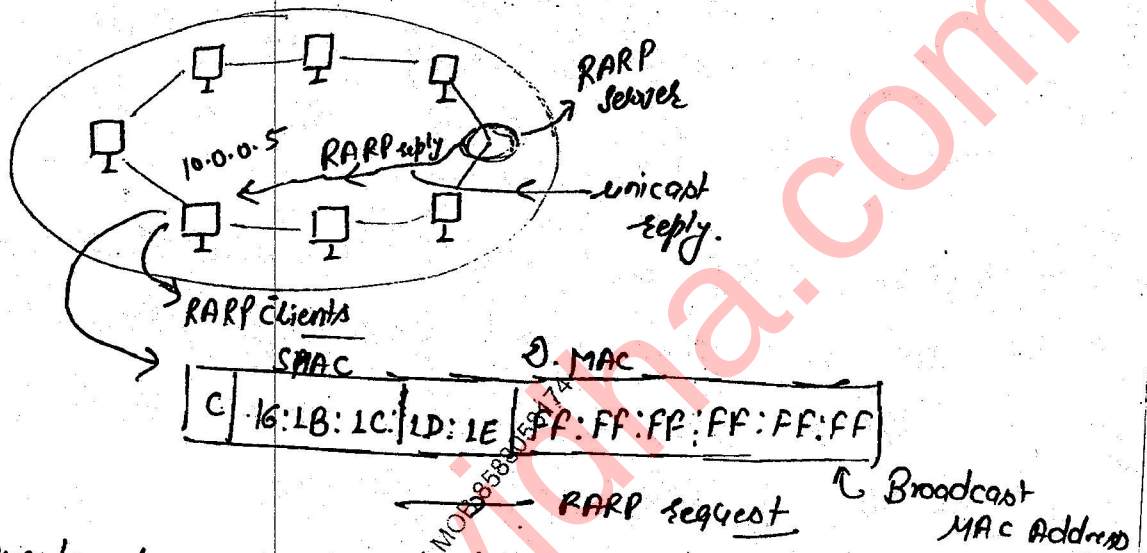
* MAC Addresses stored in ROM of computer.

ROM chip.

* The IP address of the computer is stored on HDD.

Clients $\Rightarrow$ diskless station

Server $\Rightarrow$ disk station



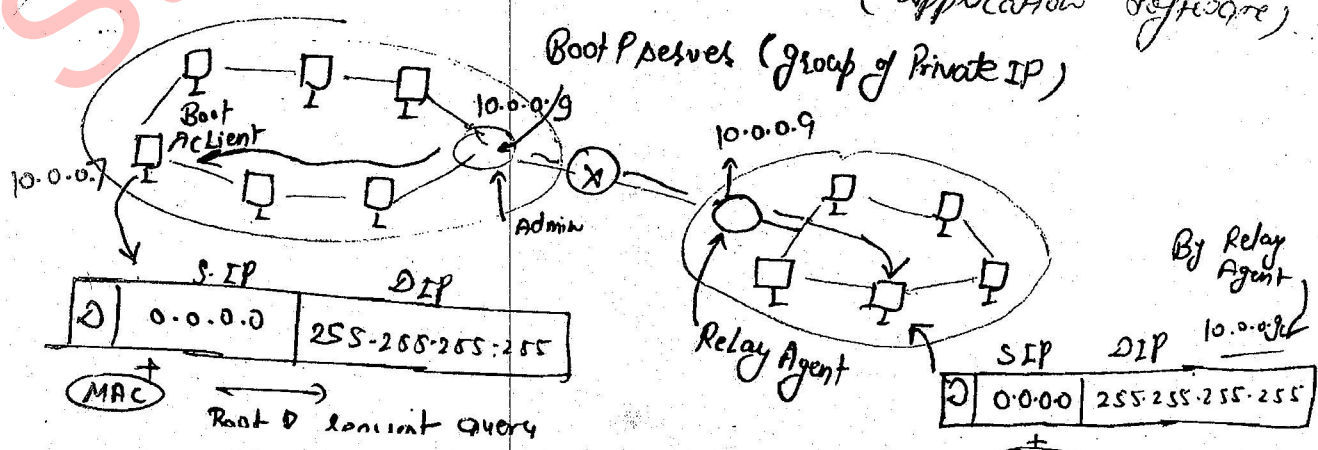
* RARP request packet is a broadcast query packet.

* RARP reply is a unicast packet replied with IP address to that computer at run time.

* The drawback of RARP is for assigning IP Address for the systems in N LAN Network then N RARP servers are required so cost becomes high.

* Boot P Protocol :-

Special address $\rightarrow$ Boot P Software (Application Software)



* The purpose of "J" is to convert limited broadcast packet into the unicast packet so that the router will allow the packet and reaches the BOOTP server.

* Using BOOTP protocol we can assign IP addresses in different LANs.

* Cost is less compared to RARP protocol. because it requires only one BOOTP server and $N-1$ relay agents for N LAN networks.

Note :- * Both RARP and BOOTP are statically binded that is, we can give only static addresses.

* In DHCP, we can assign both static IP as well as dynamic IP addresses.

Note - (2) :-

* The purpose of RARP, BOOTP, DHCP Protocol is for booting the Network. (for booting the diskless stations).

* Application of ARP to findout the physical location of system in the LAN (Network).