Date ___/___/______

Transport Layer

TCP	UDP
Resend ↑ (Error detection) Reliable	Unreliable
(3-way handshaking) Conn^n oriented	conn^n less
High overhead	less overhead
slow	fast

- flow control (tries to ~~reduce~~ synchronise speed of sender & reciever)
  - → lessens data loss
  - → lessens network traffic

- congestion control (so that the packet reaches to the reciever within the specific time).
  - high RTT / packet loss
  - → applied by sender to reduce the ~~no~~ traffic on the network.

- Byte stream protocol.
  - ↳ each & every byte identified through sequence no.

→ (Multiplexing): Server side

- Application layer sends the file in form of a message to transport layer.

Multiplex-ing
- Then it divides the message into segments & puts or attaches header info with ~~on~~ all the segments & sends it to network layer.

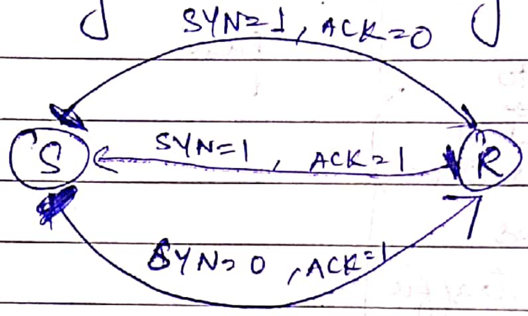
Page No.

Port No. : To recognize different applications.

Date ___ / ___ / ______

→ Reciever side : (Demultiplexing)
Transport layer takes data from network layer, in forms of segments. Attaches them together & make the file. Then sends it to the application layer.

→ 3- way handshaking :



→ TCP header :

<table>
<tr><td colspan="8">source port (16 bit)</td><td colspan="2">destination port (16 bit)</td><td>- 4 bytes</td><td rowspan="5">20 byte fixed</td></tr>
<tr><td colspan="10">sequence no. (32 bit)</td><td>- 4 bytes</td></tr>
<tr><td colspan="10">Acknowledgement no. (32 bit)</td><td>- 4 bytes</td></tr>
<tr><td>Header length (4 bits)</td><td>Unused (6-bits)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">Advertised window (16 bit)</td><td>- 4 bytes</td></tr>
<tr><td colspan="8">Internet check sum (16 bit)</td><td colspan="2">Urgent data points (16 bit)</td><td>- 4 bytes</td></tr>
<tr><td colspan="10">option (0-40 byte)</td><td>- variable</td><td></td></tr>
<tr><td colspan="10">data</td><td></td><td></td></tr>
</table>

port no. = $(0 - (2^{16} - 1)) \rightarrow 0 - 1023$

Date ___ / ___ / ______

→ Sequence no. : ~~starts with~~ random initial sequence no

→ To differentiate b/w connections.

→ Suppose random sequence no. matches then, the connⁿ which reaches later would get dropped.

→ Every byte would have different sequence no.

→ Acknowledgment no. :

• what reciever is expecting to recieve the next sequence no.

→ Header Length :

(4-bit)

$$\begin{pmatrix} 0000 \\ \vdots \\ 1111 \end{pmatrix}_2 \Bigg\} \begin{pmatrix} 0 \\ \vdots \\ 15 \end{pmatrix}_{10}$$

NOTE: Why do we need header length in TCP header?

Sol: Option field is a variable field, so length of TCP header can vary from (20 – 60) bytes

To overcome the prob. of representation of header, we use scaling factor = 4.

Header length field max = 15
TCP Header length size max = 60 bytes

60/15 = 4 = scaling factor

Header length field = $(x)_2 = (y)_{10}$

Then Header length of TCP = $(y)_{10} \times 4$ bytes.

Date __ / __ / ____

$\therefore$ Length of option field $= [(y)_{10} \times 4] - 20$ bytes.

NOTE: ~~Since~~ Since, min value for header length of TCP is $20 = (5)_{10} \times 4$

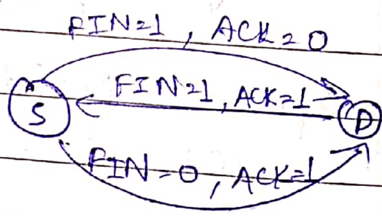
Thus, possible values for header length field can be from $(5-15)_{10}$

→ Unused (6-bits) : This field is still unused. Reserved for future usage.

→ Flags :

- URG – Urgent
- ACK – Acknowledgement – recieving acknowledgement
- PSH – Push
- RST – Reset
- SYN – Syn – used for connⁿ establishment.
- FIN – Fin – used for closing the connⁿ.

- Urgent pointer :
  Urgent data ⇒ URG = 1
  else URG = 0.



- PSH : Push :

(till ~~1500~~ 1500 bytes) A fixed size of data is sent to TCP. as sending small packets with 20 byte header size with each packet would be very inefficient.

Date __/__/__

Saathi

- RST - Reset :

Resetting the connⁿ.

- ~~Ctr~~.

→ Internet Checksum :
Characteristic of error detection and error correction.

(Total databytes sender side = Total databytes reciever side)

no error

else, error is there.

correction : resend the segment.

→ Urgent data Pointer : contains ending point of urgent data segment
A pointer that points to urgent data points

→ Option field :- Contains meta-data info.
like Max segment size
- variable sized (0 - 40 bytes)
- incase, no meta data info is there the option field contains 0 bytes.

→ Advertised window : Reciever window,

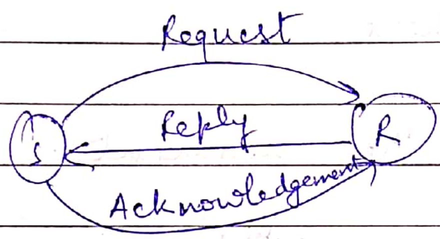
- Flow control : Don't overwhelm the reciever through too much of data.
- → Implemented through advertised window.
Contains buffer size of reciever - capacity

Date __ / __ / __

left at the reciever to take the data.

- Reciever sends it with the ACK packet.

- At handshaking time, Reciever sends the info, in accordance with that sender starts sending data.
Then as the reciever acknowledges that current data, in accordance with that sender adjusts the flow of remaining data packets



- Most of the time acknowledgement is sent with data packets. This is known as piggybacking.

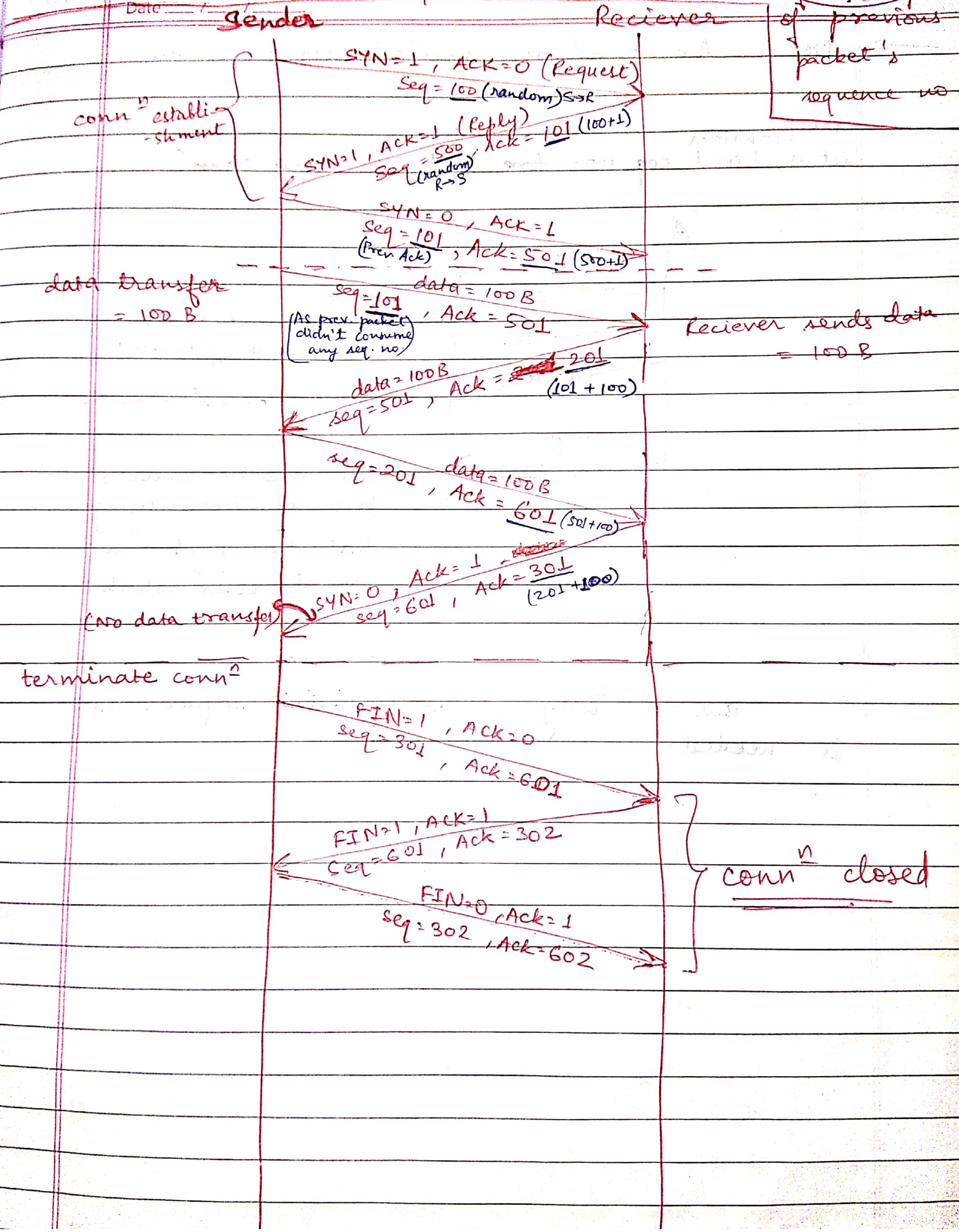
Imp

* Flow of sequence number $(0-(2^{32}-1))$

FACT: ① Request packet consume one sequence no.
② reply packet consume one sequence no.
③ acknowledgement packet doesn't consume any sequence no.

NOTE: In TCP, each & every byte is identified by its own seq. no. i.e - consumes one sequence no.

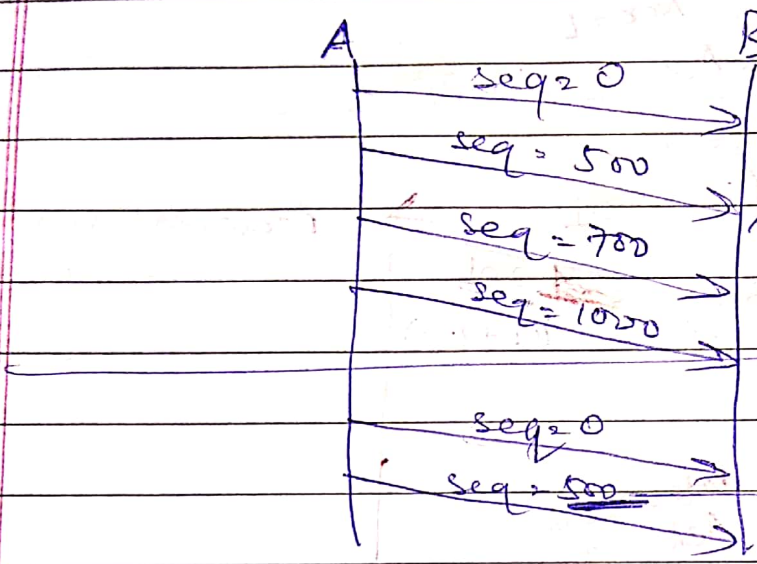
★ Reciever sends an ACK which it is expecting as a sequence no. next time. (Not random it calculates the next sequence no. on the basis of consumption of previous packet's sequence no.



Date ___ / ___ / ______

→ Need for random initial sequence no.

Proof: Let us consider Fixed seq. no. for each & every connⁿ.



This could not reach because of timeout

connⁿ closed.

This would occur a prob. of identifying that this packet is of old connⁿ which got lost or new connⁿ.

This is why random initial sequence no. is needed.

Date ___/___/___

Q- The following is a dump of TCP header in a hexadecimal format.

05320017 00000001 00000000

5002 07FF 00000000

i- destination port
ii- seq. no.
iii- length of header (size)
iv- Type of segment
v- Adv. window

$\underbrace{0532}_{\text{source}}\ \underbrace{0017}_{\text{destination}}$

$\underbrace{0000\ 0001}_{\text{sequence}}$

$\underbrace{0000\ 0000}_{\text{Ack}}$

~~5002007FF~~

Unused

000000000010

$\underbrace{5}_{\text{header length}}\ 0\ 02\ \underbrace{07FF}_{\text{Advertised window}}$

$5 \times 4 = 20$ = header length

$(5)_{16}$ =

5

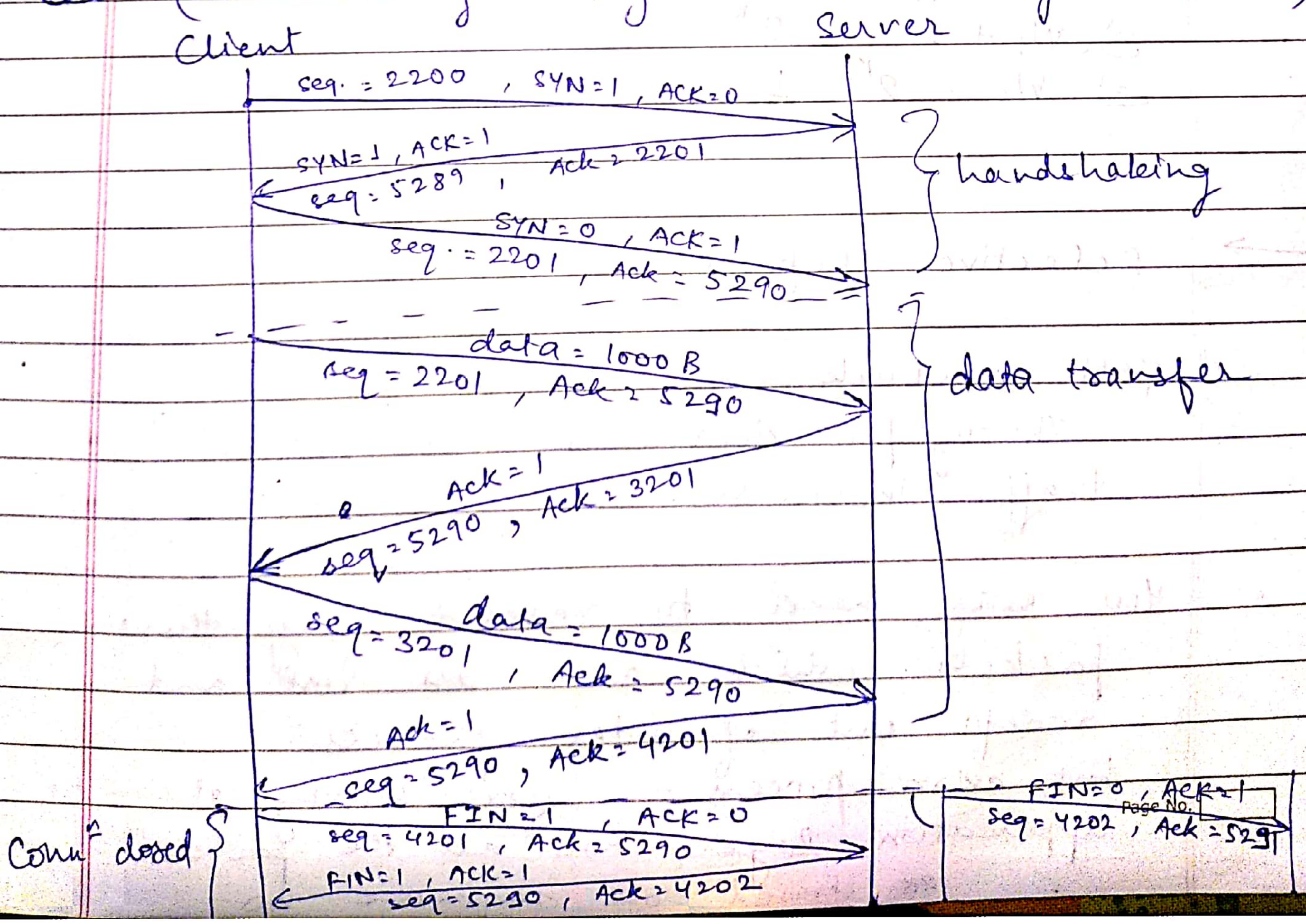
Date __/__/__

Ques- In a segment of TCP option field is 16 bytes long. Find out the ~~valte~~ value of ~~HLEN~~, HLEN, header length field.

$20 + 16 \rightarrow 36/4 \rightarrow (1001)_2$

Ques- In a TCP connⁿ, the initial sequence no from client side is 2200 & server side is 5289, the client opens a connⁿ & sends two segments carrying 1000 Bytes of data each. & closes the connⁿ. What is the value of sequence no. in each of the following cases?

i- Handshaking
ii- Data Transfer
iii- (Use timing dig. to show your ans.)



Scanned by CamScanner

Date __ /__ /____

Sliding Window Protocol

→ Go back N :
- inorder delivery
- cumulative acknowledgement
- discard out of order packets
- sender window size = N,
  reciever window size = 1.

- $W_S + W_R \leq$ Available seq. no.

- $N+1$ = total sequence no. in GBN

⇢ Suppose K is max. no. of "bits" available for sequence no. then sender window size will be =

$$W_S + W_R = 2^K$$
$$\Rightarrow W_S + 1 = 2^K$$
$$\Rightarrow W_S = 2^K - 1$$

→ Selective Repeat

• GBN drawback :
  Throughput decreases, as network traffic increases.

• You ~~only~~ need to resend only those packets which are ~~dis~~ lost and accept out of order packets.
but extra process - before sending them to application layer, sort all the packets ~~ac~~ based on sequence no.

Page No.

Scanned by CamScanner

Date ___/___/___

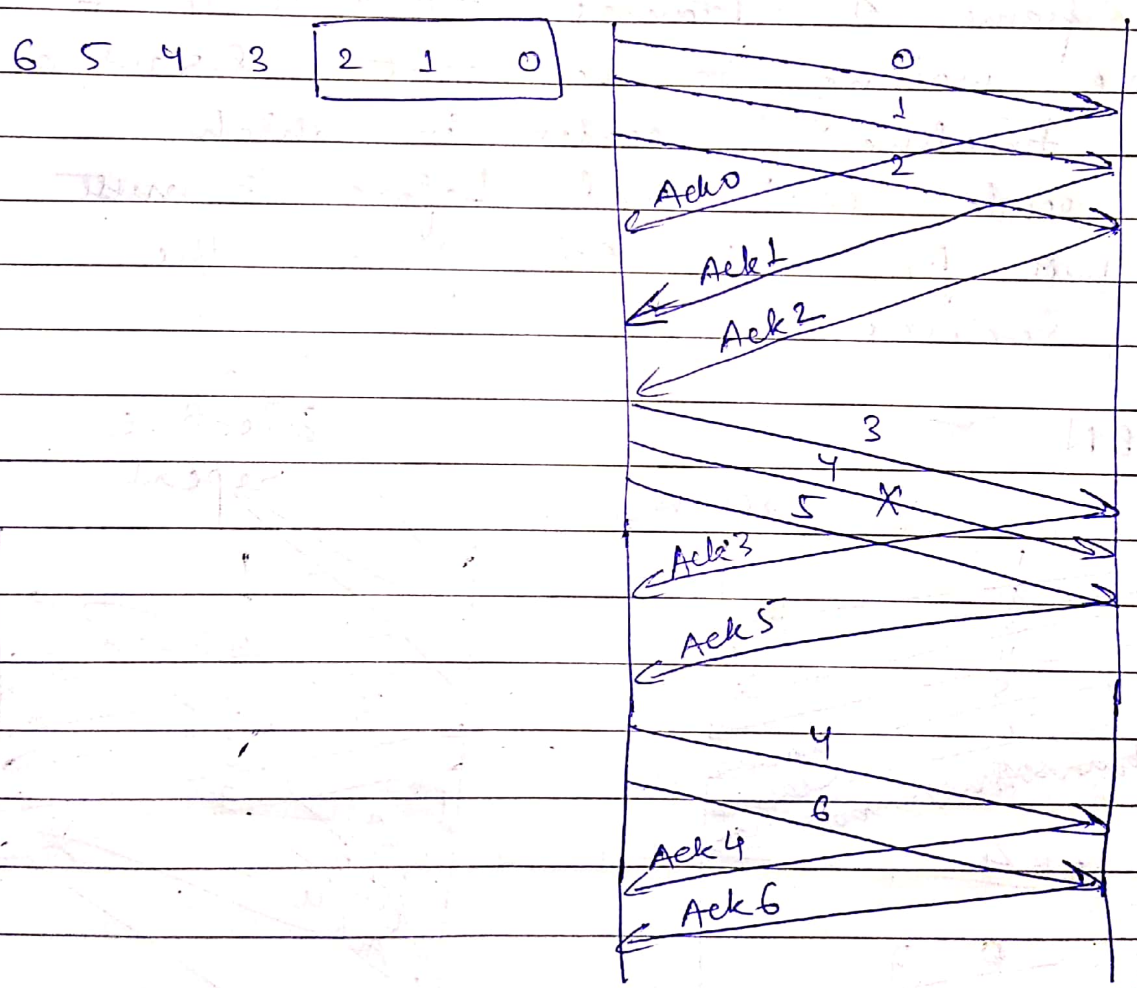
* **Selective Repeat**

GBN drawback : Repetition of packets due to discard

- accept out of order packet
- only resend those packets which got lost in N/W.
- sort all the packet after recieving at reciever side.
- use independent acknowledgement.
- sender window size = reciever window size

$W_S = W_R = N$

~~A~~ N=3



Scanned by CamScanner

Suppose K max. no. of bits available for seq. nos.

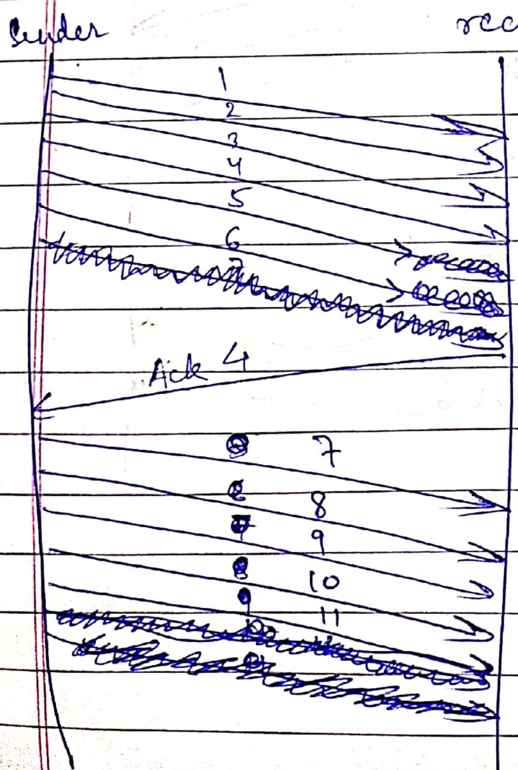
then $W_S + W_R = 2^K$

$\Rightarrow 2W_S = 2^K$

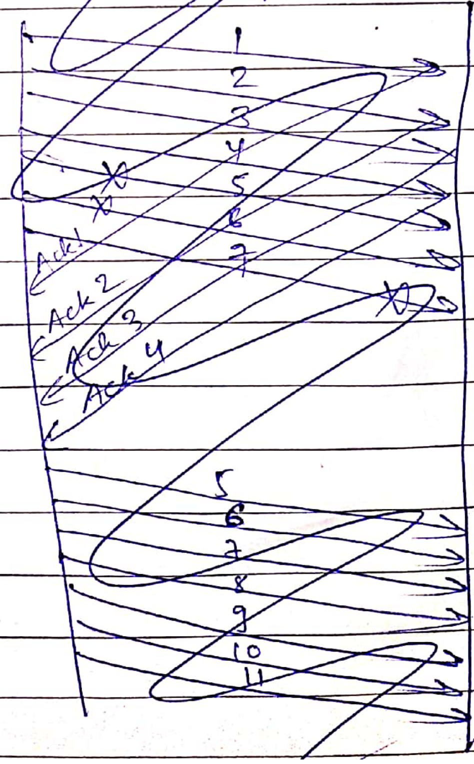
$\Rightarrow W_S = 2^{K-1}$

Ques- Go back N protocol, having a window size of 7, sending frames as: 1 2 3 4 5 6, after sending the frame sender recieved an Ack for frame 4. Frames 7 8 9 10 11 12 are waiting to be sent. Draw a timeline for sender in which sender can send before it must wait for next Ack from the reciever.

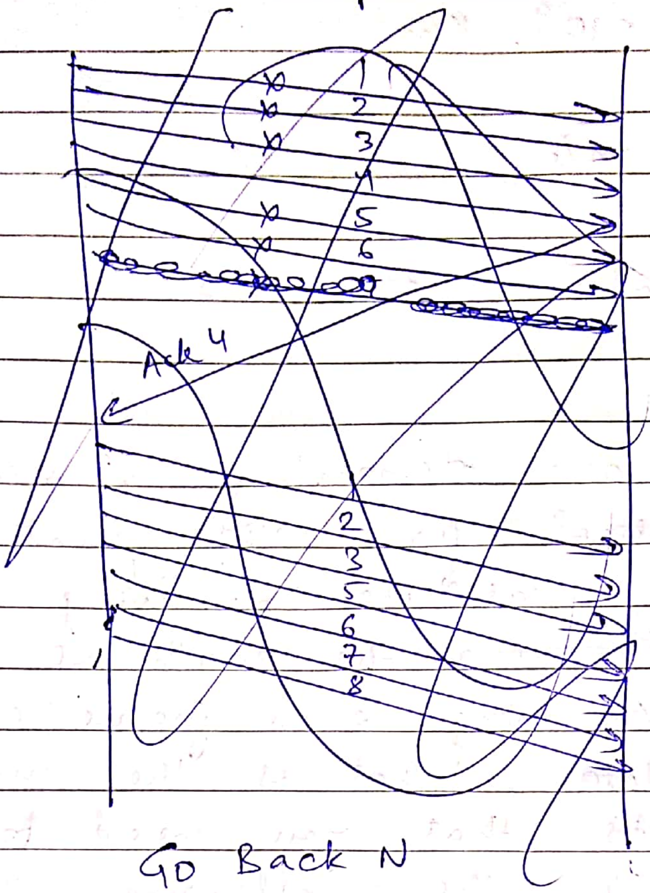
GBN



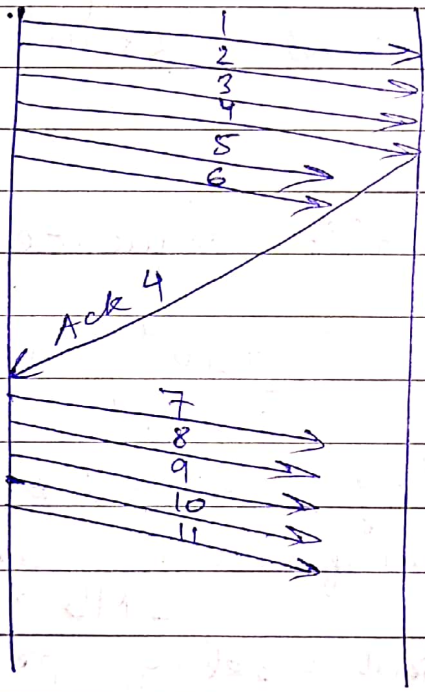
Selective repeat



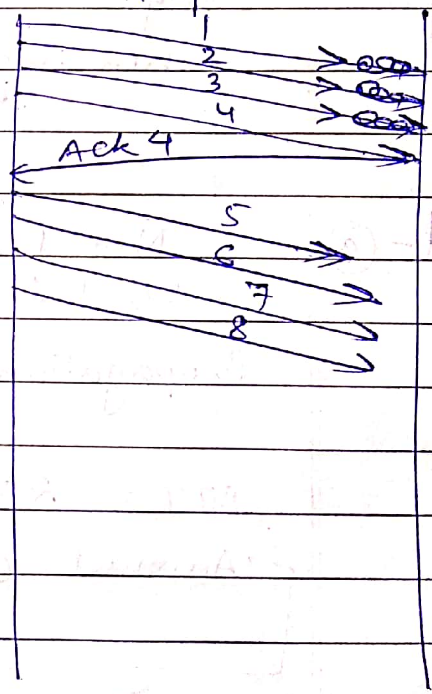
Selective repeat



Go Back N



Selective repeat



NOTE: Until & unless it is mentioned that packet got lost, we cannot assume 'it'.

Date ___/___/_____

Q- $BW = 1\ Mbps$

Altitude $= 3 \times 10^4$ Km.

$L = 1\ KB$

$S = 3 \times 10^8\ m/s$.

(a) Reciever window size $= 1$

(b) $W_R = W_S$

Suppose you are designing a sliding window protocol for 1 Mbps point to point to stationary satellite revolving at around ~~altitude~~ ~~Assuming~~ the earth at 3×10^4 KM altitude. Assuming each frame carrying 1 KB of data, what is the min. no. of bits that you need for the seq. no. in the following cases. assuming signal speed $= 3 \times 10^8$ m/s.

Sol- (a) $N + 1 \leq$ Available sequence no.

$= W_S + 1$

Propagation delay $= \dfrac{3 \times 10^4 \times 10^3}{3 \times 10^8}\ s = 0.1\ s$.

RTT $= 2 \times$ Prop. delay $= 2 \times 0.1\ s = 0.2\ s$.

Amount of data transferred in $0.2\ s$ =

$1\ Mb \times 0.2 = 0.2\ Mb$.

(Bandwidth - delay product)

(b) ~~$W_S + W_R \leq$ Available seq no.~~

~~$= 2W_S \leq$ Available seq no~~

Date __/__/__

$$\frac{\cancel{0.2}\ \cancel{MB} \times 10^5}{1 \times 10^3 \times 8} \qquad \frac{1.2 \times 10^{5}}{48 \times 10^3} = \frac{\cancel{100}^{25}}{\cancel{4}}$$

$\Rightarrow W_S = 25$

(a) If Available seq. no. $= 25 + 1$
$= 26$

$\therefore$ No. of bits $= 5$

(b) $W_S = 25 = W_R$

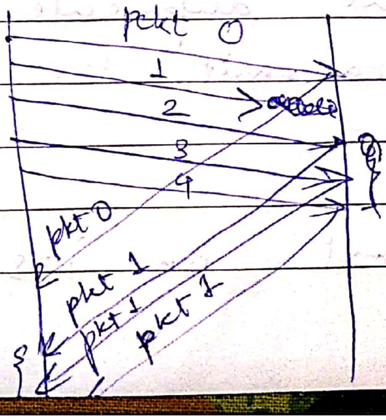
If $W_S + W_R = 50$
If No. of bits $= 6$

* Fast Retransmission (3 duplicate Ack)

We have different timers in TCP, e.g. - timeout.

timeout = 2 * RTT

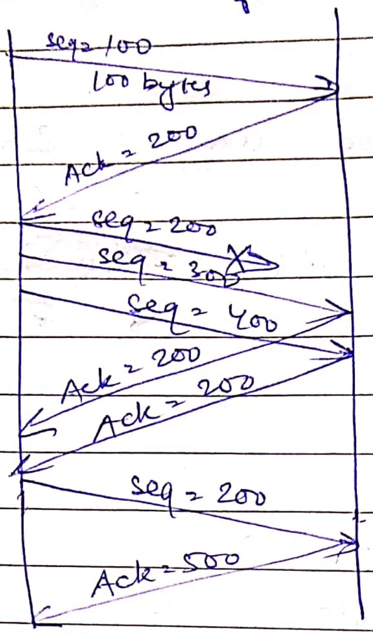
- If we don't get acknowledgement even after timeout, then we would resend the packet.
- I see whether I am getting a duplicate ack.



duplicate acks.

Date __ / __ / ____

- As soon as we recieve 3 duplicate acks, then we retransmit the packet irrespective of the timeout.



NOTE: If last packet got lost then we need to wait for the timeout.

Q- Draw a timeline dig. for sliding window algo. with $W_S = W_R = 3$ for the following -

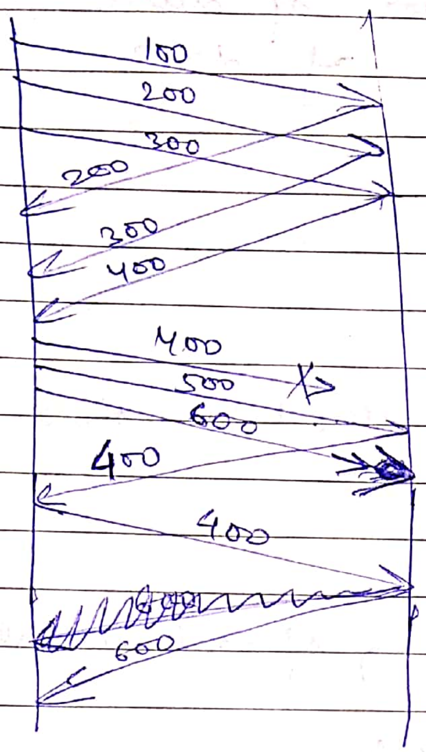
- Use a timeout interval of $2 * RTT$ Assume fast retransmission with selective retransmission is implemented & that packet with seq. no. 1 - 6 will be sent for fast retransmission if more than one duplicate Ack is recieved. Only retransmit for 1st duplicate Ack!

a - 4 is lost on its first transmission

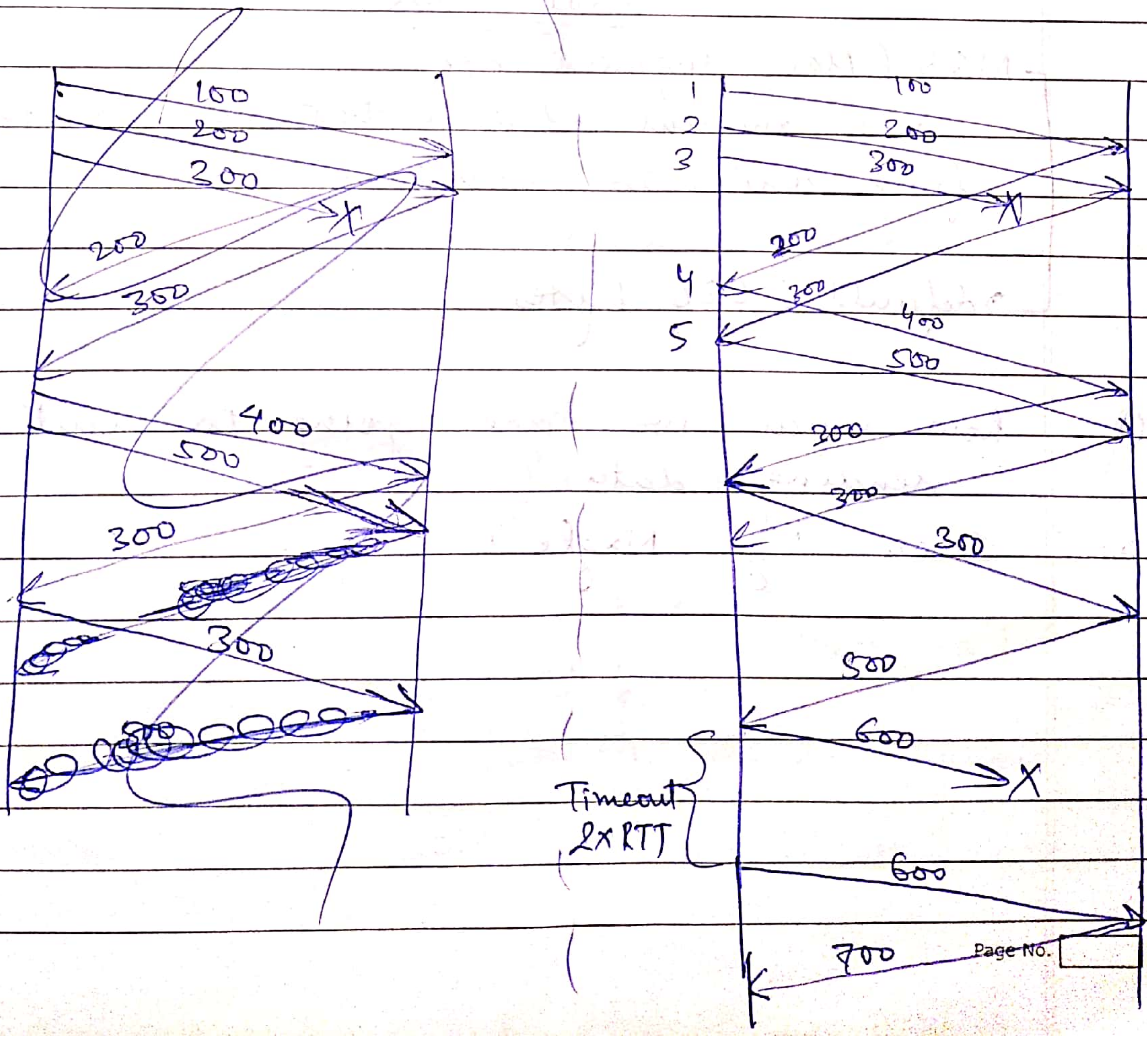
Date __/__/____

b) 3 & 6 got lost in their first transmission.

(a)



(b)



Page No.

Flow control

Silly window syndrome :
sending 41 bytes of data for sending 1 byte of data.

data	TCP	IP
1 byte	20 bytes	20 bytes.

→ Sender : sending speed is slow.

→ Reciever : recieving speed is slow.

<u>Sender side</u>

MSS (Max. segment size)
max. amount of data that can be sent from sender to reciever.

→ default : 536 bytes

Q- how much we are going to wait for sending data ?
A: Given by Nagle's algo.

Date ___/___/______

NAGLE's ALGORITHM

* For first segment send the data, irrespective of segment size.

* If there is an unacknowledged data or data in flight, then wait for either ack or for 1MSS of data.

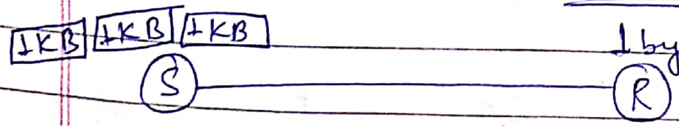
if (1st transmission)
    send packet

else if (unacknowledged data)
    wait for 1 MSS

else
    send data.

NOTE: if first packet got lost, then wait for retransmission (timeout)

Reciever side



Reciever buffer full
Advertisement window = 1

wait until it becomes larger
(Clark's solution)

Saathi

Date ___ / ___ / ______

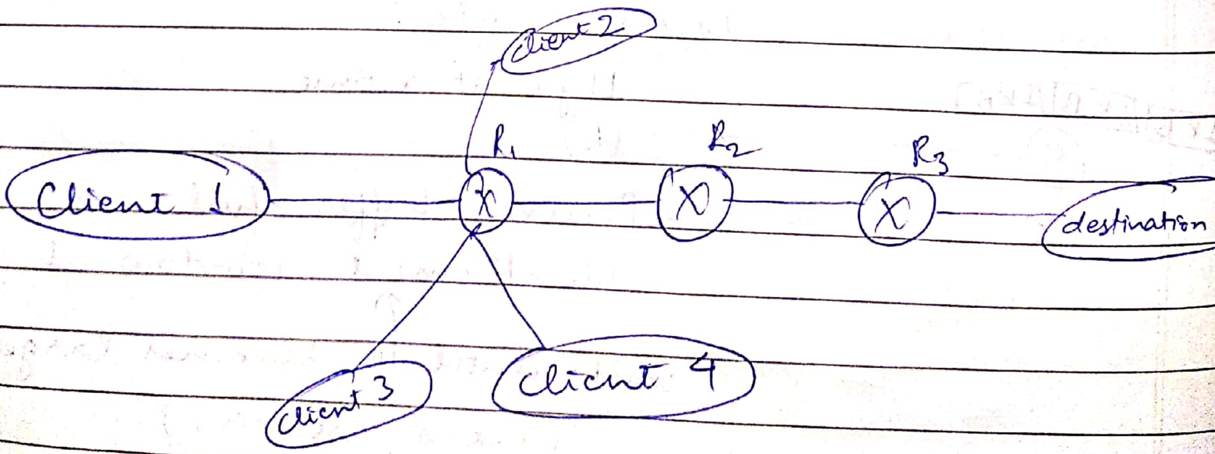
→ Clark's Sol. :

delay acknowledgement if advertisement window ~~is~~ = near zero. until either to accomodate 1 MSS or until atleast half of reciever buffer is empty.

→ Advantage of Clark's & Naggle's sol. :
- reduces traffic.
-

→ Disadvantage :
- if timeout happens, then you need to retransmit, although reciever has recieved the packets but delayed ~~ack~~. ack.

* Congestion control
- duplicate acks. (retransmission)
- ~~ack~~ time for acks. increases



Page No.

Scanned by CamScanner

Header doesn't contain any field to guess congestion on the traffic

Date ___ / ___ / ______

processing delay : checks for error + sab kuch theek hai ya nhi packet mein.

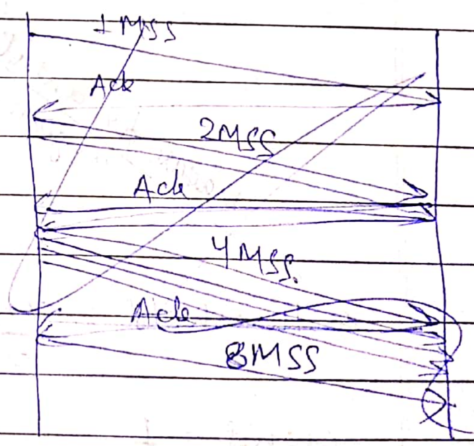
strategy : reduce no. of packets

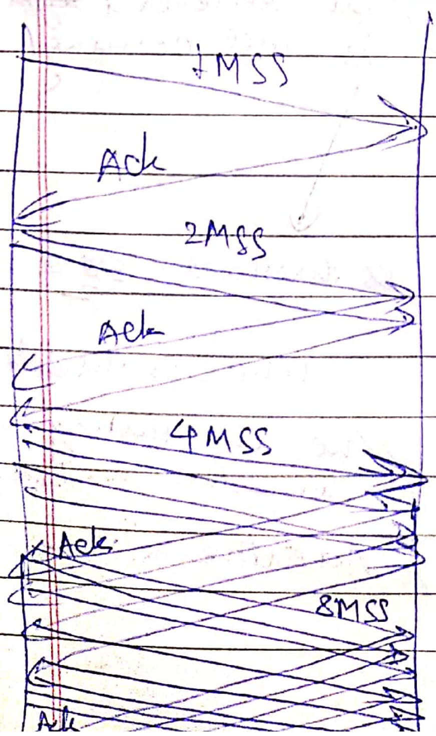
congestion window - resides at the sender.
→ perception of congestion in the network.

3 phases :- ① slow start → actually its a speedy start
② Congestion avoidance
③ Congestion detection.

→ upto a certain threshold (SS threshold (slow start)), we inc↑ congestion window size exponentially.

cwnd = 1MSS (536 bytes) default (initially)
congestion window





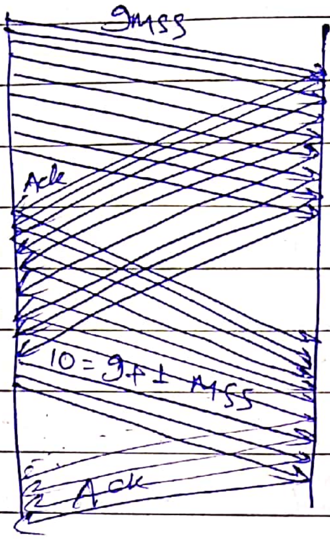
SS threshold = 8 MSS (here)

Date ___/___/______

→ As soon as we reach threshold, second phase starts.

Now cwnd & increases one-by-one.
(additive increase)
(until any packet gets lost)

So now, cwnd becomes 8+1 = 9.



Retransmission → RTO timeout (stronger reason for congestion ~~control~~)

Retransmission → Fast Retransmission
(sending segment ~~was~~ gets lost, but reciever's ~~segmt~~ ack is still coming)

RTO timeout → measure for controlling congestion

ss threshold = $\frac{cwnd}{2}$

cwnd = 1

// we are starting slow start phase again

// exponential inc ↑.

Fast Retransmission →

ss threshold = $\frac{cwnd}{2}$

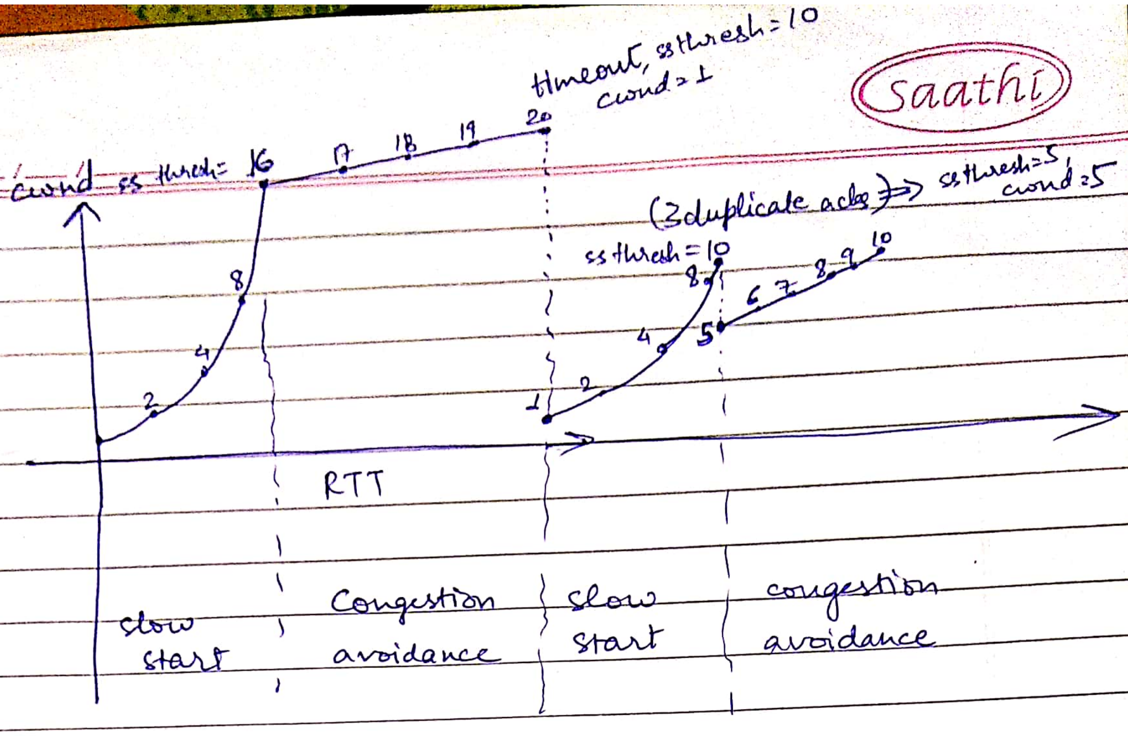
cwnd = ss threshold

// we are starting congestion avoidance

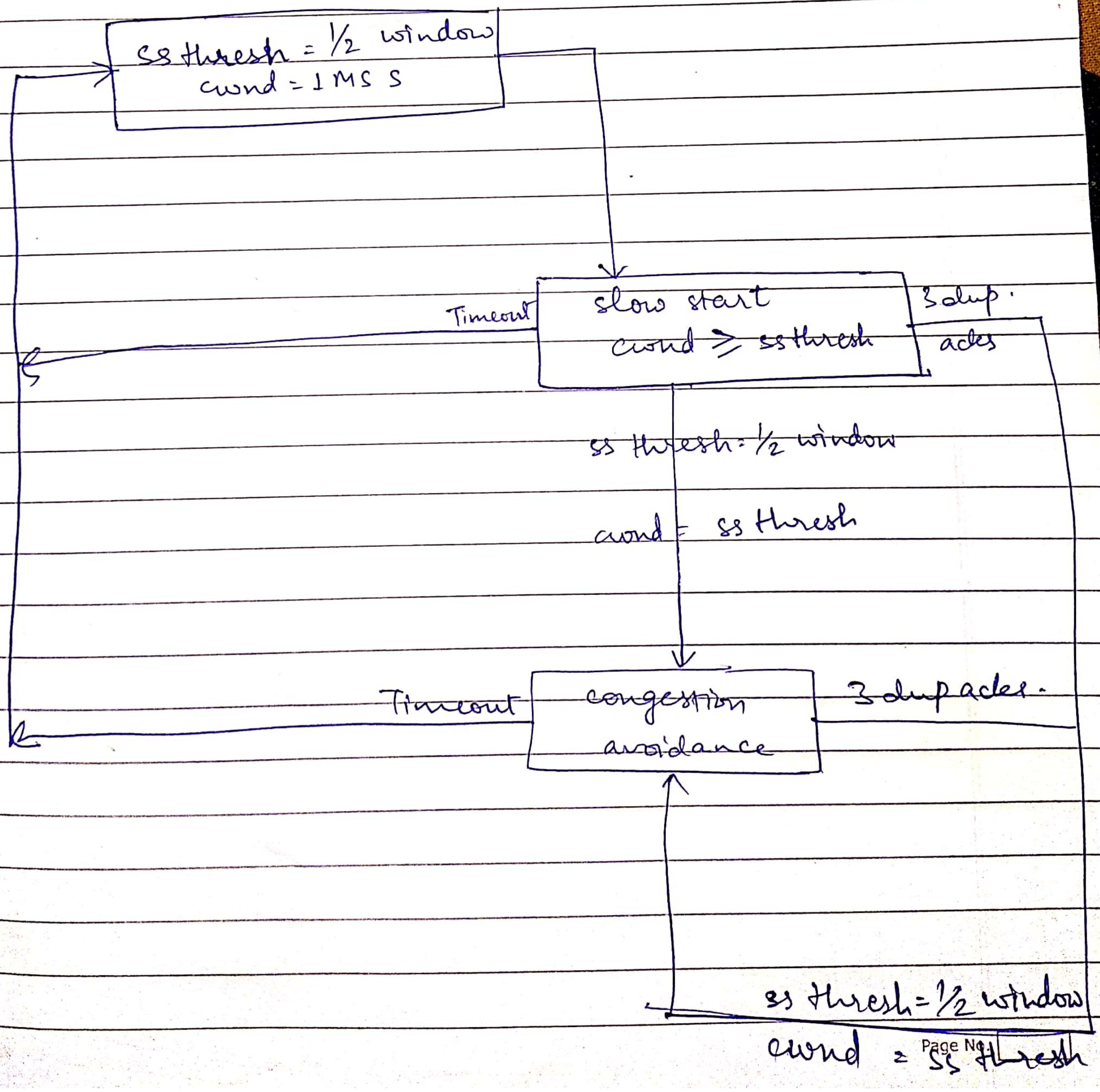
// Additive inc ↑ only

Scanned by CamScanner

Date



timeout, ssthresh = 10
cwnd = 1
cwnd
ssthresh = 16
17
18
19
20
2
4
8
(3 duplicate acks) ⇒ ssthresh = 5, cwnd = 5
ssthresh = 10
1
2
4
8
5
6
7
8
9
10
RTT
slow start
Congestion avoidance
slow start
congestion avoidance



ss thresh = 1/2 window
cwnd = 1 MSS
slow start
cwnd ≥ ssthresh
Timeout
3 dup. acks
ss thresh = 1/2 window
cwnd = ss thresh
congestion avoidance
Timeout
3 dup acks.
ss thresh = 1/2 window
cwnd = ss thresh